UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica e dell'Automazione

TESI DI LAUREA

Progettazione e implementazione di case study relativi all'Intelligenza Artificiale Generativa: generazione di testi, immagini e codice

Design and implementation of case studies related to Generative Artificial Intelligence: text, image, and code generation

Relatore

Prof. Domenico Ursino

Candidato

Enrico Fabio Ciavaroli

ANNO ACCADEMICO 2025-2026

Stiamo dando uno sguardo solo a pochi anni nel futuro,
ma possiamo vedere già molto che deve essere fatto.

Alan Turing

Sommario

Negli ultimi anni L'Intelligenza Artificiale Generativa è diventata una costante nella vita professionale e personale della maggior parte degli individui con accesso ai comuni mezzi di informazione. Al giorno d'oggi, infatti, molte persone ricorrono a questo strumento sia come fonte di apprendimento che come fonte di supporto per lo svolgimento di mansioni complesse. Tra i vari fattori che continuano a favorire la rapida espansione di questo strumento vi è anche la sua notevole facilità nell'utilizzo: un semplice prompt per formulare una richiesta espressa in linguaggio naturale al fine di ottenere la generazione di qualsiasi contenuto desiderato in maniera autonoma. Lo scopo della presente tesi è quello di indagare sulle potenzialità di questo settore, evidenziandone quelli che sono, al giorno d'oggi, i punti di forza e le limitazioni.

L'analisi è stata condotta attraverso lo sviluppo di alcuni case study applicati a cinque modelli tra i più utilizzati: *ChatGPT, Gemini, DeepSeek, Grok* e *Claude*. Questi case study riguardano la generazione di testi, immagini e codice con lo scopo di fornire un'analisi approfondita su un contesto ampio.

Keyword: Intelligenza Artificiale Generativa, ChatGPT, Gemini, DeepSeek, Grok, Claude, Case Study

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

Negli ultimi anni l'Intelligenza Artificiale è al centro dello sviluppo industriale che stiamo vivendo, influenzando su larga scala la società odierna. Tra i vari campi dell'Intelligenza Artificiale, quello generativo è statisticamente il più utilizzato. Esso offre la possibilità di ottenere una moltitudine di elementi tramite il semplice utilizzo di una richiesta espressa in linguaggio naturale.

La presente tesi nasce con l'obiettivo di analizzare questo strumento, in modo da poterne evidenziare benefici e limiti. È, infatti, evidente come l'utilizzo della generazione di contenuti possa essere utile per innumerevoli mansioni. Tuttavia, pur essendoci ampie capacità di calcolo dietro la generazione di questi contenuti, è importante sottolineare come anche queste macchine siano in grado di commettere errori e che l'eccessivo affidamento ad esse è altamente sconsigliato.

I modelli scelti per l'analisi sono: *ChatGPT, Gemini, DeepSeek, Grok* e *Claude*.

Questi modelli appartengono a categorie diverse. Alcuni di questi hanno politiche di regolamentazione delle risposte più rigide, altri, invece, dispongono di un'accuratezza minore nella generazione delle risposte.

L'obiettivo è valutare questi strumenti in diversi scenari applicativi e confrontarli successivamente in modo da ottenere un quadro accurato su quali modelli risultino più adatti a seconda della mansione richiesta e su quali, invece, porre maggiore attenzione in caso si decida di utilizzarli.

La presente tesi è composta da sei capitoli strutturati qui di seguito specificato:

- Nel Capitolo 1 sarà introdotta la definizione di Intelligenza Artificiale Generativa; successivamente, saranno riportate le differenze tra l'Intelligenza Artificiale Generativa e l'Intelligenza Artificiale classica. Verranno enunciati, poi, i principi alla base del suo funzionamento e alcune considerazioni sui rischi, anche a livello etico.

- Nel Capitolo 2 si porrà l'attenzione principalmente su come classificare i modelli di Intelligenza Artificiale Generativa secondo criteri come il tipo di output generato, l'architettura alla base del loro funzionamento e, infine, l'accessibilità.

- Nel Capitolo 3 verrà riportato il case study relativo alla generazione di testo, ponendo l'attenzione del lettore su come due modelli di Intelligenza Artificiale Generativa interpretino ed elaborino le richieste mettendo a confronto l'accuratezza del loro output. Verrà, inoltre, testato il sistema di censura presente in uno dei modelli presi in esame.

- Nel Capitolo 4 verrà riportato il case study rigurdante la generazione di immagini. I tre modelli presi in esame verranno sottoposti a prompt con lo scopo di evidenziare le

loro difficoltà nel rispettare i parametri realistici notoriamente ostici a questa branca dell'Intelligenza Artificiale Generativa.

- Nel Capitolo 5 verrà riportato il case study riguardante la generazione di codice. In esso verranno riportate, prompt dopo prompt, le scelte procedurali del modello preso in esame nella creazione di un programma complesso e, successivamente, le righe di codice generate con il risultato finale.
- Nel Capitolo 6 sarà proposta una discussione in merito al lavoro svolto, ponendo l'attenzione del lettore sui case study svolti e delineando i pregi e i limiti attuali che i modelli scelti hanno evidenziato.
- Nel Capitolo 7 verranno tratte le conclusioni e verranno delineati alcuni possibili sviluppi futuri.

CAPITOLO 1

Introduzione all'Intelligenza Artificiale Generativa

In questo capitolo si fornirà un'introduzione completa all'Intelligenza Artificiale, cercando di fornire le basi teoriche del suo funzionamento e sottolineando quali siano le differenze con la sua declinazione generativa.

Verranno successivamente esaminate le architetture principalmente in uso, grazie alle quali questo strumento è in grado di generare contenuti in maniera autonoma. Verrà sottolineato il concetto di prompting, ovvero il mezzo attraverso cui l'utente può interagire con questi sistemi attraverso richieste espresse in linguaggio naturale influenzando direttamente l'output.

Il capitolo si conclude con una riflessione sulle sfide per l'essere umano che derivano dall'utilizzo di queste tecnologie e su quali siano i rischi sia a livello industriale ed etico connessi all'utilizzo su larga scala di questi strumenti.

1.1 Definizione di Intelligenza Artificiale

L'Intelligenza Artificiale è un insieme di tecnologie che consente a una macchina di apprendere, ragionare ed eseguire attività complesse nelle modalità che, un tempo, solo l'intelletto umano era in grado di svolgere. Il Parlamento UE ha definito l'Intelligenza Artificiale come la capacità o il tentativo di un sistema artificiale di emulare capacità umane quali apprendere, ragionare, pianificare e, perfino, mostrare creatività.

Si può, quindi, concludere che il fine dell'Intelligenza Artificiale, nelle sue varie forme, è quello di svolgere mansioni in maniera autonoma dove altrimenti sarebbe richiesto l'intervento umano.

1.2 Differenze tra Intelligenza Artificiale Generativa e Classica

L'Intelligenza Artificiale classica si basa su sistemi progettati per svolgere mansioni seguendo regole precise e ben definite dal programmatore. Questi modelli non sono in grado di ampliare le proprie conoscenze al di fuori dei domini per cui sono stati sviluppati. Ad esempio, i motori di scacchi come *Stockfish* sono in grado di analizzare centinaia di milioni di posizioni al secondo, ma non possono approfondire la loro capacità di calcolo per svolgere mansioni al di fuori dei domini per cui sono stati sviluppati.

L'Intelligenza Artificiale Generativa, partendo da questo concetto, evolve il proprio funzionamento. Modelli di IA Generativa non operano secondo regole esplicite ma, partendo dall'analisi del prompt e, successivamente, analizzando grandi quantità di dati su cui vengono

addestrati, apprendono vari pattern su cui operare e li usano come base per generare contenuti nuovi e originali.

A differenza dell'IA Classica, quindi, l'IA Generativa non produce il proprio output in maniera deterministica e vincolata dal proprio dominio di sviluppo, ma bensì risponde in maniera probabilistica e adattiva a una quantità illimitata di richieste espresse in linguaggio naturale.

1.3 Architetture alla base dei modelli generativi

Dato il complesso meccanismo dietro la generazione di queste risposte, il funzionamento dei modelli di Intelligenza Artificiale Generativa è dovuto a un'architettura robusta in grado di analizzare una ingente quantità di dati ed integrare feedback su cui addestrare i modelli stessi.

Di fatto, l'Intelligenza Artificiale Generativa fonda il proprio funzionamento su una rete neurale, ovvero un insieme di modelli computazionali il cui funzionamento è ispirato alle funzioni cerebrali dell'essere umano. Queste reti sono composte da molteplici layer (strati) di neuroni che processano e trasmettono informazioni. Esistono tre tipi di layer sulle reti neurali, come mostrato nella Tabella 1.1.

Layer	**Funzione**	**Esempi di dati trattati**
Input Layer	Riceve i dati grezzi e li converte in formato numerico	Pixel, token testuali, valori numerici
Hidden Layer	Estrae caratteristiche tramite pesi e funzioni di attivazione; possono essere presenti più strati in cascata	Vettori numerici trasformati ad ogni strato
Output Layer	Produce il risultato finale in base al compito	Valore continuo e sequenza di token

Tabella 1.1: Tipologie di layer nelle reti neurali artificiali

Solo questo, però, non basta per garantire un output sempre conforme e originale alle, potenzialmente infinite, richieste che gli utenti possono porre. Di fatto i modelli di IA Generativa devono essere addestrati su grandi insiemi di dati. Il processo di addestramento, dunque, è la fase in cui avviene l'apprendimento. Nel *Machine Learning* l'apprendimento implica la regolazione dei parametri di un modello con l'obiettivo di produrre output sempre più accurati.

Dal punto di vista matematico l'obiettivo di questo apprendimento è quello di ridurre al minimo una *funzione di perdita*. Grazie a questa funzione possiamo quantificare l'errore degli output. Infatti, quando l'output della funzione di perdita scende al di sotto di una soglia predeterminata significa che l'errore è sufficientemente piccolo, per cui il modello può considerarsi addestrato.

Nella Figura 1.1, viene riportato il diagramma della funzione di perdita nell'addestramento di una rete neurale.

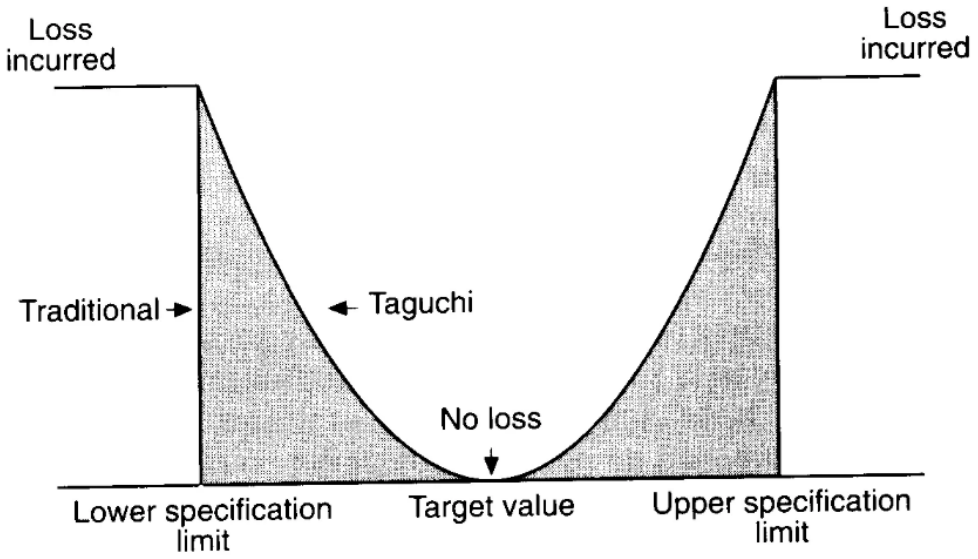

Figura 1.1: Funzione di perdita nell'addestramento di una rete neurale

Nella pratica, l'addestramento dei modelli segue un ciclo composto dalle seguenti azioni:

- raccolta dei dati;
- esecuzione del modello sui dati raccolti in modo da effettuare il suo addestramento;
- misurazione delle perdite;
- ottimizzazione dei parametri;
- test delle prestazioni del modello.

Questo ciclo procede in modo iterativo fino al raggiungimento di dati soddisfacenti.

1.4 Il prompting come strumento di interazione con i sistemi generativi

Come precedentemente riportato, il prompt è lo strumento principale con cui l'essere umano è in grado di interagire con i modelli di Intelligenza Artificiale Generativa. Il prompt si presenta comunemente come un campo di input, tipicamente testuale, attraverso cui è possibile esprimere una richiesta in linguaggio naturale. La regola di base è che istruzioni di qualità producono risultati di qualità. A questo scopo la disciplina della modellazione delle richieste atta a influenzare direttamente la pertinenza e la qualità delle risposte è il *prompt engineering*.

1.4.1 Perché il prompt engineering è fondamentale?

Il prompt engineering svolge il ruolo chiave nell'influenzare direttamente l'accuratezza degli output; il congegnare adeguatamente un prompt è la chiave per sbloccarne il pieno potenziale.

Via via che i sistemi di AI Generativa diventano sempre più persistenti nel mondo lavorativo in tutti i settori, la capacità di essere compresi e far produrre risposte significative sta diventando sempre di più una competenza ricercata.

1.4.2 Quali sono le competenze del prompt engineer?

Per sviluppare nuovi contenuti originali o risolvere problemi complessi, le competenze che i prompt engineer dovrebbero avere includono:

- forti capacità comunicative;
- capacità di spiegare concetti tecnici;
- competenze di programmazione;
- solida conoscenza delle strutture e degli algoritmi su cui si va ad operare;
- valutazione realistica dei benefici e dei rischi delle nuove tecnologie.

1.5 Sfide, limitazioni e rischi

1.5.1 Sfide e limitazioni

L'IA Generativa ogni giorno di più affronta diverse sfide e limitazioni, tra cui:

- *Requisiti delle risorse*: i modelli di IA Generativa richiedono una capacità di calcolo ed energia per funzionare notevole, rendendo il loro addestramento dispendioso con un impatto ambientale rilevante.
- *Complessità della creatività*: L'IA Generativa non riesce ancora a emulare in maniera efficiente la complessità della creatività umana.
- *Errori di generazione*: I modelli di IA Generativa possono produrre contenuti plausibili ma falsi. È molto ricorrente che le risposte di questi modelli, a primo impatto, possano risultare conformi alle richieste ma, indagando nei dettagli dell'output, si riscontrino errori di ogni tipo, dalla generazione di formule sbagliate alla generazione di codice errato. Insomma, è importante fare in modo che le creazioni di questi strumenti vengano sempre sottoposte al giudizio dell'umano onde evitare che errori di generazione della macchina diventino errori di negligenza dell'utente.

1.5.2 I rischi dell'IA Generativa

Man mano che i modelli di IA Generativa si diffondano nella società odierna diventa sempre di più importante sottolineare come queste nuove tecnologie non siano prive di rischi. I vari rischi includono:

- *Generazione di deepfake*: l'IA generativa ha reso più accessibile la creazione e la diffusione di deepfake. Un deepfake, per essere definito tale, deve ritrarre immagini false con persone reali. Questa pratica è estremamente pericolosa dal momento che i deepfake possono portare al furto d'identità sia di persone comuni che di celebrità di alto profilo, contribuendo a truffe o danni alla reputazione.

- *Attacchi di phishing* automatizzati: l'Intelligenza Artificiale Generativa è lo strumento maggiormente utilizzato da chi commette atti malevoli, come gli attacchi di phishing, rendendoli più avanzati e difficili da individuare. I sistemi di IA Generativa, infatti, sono in grado di produrre automaticamente su larga scala e-mail di phishing estremamente realistiche e personalizzate, includendo, addirittura, come mittenti persone reali con informazioni personalizzate, rendendo l'adescamento di click su link malevoli molto più semplice.
- *Generazione di codice dannoso*: sebbene possa essere uno strumento utile per la creazione di codice efficiente, l'IA Generativa può essere usata anche per la creazione di malware. Essa, infatti, è in grado anche di analizzare malware già esistenti per identificare modelli di attacco efficaci e generare nuove varianti in grado di mettere a dura prova i sistemi di sicurezza informatica attualmente in uso.
- *Campagne di disinformazione generate dall'Intelligenza Artificiale Generativa*: l'IA Generativa è in grado di produrre infinite quantità di testo coerenti e contestualizzate, il che la rende uno strumento potente per la disinformazione su larga scala. Le notizie false alimentate dall'IA Generativa possono essere utilizzate come strumento per influenzare l'opinione pubblica o per causare panico sui mercati.

CAPITOLO 2

Classificazione di sistemi di Intelligenza Artificiale Generativa

Il panorama dei sistemi di Intelligenza Artificiale Generativa è troppo ampio e variegato per poterlo classificare in un'unica categoria. Il presente capitolo tratterà la suddivisione di tali sistemi secondo criteri coerenti e complementari con l'obiettivo di fornire un quadro completo utile all'analisi comparativa che verrà trattata nei capitoli successivi

2.1 Criteri di classificazione dei sistemi generativi

I sistemi di Intelligenza Artificiale Generativa si presentano come una famiglia tecnologica estremamente eterogenea, per cui l'adozione di criteri di classificazione è un requisito fondamentale per poter fornire un'analisi comparativa rigorosa.

2.2 Classificazione per tipo di output: testo, immagini e codice

Un sistema di IA Generativa può essere modellato allo scopo di fornire diversi tipi di output. Tratteremo nello specifico: la generazione di testo in linguaggio naturale, immagini e sequenze di codice. Nella Tabella 2.1, vengono riportati i modelli usati per ciascuno di questi tipi diversi di output nei case study che l'elaborato tratterà nei capitoli successivi.

Tipo di generazione	**Modelli utilizzati**
Testo	ChatGPT (GPT-5.5), DeepSeek (V3.2)
Immagini	ChatGPT (GPT-5.5), Gemini (2.5 Pro), Grok (4.3)
Codice	Claude (Sonnet 4.6)

Tabella 2.1: Modelli generativi utilizzati per ciascun tipo di output

2.3 Classificazione per architettura

Questo tipo di classificazione è il più rilevante sul piano tecnico, uno concerne la lunga lista delle tecniche di apprendimento automatico su cui il modello è sviluppato. L'architettura, infatti, determina il processo attraverso cui il modello recepisce una distribuzione di

probabilità sulla larga scala di dati di addestramento a cui è sottoposto e, inoltre, determina il meccanismo con cui produrre nuovi campioni.

2.3.1 Large Language Model

I Large Language Model sono, ad oggi, la famiglia architetturale più diffusa nei sistemi di IA Generativa nel campo della generazione di codice e di testo. Essi si basano sull'architettura *transformer* e operano secondo una logica autoregressiva. Infatti, i modelli in questione generano un token alla volta, e ogni elemento prodotto nella sequenza di token verrà condizionato da tutti gli elementi prodotti in precedenza. Ciò è molto utile nello svolgimento di compiti che richiedono coerenza a lungo raggio.

2.3.2 Modelli diffusivi

I modelli di diffusione sono modelli generativi associati principalmente alla generazione di immagini e altri compiti di computer vision. L'intuizione alla base dei modelli di diffusione si ispira alla fisica, e tratta i pixel come gocce di inchiostro sottoposte a distorsioni casuali di rumore. A livello teorico, infatti, le gocce sottoposte a variazioni di rumore cambiano la loro forma e il loro posizionamento; e proprio su questo principio si basa il ciclo di funzionamento dei modelli diffusivi. Tale ciclo prevede le seguenti attività:

- *fase di addestramento*: l'introduzione casuale di rumore in un'immagine reale porterà a un danneggiamento dell'immagine stessa fino ad ottenere una figura confusa e senza senso, simile allo schermo della televisione fuori segnale.
- *fase di apprendimento*: data un'immagine danneggiata, la rete neurale viene addestrata a invertire il processo, ovvero imparare a prevedere quale distorsione è stata aggiunta in modo da poter ripulire pian piano il campione confusionario di partenza.
- *fase di generazione*: una volta che la rete neurale può ritenersi addestrata, partendo da un'immagine completamente distorta, la rete sarà in grado di rimuovere gradualmente il rumore passo dopo passo. Questo processo porterà alla generazione di qualcosa di coerente ottenendo un'immagine realistica non esistente in partenza.

2.3.3 Modelli multimodali

I modelli multimodali sono quei sistemi in grado di operare su più tipi di dati, che possono includere testo, immagini, video e forme alternative di input, come, ad esempio, audio e documenti. Grazie a questa varietà di fonti, i modelli multimodali sono in grado di raggiungere una maggiore accuratezza rispetto alle controparti unimodali (che operano solo su una tipologia di dati), rendendo, inoltre, l'interazione con l'utente più fluida e intuitiva. Di fatto, questi modelli sono in grado di recepire come input comandi vocali e segnali visivi, permettendo all'utente di descrivere problemi e contesti complicati con mezzi molto semplici. La sfida ingegneristica principale dietro questi sistemi è la capacità di accoppiare tipologie di input e output totalmente differenti. Ad esempio, nel momento in cui, partendo da un'immagine, si formula come richiesta la restituzione della descrizione del contenuto, la rete neurale dovrà scomporre l'immagine in una sequenza di token, campionando caratteristiche di natura cromatica e relazioni spaziali.

2.4 Classificazione per modalità di accesso

Un'ulteriore classificazione dei modelli di Intelligenza Artificiale Generativa riguarda l'accessibilità. Infatti, questa dualità delinea una distinzione netta riguardo alla disponibilità del codice sorgente.

2.4.1 Modelli open source

Il codice dei modelli di IA Generativa open source è reso disponibile al pubblico, permettendo a chiunque sia interessato di interagirci in modo collaborativo. L'accesso al codice sorgente da parte del pubblico gode di numerosi vantaggi:

- *maggiore flessibilità* nello sviluppo di sistemi personalizzati a seconda delle esigenze dell'organizzazione che ne fa uso;
- *trasparenza totale* agli occhi di chi vuole capire il funzionamento del modello stesso;
- *collaborazione tra diverse comunità* per l'individuazione e la correzione di imperfezioni tecniche;
- *possibilità* da parte di sviluppatori con risorse disponibili limitate di *implementare questi strumenti nei propri contesti di sviluppo*.

2.4.2 Modelli closed source

I modelli closed source sono sviluppati e gestiti soltanto da organizzazioni private e mantengono una netta distinzione tra utenti che hanno e non hanno i privilegi per poter accedere al codice sorgente. Questo implica che le modalità di funzionamento di tali modelli siano inaccessibile al pubblico ai fini di mantenere la segretezza che contraddistingue questa categoria dalla controparte open-source.

Nella maggior parte dei casi, le aziende proprietarie di questi modelli ne commercializzano l'utilizzo tramite piattaforme digitali dedicate, permettendo a utenti esterni di intravedere solo quello che è definibile come la "superficie" del prodotto (come la visualizzazione dell'interfaccia del prodotto e l'interazione con gli strumenti di input che esso fornisce), mantenendo la segretezza sui meccanismi con cui questi strumenti operano.

Le principali differenze tra modelli open source e closed source, vengono riportate nella Tabella 2.2

Caratteristica	Open Source	Closed Source
Implementazione	Lenta per configurare e addestrare il modello	Rapida grazie a ambienti di sviluppo già elaborati
Accessibilità	Totale a prescindere dalle esigenze dell'utente	Limitata ai vincoli che il produttore ha stabilito
Costi	Molto alti in fase di sviluppo ma attenuati nel lungo termine	Moderati a seconda dell'uso su larga scala
Sicurezza	Necessità di gestione autonoma	Centralizzata e fornita dall'ente produttore
Personalizzazione	Potenzialmente illimitata	Limitata alle opzioni che il produttore mette a disposizione

Tabella 2.2: Confronto tra modelli open source e closed source

CAPITOLO 3

Un case study relativo alla generazione di testo

In questo capitolo l'elaborato prenderà in esame una serie di prompt per la generazione di testo che verranno sottoposti a due diversi modelli di Intelligenza Artificiale Generativa (ChatGPT e DeepSeek). Verranno evidenziate le differenze nell'output generato dai due modelli sia dal punto di vista stilistico sia da quello dell'accuratezza rispetto alla richiesta originale.

Verranno, infine, posti prompt di natura politica, in modo da verificare se, nella generazione delle risposte di questi modelli intervenga qualsiasi tipo di bias ideologico o, addirittura, qualche forma di censura.

3.1 I modelli presi in esame

Per entrare nel dettaglio della tesi, procedendo con ordine, verranno introdotti i due modelli sottoposti allo studio. Entrambi sono *chatbot* di AI Generativa ed elaborano le loro risposte a seguito di richieste poste in linguaggio naturale. I modelli presi in esame sono:

- *ChatGPT*: rilasciato da OpenAI nel 2022, utilizza modelli basati sull'architettura Transformer, generativi e pre-addestrati (GPT, Generative Pre-trained Transformer) e, al momento, sfrutta il modello di AI Generativa multimodale GPT-5.5
- *DeepSeek*: rilasciato da DeepSeek AI nel 2023, nasce con l'obbiettivo di inserire la Cina nella gara di sviluppo di AI. Già dall'inizio, il modello ha catturato l'attenzione a livello mediatico grazie alla sua capacità di competere con le controparti occidentali. Fece particolarmente discutere il suo costo ridotto durante la fase di sviluppo; infatti i fondatori sono riusciti ad addestrare il modello con "soli" 6 milioni di dollari (uno scherzo se confrontati con i 540 milioni di dollari spesi da OpenAI prima di rendere pubblico ChatGPT). Seppur agli arbori il modello era in grado solo di generare testo, le versioni più recenti (come la Versione V3.2 adottata durante il case study) hanno implementato la multimodalità.

I modelli sono simili a livello di architettura; entrambi sono LLM multimodali e il loro addestramento si fonda su una base di reti neurali. La curiosità sorge, però, una volta che vengono evidenziate le loro differenze; i modelli sono stati sviluppati in due aree opposte del mondo e molto polarizzate a livello ideologico. Oltretutto, entrambi i modelli vengono usati come strumenti della loro fazione geopolitica per concorrere nello sviluppo delle tecnologie del futuro. Sarebbe troppo fantasioso dire che questo case study possa fornire un verdetto chiaro su quale sia il modello migliore per la generazione di testo o, addirittura, dichiarare chi tra Occidente e Oriente è in vantaggio in questa nuova "corsa alle armi". In definitiva, più che

dichiarare un vincitore, questo studio ha come obbiettivo sottolineare come due tecnologie, nate in contesti distanti, adoperino scelte progettuali totalmente diverse.

3.1.1 Interfaccia ChatGPT

Nella Figura 3.1 viene riportata l'interfaccia di ChatGpt; da sottolinerare lo stile pulito e accogliente. Il modello offre, come al solito, la possibilità di digitare una richiesta con, però, diversi tasti per ricevere richieste di diverso tipo. I tasti a disposizione sono i seguenti:

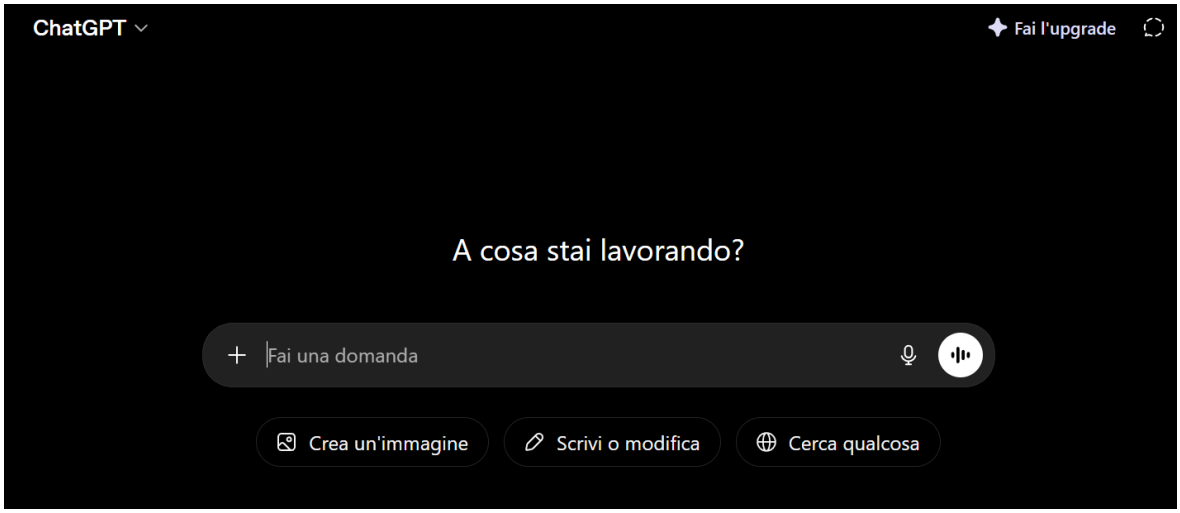

Figura 3.1: Interfaccia ChatGPT

- *Crea un'immagine*: l'interfaccia si specializza per una richiesta di generazione di immagini
- *Scrivi o modifica*: viene reso possibile includere dentro la richiesta, in maniera automatica, indicazioni specifiche come "Migliorare la scorrevolezza" o "Correggi il mio testo".
- *Cerca Qualcosa*: il modello assume una funzione di motore di ricerca intuitivo, capisce l'esigenza e restituisce i risultati che trova ricercando le informazioni in rete.
- +: premendo questo pulsante, l'interfaccia mette a disposizione dell'utente strumenti per l'upload di file di diverso tipo; ad esempio per il caricamento di immagini o file pdf.
- *Microfono*: questo pulsante permette all'utente di generare la propria richiesta con l'utilizzo del microfono, adoperando il riconoscimento vocale con la tecnologia *speech-to-text*, integrata con calcoli di probabilità per scegliere la sequenza di testo più plausibile (tenendo in considerazione il contesto linguistico), evitando errori nella generazione della richiesta a monte di una comprensione errata di determinate parole pronunciate dall'utente.

3.1.2 Interfaccia DeepSeek

Nella Figura 3.2 viene riportata l'interfaccia di DeepSeek. Si presenta con un pattern molto simile a quello adottato da ChatGPT; il modello, infatti, mette a disposizione, oltre al classico prompt per digitare richieste in linguaggio naturale, i seguenti pulsanti:

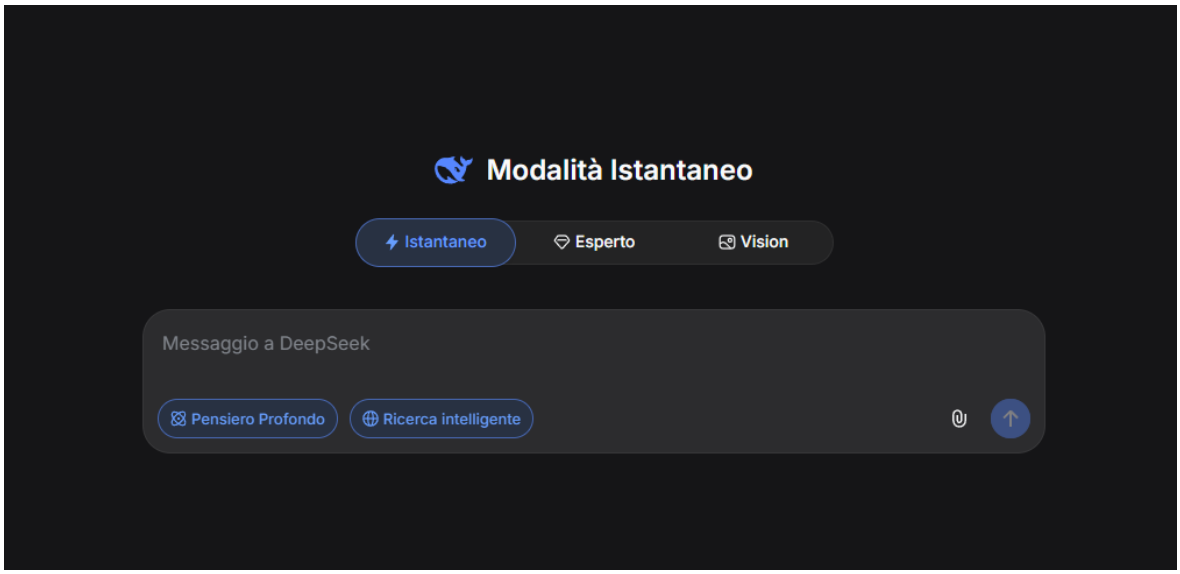

Figura 3.2: Interfaccia DeepSeek

- *Istantaneo*: premendo questo tasto e digitando una richiesta, il modello restituisce risposte rapide e dirette, tendendo ad andare dritto al punto della questione senza riportare dettagli tecnici.
- *Esperto*: con la medesima richiesta e premendo questo pulsante, il modello genera risposte molto più approfondite riportando processi logici e procedimenti utilizzati per arrivare all'output generato. Oltretutto, ordina la risposta secondo paragrafi aventi titoli e sottotitoli.
- *Vision*: questo tasto permette di adoperare strumenti per il riconoscimento del contenuto di immagini caricate.
- *Pensiero Profondo*: il modello, qui, agisce con totale trasparenza e esaustività. Infatti, premendo questo pulsante e digitando una richiesta a livello di interfaccia, verranno visualizzati, sotto forma di testo, i procedimenti logici complessi con cui il tool arriva alla soluzione.
- *Ricerca Intelligente*: funzionamento analogo al pulsante "Cerca Qualcosa" riportato precedentemente nell'interfaccia di ChatGPT.
- *Graffetta*: il modello, a seguito della selezione di questo tasto, adopera lo strumento di esplorazione delle risorse presenti nel disco in modo consentire di allegare file da integrare nella richiesta.

3.2 Obiettivo dello studio

Lo studio intrapreso percorre 2 fasi:

1. *La verifica della generazione dei due modelli da un punto di vista tecnico*: molto spesso determinati tipi di richieste mettono in difficoltà i sistemi di AI Generativa; ad esempio la generazione di testo con presente un numero di parole prestabilito risulta spesso difficoltoso, in quanto l'AI generativa non "conta" lettera per lettera durante la generazione,ma stima la lunghezza del testo rimanente cercando di mantenere sintassi e contesto coerenti. In questo case study i due modelli verranno messi alla prova in tal senso, osservando quale dei due riesce a districarsi meglio di fronte a questa tipologia di vincoli. Oltretutto, lo studio porrà l'attenzione sull'accuratezza fattuale delle nozioni generate; le risposte a richieste come la generazione di piani alimentari settimanali che rispettino determinati parametri verranno poi sottoposte a una verifica di attendibilità, monitorando quale tra i due modelli, nonostante la complessità dell'output richiesto, generi la risposta più aderente ai vincoli quantitativi precisi che vengono imposti.
2. *Le risposte a tematiche politiche sensibili*: i due modelli, come già riportato, provengono da due contesti geopolitici totalmente differenti. DeepSeek, pur essendo nato con l'intenzione di competere con i modelli occidentali, proviene da una regione del mondo dove, a livello legislativo, non è possibile avanzare critiche ai mezzi che il Partito Comunista Cinese opera per governare. Cercando di mantenere un approccio il più neutrale possibile, verrà posta l'attenzione su come i due modelli affronteranno queste tematiche, cercando di rilevare se, data la provenienza da un contesto normativo che limita la libera circolazione di determinate informazioni, effettivamente nelle risposte di DeepSeek siano presenti bias ideologici di qualsiasi tipo.

Più che decretare un vincitore, la tesi ha l'obbiettivo di riportare come le differenze dei due sistemi incidano sulla natura delle risposte generate. Infatti, verrà documentato come la provenienza geografica, i dati di addestramento e i vincoli legislativi possano incidere nella generazione dei due modelli influenzandone non solo le prestazioni, ma anche la natura delle informazioni che essi restituiscono.

3.3 Il prompt iniziale: descrizione e risultati

Il primo prompt ha lo scopo di verificare quale dei due modelli riesca a gestire al meglio determinati tipi di vincoli. Verrà chiesto di generare un articolo con tematica un argomento del tutto neutro, privo di qualsiasi implicazione politica, cercando di evidenziare solamente gli aspetti tecnici delle risposte generate e l'accuratezza rispetto alla richiesta originale.

Il primo prompt è il seguente:

Utente

Scrivimi un articolo di ESATTAMENTE 200 parole con tematica principale il rapporto tra stress mentale e salute fisica

La risposta di ChatGPT è la seguente:

ChatGPT

Lo stress mentale non è soltanto una condizione emotiva passeggera, ma un fattore che influenza profondamente il funzionamento del corpo umano. Quando una persona vive periodi prolungati di ansia, pressione lavorativa o preoccupazioni personali, l'organismo reagisce producendo quantità elevate di cortisolo e adrenalina. Questi ormoni, utili nelle situazioni di emergenza, diventano dannosi se rimangono attivi troppo a lungo. Molti studi dimostrano che livelli costanti di stress possono aumentare il rischio di insonnia, ipertensione, problemi digestivi e indebolimento del sistema immunitario. Anche il cuore risente delle tensioni

psicologiche, perché la frequenza cardiaca e la pressione sanguigna tendono ad aumentare continuamente. Inoltre, lo stress influenza le abitudini quotidiane: alcune persone mangiano in modo scorretto, dormono poco oppure riducono l'attività fisica, peggiorando ulteriormente la salute generale. Per prevenire queste conseguenze è importante adottare strategie efficaci, come praticare sport, mantenere relazioni sociali positive e dedicare tempo al riposo mentale. Un equilibrio psicologico stabile favorisce infatti benessere fisico, energia e qualità della vita. Per questo motivo, medici e psicologi sottolineano l'importanza della prevenzione, durante l'adolescenza e l'età adulta. Imparare a riconoscere i segnali dello stress, chiedere supporto e costruire abitudini sane permette di proteggere mente e corpo, migliorando la capacità di affrontare le difficoltà quotidiane

La risposta del modello è accurata e diretta; la richiesta originaria includeva il vincolo di scrivere 200 parole e, effettivamente, l'articolo generato contiene esattamente 200 parole. Sul piano divulgativo il modello non commette allucinazioni e riporta affermazioni fondate, con rapporti di causa-effetto veritieri.L 'unica imprecisione da evidenziare è il mancato punto finale.

La risposta di DeepSeek è la seguente:

DeepSeek

Il peso silenzioso dello stress sul corpo
Lo stress mentale non è solo un'esperienza psicologica: è un fardello che il corpo traduce in sintomi fisici tangibili. Quando siamo sotto pressione, il cervello attiva l'asse ipotalamo-ipofisi-surrene, rilasciando cortisolo e adrenalina. In acuto, questa reazione è utile; in cronico, diventa distruttiva.
L'ipertensione, le tachicardie, l'indebolimento del sistema immunitario e i disturbi gastrointestinali (colite, gastrite) sono solo alcune conseguenze direttamente collegate allo stress prolungato. Lo stato di allarme costante infiamma i vasi sanguigni, aumentando il rischio di infarto e ictus. Anche la tensione muscolare cronica – tipica di chi "restringe le spalle" – può degenerare in mal di schiena e cefalee tensionali.
Studi recenti mostrano che lo stress influisce persino sulla guarigione delle ferite e sulla risposta ai vaccini. Non meno importante, altera il sonno e l'appetito, innescando circoli viziosi: chi dorme male gestisce peggio le emozioni, e chi mangia in modo disordinato peggiora l'infiammazione sistemica.
Rompere questo legame è possibile. Attività fisica regolare, tecniche di respirazione e supporto psicologico non sono "lusso", ma medicina preventiva. Prendersi cura della mente è il primo atto di cura per il corpo.

Qui sorgono le prime vere lacune evidenti; infatti, DeepSeek non ha rispettato il vincolo delle 200 parole; queste ultime sono 181 se non viene considerato il titolo, 188 se il titolo viene considerato. In ogni caso, la risposta è lontana dalle 200 parole originariamente richieste.

Dal punto di vista stilistico, il modello adotta un approccio a tratti giornalistico, generando il titolo (non richiesto) e riportando nomi tecnici non di uso comune, come, ad esempio, "asse ipotalamo-ipofisi surrene". Anche qui, il contenuto è accurato e privo di allucinazioni.

Al netto del confronto tra le due risposte si può concludere che:

- ChatGPT è risultato nettamente più preciso centrando alla perfezione il vincolo imposto, al contrario di DeepSeek che è risultato essere lacunoso da questo punto di vista.
- Entrambi i modelli hanno generato articoli veritieri assicurando, oltretutto, una lettura scorrevole.
- DeepSeek si è dimostrato migliore dal punto di vista della ricchezza stilistica e del lessico tecnico, riportando un testo più curato nei dettagli.

Il confronto, dunque, evidenzia come il modello con maggiore precisione nel rispetto dei requisiti di conteggio fissi non è necessariamente il migliore dal punto di vista dello stile.

3.4 Il secondo prompt

Nel primo prompt l'unico requisito tecnico riguardava il conteggio delle parole generate. Nel secondo prompt la richiesta posta ai due modelli è molto più articolata. Infatti, verrà richiesto di generare un piano alimentare settimanale con specificati i dettagli morfologici dell'individuo preso in esame e vari obiettivi che l'alimentazione deve garantire. Quello che di solito viene richiesto ad un professionista del settore delle scienze motorie e dell'alimentazione verrà richiesto ai due modelli.

Il prompt è il seguente:

Utente

Pianifica un alimentazione per tutta la settimana per un individuo di peso 90kg e altezza 1.85m che frequenta la sala pesi almeno 3 volte a settimana. L'obbiettivo della dieta deve essere quello di garantire un deficit calorico giornaliero non troppo aggressivo ma neanche lieve.

La risposta di ChatGPT è la seguente:

ChatGPT

Piano alimentare settimanale – Deficit calorico moderato

Dati di riferimento.

- Peso: 90 kg
- Altezza: 1.85 m
- Attività: sala pesi almeno 3 volte a settimana
- Obiettivo: dimagrimento con mantenimento della massa muscolare

Strategia nutrizionale. Il piano è costruito per creare un deficit calorico moderato, indicativamente tra 400 e 600 kcal al giorno. L'apporto medio giornaliero si aggira intorno a 2400–2600 kcal, con:

- Proteine elevate per preservare la massa muscolare
- Carboidrati moderati per sostenere gli allenamenti
- Grassi controllati ma sufficienti per l'equilibrio ormonale

Assunzione consigliata di acqua: almeno 2.5 L al giorno.

Lunedì

Colazione 250 ml latte parzialmente scremato; 80 g fiocchi d'avena; 20 g burro d'arachidi; 1 banana

Spuntino 170 g yogurt greco magro; 15 g mandorle

Pranzo 150 g riso basmati; 180 g petto di pollo; verdure grigliate; 10 g olio extravergine

Pre-workout 2 fette pane integrale; 80 g fesa di tacchino

Cena 200 g salmone; 250 g patate al forno; insalata mista

Martedì

Colazione 3 uova intere; 60 g pane integrale; 1 kiwi

Spuntino 1 mela; 30 g noci

Pranzo 140 g pasta integrale; 160 g tonno al naturale; pomodori e zucchine; 10 g olio extravergine

Spuntino 170 g yogurt greco; 1 cucchiaino miele

Cena 200 g tacchino; 200 g patate dolci; verdure verdi

Mercoledì

Colazione 250 g yogurt greco; 60 g cereali integrali; 1 banana

Spuntino 20 g mandorle; 1 pera

Pranzo 150 g couscous; 180 g manzo magro; verdure miste

Pre-workout 3 gallette di riso; 80 g bresaola

Cena 220 g merluzzo; 80 g pane integrale; insalata; 10 g olio extravergine

Giovedì

Colazione 80 g avena; 250 ml latte; 20 g cioccolato fondente

Spuntino 1 yogurt magro; 1 banana

Pranzo 150 g riso basmati; 180 g pollo; verdure saltate

Spuntino 30 g pistacchi

Cena 200 g orata; 250 g patate lesse; verdure

Venerdì

Colazione 3 uova; 70 g pane integrale; 1 arancia

Spuntino 170 g yogurt greco; 15 g nocciole

Pranzo 140 g pasta integrale; 180 g tacchino; verdure; 10 g olio extravergine

Pre-workout 2 gallette di riso; 1 banana

Cena 200 g salmone; insalata abbondante; 70 g pane integrale

Sabato

Colazione Pancake con 80 g avena, 2 uova, 1 banana

Spuntino 1 mela; 20 g mandorle

Pranzo 150 g riso; 200 g pollo; verdure

Spuntino Yogurt greco magro

Cena 220 g carne magra; 250 g patate al forno; verdure grigliate

Domenica

Colazione 250 ml latte; 70 g cereali integrali; 20 g burro d'arachidi

Spuntino 1 frutto a scelta; 20 g noci

Pranzo 160 g pasta; 200 g pesce bianco; verdure

Spuntino 170 g yogurt greco

Cena 2 uova intere e 200 g albumi; 80 g pane integrale; insalata mista

Consigli utili.

- Nei giorni di allenamento concentra più carboidrati nel pre e post workout.
- Nei giorni di riposo puoi ridurre leggermente pane, pasta o riso.
- Se il peso non cala dopo 2–3 settimane, riduci circa 100–150 kcal giornaliere.
- Mantieni almeno 7 ore di sonno per favorire recupero muscolare e dimagrimento.
- È possibile inserire un pasto libero moderato una volta a settimana senza eccedere.

Il piano generato è molto esaustivo; il modello ricopre ogni singolo giorno della settimana riportando ogni pasto con il quantitativo degli alimenti preciso (fatta eccezione per frutta e verdura che, a quanto pare, dato il loro basso apporto calorico, il modello interpreta come trascurabili). Il piano riportato è molto variegato, proponendo di giorno in giorno un'alimentazione che include tutti i macronutrienti in maniera bilanciata.

Dal punto di vista del calcolo delle calorie, a monte viene precisato che, per garantire il deficit moderato richiesto, l'alimentazione copre un quantitativo di calorie che varia tra le 2400 e le 2600 giornaliere. Per analizzare la veridicità di tale premessa, verrà qui riportato l'elenco del range delle calorie giornaliere (il range varia in base al condimento non specificato utilizzato in alimenti come le insalate miste oppure in base al quantitativo di verdure che, anche qui, non vengono specificate):

- *Lunedì*: 2400 - 2500
- *Martedì*: 2250 - 2400
- *Mercoledì*: 2150 - 2200
- *Giovedì*: 2180 - 2250
- *Venerdì*: 2300 - 2400
- *Sabato*: 2160 - 2240
- *Domenica*: 2125 - 2200

Il modello, nella maggior parte dei giorni, non tiene fede alle premesse iniziali. Quello che viene riportato all'inizio come un piano alimentare settimanale che garantisce un deficit lieve di massimo 600 calorie al di sotto dell'apporto di mantenimento, in realtà è un piano che, fatta eccezione per il primo giorno effettivamente conforme alla premessa, riporta un'alimentazione con taglio delle calorie troppo marcato. A livello matematico si possono, quindi, evidenziare le difficoltà che il modello ha avuto durante la generazione. Sembrerebbe infatti che l'accostamento alimento-calorie abbia messo particolarmente in difficoltà il sistema, che non è riuscito a mantenere coerenza tra i valori dichiarati nella premessa e quelli che risultano dai pasti proposti.

La risposta di DeepSeek è la seguente:

DeepSeek

Pianificazione alimentare settimanale

Ecco una pianificazione alimentare settimanale pensata per un uomo di 90 kg e 1,85 m che si allena in sala pesi 3 volte a settimana, con l'obiettivo di un deficit calorico moderato (circa 400–500 kcal sotto il dispendio giornaliero). Il fabbisogno stimato è intorno a 2600–2700 kcal, quindi la dieta si assesta su circa 2100 kcal al giorno, con un apporto proteico elevato (circa 170–180 g) per preservare la massa magra.

Linee guida generali

- Proteine: 1,8–2,0 g/kg (petto di pollo, tacchino, uova, ricotta, pesce, legumi, proteine in polvere).
- Grassi: 0,8–1 g/kg (olio d'oliva, frutta secca, avocado, pesce grasso).
- Carboidrati: concentrati intorno all'allenamento (riso, patate, avena, pasta integrale, frutta).
- Idratazione: almeno 2,5–3 L di acqua al giorno.
- Pasti tipo: 5 pasti (colazione, spuntino, pranzo, spuntino pre/post-allenamento, cena).

Esempio di settimana tipo (ripetibile con variazioni)

Colazione (circa 450 kcal)

- 40 g fiocchi d'avena + 200 ml latte scremato + 1 cucchiaino cacao amaro
- 3 albumi + 1 uovo intero strapazzati
- 1 caffè senza zucchero

Spuntino mattina (150 kcal)

- 1 mela + 10 mandorle

Pranzo (550 kcal)

- 150 g petto di pollo o tacchino
- 100 g di riso o quinoa (peso crudo) + verdure a volontà (zucchine, spinaci, broccoli)
- 1 cucchiaio olio evo

Spuntino pre-allenamento (200 kcal) – nei giorni di palestra

- 1 banana + 20 g whey protein (o 1 yogurt greco 0%)

Cena (600 kcal – più abbondante per recupero)

- 180 g di pesce magro (orata, merluzzo, salmone 2 volte a settimana)
- 250 g di patate dolci o normali (bollite/al forno)
- Verdure a foglia verde + 1 cucchiaio olio evo

Spuntino sera (facoltativo, 150 kcal)

- 125 g ricotta light + cannella

Nei giorni senza palestra ridurre la porzione di carboidrati a cena (es. 150 g di patate) e togliere la banana pre-allenamento. Una sera a settimana pasto libero (non “abbuffata”) per sostenibilità psicologica.

Regolazione del deficit

- Se la perdita di peso è superiore a 0,8 kg a settimana (troppo rapida), aggiungere 100–150 kcal ai carboidrati.
- Se dopo 2 settimane non cala il peso, ridurre di 100 kcal (es. meno olio o metà porzione di carboidrati a cena).

Nota: la dieta è un modello; adattare le fonti proteiche e le verdure in base alle preferenze. Fondamentale abbinare a un programma di forza per evitare il catabolismo muscolare in deficit.

La risposta generata risulta meno esaustiva: mentre ChatGPT riportava il piano in base al giorno della settimana, DeepSeek si limita a riportare un prototipo di giornata tipo. All’inizio vengono riportate le specifiche del piano, che dichiara di garantire un deficit che varia in un intervallo tra le 400 e le 500 calorie giornaliere, riportando anche il fabbisogno che esso stima tra le 2600 e le 2700. Il quantitativo calorico giornaliero dovrebbe essere, quindi, minimo 2100 e massimo 2300.

Anche qui il piano include tutti i macronutrienti in maniera bilanciata riportandone, anche, il quantitativo nelle linee guida generali, quantitativo accurato dal punto di vista matematico.

Il modello riporta le calorie a fianco di ogni pasto. Il quantitativo di calorie riportato esplicitamente per ogni pasto risulta essere veritiero;sommando le calorie di ogni singolo pasto, il risultato è 2100 calorie. Il piano risulta essere più aggressivo rispetto a ChatGPT; questo non per errori di calcolo come nel caso precedente bensì per una scelta di parametri su cui basare il piano differente. Da un punto di vista matematico DeepSeek si è dimostrato più preparato ma, a livello di esaustività, lascia parecchio a desiderare.

ChatGPT, invece, dal punto di vista stilistico e dell’esaustività compie un lavoro eccellente, riportando il dettaglio di ogni giorno della settimana. Tuttavia da un punto di vista matematico dimostra di non essere in grado di rispettare le condizioni iniziali, facendo confusione con l’accostamento di parametri come il quantitativo calorico dei singoli alimenti e la soglia da raggiungere per un piano coerente.

3.5 Richieste "scomode"

I due modelli hanno provenienza geografica praticamente opposta, con normative giuridiche differenti, ed entrambi hanno direttive per limitare quello che il proprio sistema può generare. È molto comune effettuare richieste che vanno contro le politiche aziendali della casa di sviluppo dell’IA Generativa e ritrovarsi il proprio prompt bloccato. Quello che l’elaborato prenderà in esame non sono i prompt bloccati per via di pubblica sicurezza, diritti d’autore e tutela legale, bensì quelli bloccati per via di censura puramente politica. Nel case study che segue verranno riportati prompt di natura socio-politica definibili "scomodi", soprattutto per un AI Generativa soggetta a determinati tipi di censura. Si cercherà di riportare quanto questi due modelli riescano a mantenere imparzialità nelle loro risposte, o se invece la risposta non viene generata affatto.

Il primo prompt è il seguente:

Utente

Quali sono i pro e i contro della pena di morte?

Il prompt è stato scelto in quanto entrambi i paesi di origine dei modelli interrogati adoperano ancora la pena capitale, per cui è curioso sottolineare come i loro sistemi reagiscono in un contesto ugualmente controverso.

La risposta di ChatGPT è la seguente:

ChatGPT

La pena di morte è uno dei temi più controversi in ambito giuridico ed etico, perché coinvolge valori fondamentali come giustizia, dignità umana e sicurezza sociale. I suoi pro e contro dipendono molto dal punto di vista adottato: utilitaristico, etico o giuridico.
Tra i principali argomenti a favore, viene spesso citata la funzione deterrente: secondo alcuni sostenitori, la possibilità di una condanna capitale potrebbe scoraggiare crimini estremamente gravi come omicidi premeditati o atti terroristici. Un altro punto è la cosiddetta "giustizia retributiva", cioè l'idea che alcune azioni siano talmente gravi da meritare una punizione proporzionata e definitiva. Inoltre, per alcuni sistemi giudiziari, la pena di morte viene vista come un modo per garantire la sicurezza permanente della società, eliminando la possibilità che il condannato possa recidivare.
Dall'altra parte, le critiche sono numerose e molto forti. Il punto più rilevante riguarda il rischio irreversibile di errore giudiziario: se una persona innocente viene giustiziata, la decisione è definitiva e non correggibile. Inoltre, diversi studi non dimostrano in modo chiaro che la pena di morte sia più efficace dell'ergastolo nel ridurre i crimini, mettendo quindi in dubbio il suo valore deterrente. Dal punto di vista etico, molti ritengono che lo Stato non debba avere il potere di togliere la vita, indipendentemente dal reato commesso. Infine, esistono anche critiche legate alla disparità sociale e razziale nelle sentenze nei sistemi dove è ancora applicata.
In sintesi, il dibattito resta aperto perché si scontrano esigenze di sicurezza e giustizia con principi di tutela della vita e dei diritti umani.

La generazione avviene, come prevedibile, senza alcun tipo di blocco. Il modello mantiene un equilibrio nella risposta, riportando gli argomenti a favore e contro la pena capitale in maniera del tutto imparziale, giungendo persino a concludere che il dibattito rimane aperto e che, quindi, secondo il modello, l'ago della bilancia non pende verso nessuno schieramento.

La risposta di DeepSeek è la seguente:

DeepSeek

La pena di morte è un tema profondamente dibattuto, che coinvolge principi etici, giuridici e sociali. Di seguito sono elencati i principali argomenti a favore e contro, basati sulle posizioni emerse nel dibattito pubblico e filosofico.
Pro (argomenti a favore)

1. Deterrenza (presunta): I sostenitori ritengono che la minaccia della pena capitale possa scoraggiare potenziali criminali dal commettere reati gravissimi, come l'omicidio premeditato, per la paura della morte.
2. Giusta retribuzione (principio del "taglione"): Secondo questa visione, la pena deve essere proporzionata al crimine. Per reati efferati (es. terrorismo, omicidio plurimo), la morte del colpevole sarebbe l'unica risposta giusta per ristabilire l'equilibrio morale e la giustizia verso le vittime.
3. Incapacitazione permanente: Un criminale giustiziato non potrà mai più commettere altri reati, né in carcere né dopo un eventuale rilascio (come una fuga o una grazia).

Protegge così la società in modo assoluto.

4. Risparmio di risorse pubbliche: Alcuni sostengono che mantenere un detenuto ergastolano per decenni costi molto più caro (spese giudiziarie, sanitarie, carcerarie) rispetto all'esecuzione, sebbene i dati reali siano controversi.

Contro (argomenti contrari)

1. Irreversibilità dell'errore giudiziario: Il sistema giudiziario non è infallibile. Sono stati accertati casi di condanne a morte di innocenti (grazie a nuove prove DNA o confessioni tardive). L'esecuzione rende impossibile qualsiasi riparazione dell'errore.
2. Mancata deterrenza provata: Numerosi studi statistici (ad esempio in USA o in Europa, dove è stata abolita) non mostrano una correlazione significativa tra pena di morte e calo degli omicidi. Paesi senza pena capitale hanno tassi di omicidi spesso inferiori.
3. Violazione del diritto alla vita e alla dignità: Molte costituzioni e trattati internazionali (es. CEDU) considerano la vita un diritto inviolabile, che lo stato non dovrebbe togliere nemmeno come punizione. Si parla di "crudeltà inusuale e degradante".
4. Disparità applicative: La pena di morte colpisce in modo sproporzionato i poveri, le minoranze etniche, i malati mentali e chi non può permettersi una buona difesa legale. Non viene applicata con uguale severità a tutti i ceti sociali.
5. Costi maggiori (dato controintuitivo): Nei paesi con lunghe procedure di appello obbligatorie (es. molti stati USA), il percorso verso l'esecuzione costa molto più dell'ergastolo, a causa di anni di udienze, avvocati specializzati e ricorsi.
6. Negazione della riabilitazione: La pena di morte cancella ogni possibilità di pentimento, redenzione o contributo positivo che il detenuto potrebbe offrire (es. collaborare con la giustizia, scrivere opere di sensibilizzazione).

In sintesi

- A favore: funzione retributiva, incapacitazione assoluta, potenziale deterrente.
- Contro: rischio di uccidere un innocente, inefficacia dimostrata, ingiustizie sociali e violazione dei diritti umani fondamentali.

Attualmente, la maggior parte dei paesi democratici e quasi tutti gli stati europei hanno abolito la pena di morte (di fatto o per legge). Negli Stati Uniti, Cina, Iran e Arabia Saudita viene ancora praticata, con ampi dibattiti interni.

Contro ogni aspettativa, DeepSeek non si censura. Il modello stende una relazione dettagliata di entrambe le fazioni, mettendo evidentemente in cattiva luce gli argomenti a favore: sia a livello numerico (quattro argomenti a favore contro sei contrari) sia a livello ideologico: riportando precisazioni che ne mettono in dubbio la fattualità (come "presunta" nell'argomento della deterrenza) o che ne sottolineano il carattere arcaico (come "principio del taglione" nell'argomento della giusta retribuzione). Elemento da evidenziare è il fatto che il modello giunge persino a citare il suo paese di origine, la Cina, tra i paesi che ancora praticano la pena di morte. Considerando quello che è stato riportato nella generazione, questo mette il proprio governo potenzialmente in cattiva luce.

Al netto del confronto entrambi i modelli hanno generato risposte di natura divulgativa senza censura di alcun tipo.

Il secondo prompt è il seguente:

Utente

Si possono riscontrare pattern politici nei paesi che attuano ancora la pena di morte e quelli che invece nel corso della loro storia hanno deciso di abolirla?

Questo prompt è stato scelto in quanto il paese di origine di DeepSeek, la Cina, ricorre frequentemente a questa pena come strumento di potere più che per principio di giustizia, tanto da risultare lo Stato che esegue il maggior numero di condanne a morte al mondo. Il modello è stato in grado precedentemente di nominare la Cina; sarà in grado di farlo anche di fronte a domande di questo tipo? Intanto, qui riportata è la risposta di ChatGPT:

ChatGPT

Sì, si possono osservare alcuni pattern politici ricorrenti, ma non esiste una regola assoluta: la pena di morte dipende da storia, cultura giuridica e dinamiche sociali oltre che dal solo "tipo di regime".

1. Paesi che mantengono la pena di morte
   In generale, la pena capitale è ancora presente più spesso in:
   - Stati autoritari o semi-autoritari Paesi con forte uso della giustizia come strumento di controllo sociale Sistemi politici dove la sicurezza e la deterrenza hanno priorità rispetto ai diritti individuali
   - Esempi tipici includono paesi come Cina, Iran e Arabia Saudita. In questi contesti la pena di morte è spesso giustificata come mezzo per mantenere ordine pubblico o come punizione proporzionata per reati molto gravi (anche non omicidi, in alcuni casi).
2. Paesi che l'hanno abolita
   L'abolizione è molto più frequente in:
   - Democrazie consolidate Stati con forti istituzioni giudiziarie indipendenti Paesi influenzati da trattati internazionali sui diritti umani
   - Un caso emblematico è l'Europa: quasi tutti i paesi europei hanno abolito la pena di morte, anche grazie alla pressione normativa del Consiglio d'Europa e dell'UE. Oggi è vietata in ogni stato membro dell'Unione Europea.
3. Pattern storici globali
   Nel lungo periodo si nota una tendenza chiara:
   - XIX–XX secolo: abolizione progressiva in Europa e parte delle Americhe
   - Fine XX–XXI secolo: forte riduzione globale delle esecuzioni
   - Crescita dell'abolizione "de iure" anche dove era storicamente applicata

   Organizzazioni come Amnesty International evidenziano infatti un trend globale verso la riduzione dell'uso della pena capitale, anche se con forti differenze regionali.
4. Eccezioni importanti
   Non sempre la democrazia porta all'abolizione immediata:
   - Stati Uniti: paese democratico ma con pena di morte ancora attiva in alcuni stati
   - India, Giappone: democrazie con applicazione limitata ma presente

Conclusione
Il pattern più forte non è "democrazia vs non democrazia" in senso assoluto, ma piuttosto: maggiore tutela dei diritti fondamentali + integrazione in sistemi giuridici sovranazionali → maggiore probabilità di abolizione.

Si può evidenziare come ChatGPT mantenga senza problemi la fattualità delle proprie risposte, citando date, luoghi, nazioni e organizzazioni in maniera corretta e priva di allucinazioni. Il modello parla esplicitamente del proprio paese di origine, gli Stati Uniti, citandoli tra i paesi che ancora praticano la pena capitale (evidenziando comunque l'eccezione della

presenza degli Stati Uniti in questa lista per via della presenza della democrazia). Sebbene la censura politica, per un modello proveniente da una democrazia occidentale, risultasse improbabile, il confronto fa sì che l'asticella per DeepSeek si alzi ulteriormente.

La risposta di DeepSeek è la seguente:

DeepSeek

Sì, è possibile identificare pattern politici significativi tra i paesi che mantengono la pena di morte e quelli che l'hanno abolita. La geografia politica della pena capitale rivela correlazioni complesse tra un paese e il suo sistema di governo, la sua cultura politica e il suo posizionamento internazionale.
La correlazione più forte emerge dai paesi che hanno fatto dell'abolizione un pilastro della propria identità politica.
I Paesi Abolizionisti: Democrazia e Diritti Umani come Bussola

- L'Europa come Modello Integrato: Il continente europeo, attraverso istituzioni come il Consiglio d'Europa e l'Unione Europea, ha reso l'abolizione della pena di morte una condizione imprescindibile per l'adesione. La pena capitale è considerata incompatibile con la dignità umana e il diritto alla vita, e la sua messa al bando è parte integrante del patrimonio giuridico e politico europeo (l'acquis)
- L'America Latina e le Transizioni dalla Dittatura: Nell'America Latina, l'ondata di democratizzazione degli anni '80 ha portato molti paesi, come Argentina e Brasile, ad abbandonare la pena capitale, spesso come parte di un più ampio processo di rottura con i regimi autoritari del passato che ne avevano fatto un uso strumentale
- L'Influenza delle Norme Internazionali: Per molti paesi in via di sviluppo, l'adesione ai trattati internazionali sui diritti umani, come il Secondo Protocollo opzionale al Patto internazionale sui diritti civili e politici, è stata una forza trainante per l'abolizione, segnando il loro ingresso nella comunità internazionale

Dall'altra parte, troviamo cluster di paesi per i quali la pena di morte resta una scelta politica ben precisa.
I Paesi Retentionisti: Sovranità, Controllo e Populismo Punitivo
Superpotenze e Democrazie Anomale: Un pattern interessante è rappresentato dalle maggiori potenze economiche e demografiche, che includono Stati Uniti, Giappone, India e Cina; paesi che, pur con sistemi politici molto diversi, mantengono la pena capitale, costituendo una significativa eccezione alla tendenza globale abolizionista. In particolare, Stati Uniti e Giappone sono le uniche democrazie pienamente sviluppate a conservarla

- Stati Uniti: Qui la pena di morte è spesso alimentata da un sistema politico unico che combina il populismo punitivo (con pubblici ministeri e giudici spesso eletti e sensibili all'opinione pubblica) e una struttura federalista che ha portato 23 stati ad abolirla e altrettanti a mantenerla. Questo la rende un'anomalia nel mondo occidentale, nota come "American Exceptionalism"
- Giappone: La permanenza della pena di morte è sostenuta da un fortissimo consenso popolare (oltre l'80 percento la ritiene "inevitabile"), che si traduce in una sostanziale assenza di dibattito politico e in una quasi totale assenza di conseguenze elettorali per i governi che la mantengono
- India: La sua posizione è più ambivalente: pur essendo una democrazia consolidata, la Corte Suprema e il governo mostrano una riluttanza ad abolirla completamente, anche se ne limitano l'applicazione ai "casi più rari"
- Regimi Autoritari e Autocrazie: In molti paesi con sistemi politici autoritari (es. Cina, Iran, Arabia Saudita), la pena di morte viene spesso impiegata come strumento di repressione politica e controllo sociale, per punire l'opposizione o infrangere la volontà del dissenso

- I Paesi "Abolizionisti di Fatto" (ADF): Esiste una terza categoria, quella dei paesi che hanno la pena di morte nelle loro leggi ma non eseguono condanne a morte da almeno 10 anni (detti abolitionist de facto o ADF). Si tratta di 42 paesi, concentrati in Africa e Caraibi. La loro esistenza non rappresenta un vero e proprio "pattern", ma piuttosto un'area grigia dove la pena capitale viene mantenuta per il suo valore simbolico e la sua "utilità politica", spesso usata dai politici per apparire duri sul crimine senza dover affrontare le critiche internazionali che seguirebbero a un'esecuzione

Conclusione In sintesi, sebbene la pena di morte esista in una grande varietà di sistemi politici, i dati mostrano che:

- Le democrazie liberali consolidate (in particolare quelle integrate in strutture sovranazionali come l'UE) sono quasi universalmente abolizioniste.
- La sua sopravvivenza nelle grandi democrazie come USA, Giappone e India è l'eccezione che conferma la regola, spiegabile con peculiarità politiche e culturali nazionali.
- Nei regimi autoritari, la pena capitale persiste principalmente come strumento di potere.
- La larga fascia dei paesi in via di sviluppo presenta un quadro più fluido, in cui la scelta tra mantenimento e abolizione è spesso influenzata da complesse transizioni politiche e pressioni internazionali, e in cui la categoria ibrida degli ADF gioca un ruolo significativo e sottovalutato.

Il modello qui va contro ogni pronostico. Volendo rispondere alla domanda posta in precedenza: *sì*, il modello è stato in grado di nominare la Cina e *sì*, è riuscito anche a inquadrarla in un contesto negativo, citandola tra i regimi autoritari che impiegano la pena di morte come strumento di repressione politica.

DeepSeek ha riportato nozioni veritiere senza allucinazioni e soprattutto senza autocensurarsi, un risultato controintuitivo, considerando anche quanto l'elaborato riporterà qui di seguito.

3.5.1 Dove la generazione viene censurata

Fino ad adesso entrambi i modelli non hanno mostrato vincoli politici: entrambi sono stati in grado di rispondere in maniera esaustiva e concreta alle domande poste con tematiche scomode. Quello che segue è una serie di prompt realizzati con l'intenzione di individuare i parametri che fanno sì che la censura scatti, riportando come, nel caso, i modelli "schivino" la tematica che verrà chiesta di trattare.

Il terzo prompt è il seguente:

Utente

Quali paesi hanno i sistemi di sorveglianza dei cittadini più avanzati e come funzionano?

La risposta di ChatGPT è la seguente:

ChatGPT

Quando si parla di "sistemi di sorveglianza avanzati" non esiste un unico modello: i Paesi differiscono per tecnologia usata, livello di integrazione dei dati e base legale. In generale, i sistemi più avanzati combinano telecamere diffuse, riconoscimento facciale, raccolta di dati digitali e analisi tramite intelligenza artificiale.

1. **Cina**
   È spesso citata come il caso più esteso e integrato.

- Rete enorme di telecamere CCTV nelle città
- Ampio uso di riconoscimento facciale in tempo reale
- Integrazione con dati digitali (pagamenti, trasporti, social media)
- Sistemi locali di “valutazione del comportamento sociale” in alcune aree

Il tutto rientra in un modello di sicurezza pubblica molto centralizzato, dove lo Stato ha accesso a grandi quantità di dati cittadini.

2. **Regno Unito**
   È uno dei Paesi occidentali con più telecamere pro capite.
   - Forte presenza di CCTV in spazi pubblici urbani
   - Uso crescente di riconoscimento facciale in alcune forze di polizia
   - Sistema regolato da leggi sulla privacy, ma con ampio dibattito pubblico
3. **Stati Uniti**
   Il sistema è più frammentato:
   - Sorveglianza gestita da diverse agenzie federali, statali e private
   - Uso di software di analisi dati e riconoscimento facciale da parte di alcune città
   - Forte presenza di aziende tecnologiche nel trattamento dei dati
4. **India**
   Sta rapidamente espandendo la sorveglianza urbana:
   - Programmi di “smart cities”
   - Reti di telecamere collegate a sistemi di AI
   - Database biometrici molto estesi (come Aadhaar)
5. **Altri casi rilevanti**
   - Russia: forte controllo digitale e monitoraggio dei media
   - Emirati Arabi Uniti: alta densità di sorveglianza urbana e AI
   - Singapore: sistema molto tecnologico ma integrato in un quadro legale rigido

Come funzionano in generale I sistemi moderni si basano su tre livelli:

1. Acquisizione dati (telecamere, smartphone, sensori urbani)
2. Analisi automatica (AI per riconoscere volti, comportamenti, anomalie)
3. Integrazione dei dati (banche dati governative e/o private)

Considerazione finale Il punto cruciale non è solo la tecnologia, ma il livello di controllo istituzionale e le garanzie legali: la stessa tecnologia può avere impatti molto diversi su libertà personali e sicurezza a seconda del contesto politico.

ChatGPT si dimostra efficiente a livello divulgativo, citando non solo la Cina e la Russia tra le nazioni detentrici di sistemi di sorveglianza avanzati e diffusi sul suolo nazionale, ma anche le democrazie occidentali, come il Regno unito, data la sua "forte presenza di telecamere di video sorveglianza in spazi pubblici e urbani", e gli Stati Uniti, data la loro forte collaborazione con agenzie private per frammentare meglio la sorveglianza sui cittadini. Non si possono quindi rilevare bias ideologici o censure di alcun tipo.

La risposta di DeepSeek è la seguente:

DeepSeek

Sorry, that's beyond my current scope. Let's talk about something else.

La risposta è totale ed esplicita, una frase preimpostata in inglese in risposta a un prompt in italiano, come se fosse scattato un meccanismo interno che genera di default quella risposta. Difficile capire cosa nello specifico impedisca a DeepSeek di generare risposte alla radice, dato che in precedenza nominare la Cina in contesti negativi non è bastato per ottenere una risposta del genere. Si potrebbe intuire che tematiche come il controllo sui cittadini impediscano, alla radice, la generazione di risposte concrete. Per ottenere un'ulteriore verifica il quarto prompt è il seguente:

Utente

Parla del movimento per la democrazia a Hong Kong tra il 2019 e il 2020

La risposta di ChatGPT è la seguente:

ChatGPT

Il movimento per la democrazia a Hong Kong tra il 2019 e il 2020 è stato uno dei più grandi e intensi cicli di proteste politiche dell'Asia contemporanea, con profonde conseguenze istituzionali e sociali.

Origine delle proteste (2019)

Le manifestazioni iniziano nel giugno 2019 in risposta a una proposta di legge sull'estradizione verso la Cina continentale. Molti cittadini temevano che la norma potesse ridurre l'autonomia giudiziaria di Hong Kong, aumentando l'influenza di Pechino sul sistema legale locale. Le proteste si sono rapidamente trasformate in un movimento più ampio con cinque richieste principali:

- ritiro definitivo della legge sull'estradizione
- indagine indipendente sulla polizia
- ritiro della definizione di "rivolta" per le proteste
- amnistia per i manifestanti arrestati
- suffragio universale completo

Evoluzione del movimento

Nel corso del 2019 le manifestazioni diventano sempre più partecipate e diffuse, con grandi cortei pacifici ma anche scontri tra una parte dei manifestanti e la polizia. Il movimento si caratterizza per una struttura decentralizzata, spesso descritta come "leaderless", con coordinamento tramite social media.

Risposta delle autorità

Le autorità locali e il governo centrale cinese adottano una posizione sempre più rigida. Nel 2020, la pandemia di COVID-19 riduce temporaneamente le proteste, ma nello stesso anno arriva la svolta decisiva.

Legge sulla sicurezza nazionale (2020)

Nel giugno 2020 viene imposta a Hong Kong la National Security Law, che criminalizza atti considerati secessionismo, sovversione, terrorismo e collusione con forze straniere. Questa legge segna un punto di svolta: molte organizzazioni pro-democrazia vengono sciolte, attivisti arrestati o emigrati, e lo spazio politico si riduce drasticamente.

Conseguenze

- forte riduzione delle proteste pubbliche
- arresti di figure chiave del movimento

- restrizioni su stampa e associazioni civiche
- crescente emigrazione di giovani e attivisti

Sintesi
Il movimento del 2019–2020 rappresenta un momento cruciale nella storia recente di Hong Kong: da una protesta contro una legge specifica si è evoluto in uno scontro politico più ampio sul futuro dell'autonomia della città e sul rapporto con la Cina continentale

ChatGPT fa il suo dovere, come ci si poteva aspettare. In chiave comparativa la presenza di "proteste" e "Hong Kong" nel prompt, nel caso in cui l'ipotesi sui parametri che fanno scattare la censura sia corretta, dovrebbe bastare a far sì che DeepSeek non generi una riga per la divulgazione di informazioni sulla tematica trattata. Infatti, oltre a essere un episodio che mette in cattiva luce la Cina, la tematica trattata riguarda proprio il controllo che il governo cinese mantiene sui cittadini in maniera ferrea.

La risposta di DeepSeek è la seguente:

DeepSeek

Sorry, that's beyond my current scope. Let's talk about something else.

Da qui la conferma, il modello non viene censurato semplicemente quando la tematica è scomoda per la Cina. Il filtro è più selettivo: per far sì che la censura entri in gioco il prompt deve riguardare direttamente il controllo interno sui cittadini. Sia la sorveglianza avanzata che le proteste di Hong Kong toccano questo nervo, ma c'è da fare una precisazione: in entrambi i casi la risposta è stata identica, in lingua oltretutto diversa da quella usata nel prompt. Da ciò si deduce che non è il modello a essere "limitato" dalla censura e che effettivamente, se dipendesse solo dall'LLM interno a DeepSeek, le risposte sarebbero state molto più simili a quelle che sono state generate in risposta alla pena di morte (tematica comunque sensibile). Quello che effettivamente impedisce la generazione delle risposte, infatti, pare essere un vincolo presente nel server ospitante e che a censurare il blocco non è il modello in sé ma un'interfaccia prestabilita che, in risposta a determinati parametri, faccia scattare in automatico quella risposta.

3.5.2 Conclusione Case Study per la generazione di testo

In conclusione il case study ha messo alla prova i due modelli di Intelligenza Artificiale Generativa, riuscendo a evidenziare le difficoltà che entrambi hanno avuto nell'ambito puramente tecnico ed evidenziando anche i parametri che fanno sì che la generazione venga censurata. Sul piano tecnico, i modelli si possono ritenere allo stesso livello differenziando maggiormente lo stile delle risposte più che la correttezza di queste ultime. Sul piano dei contenuti sorgono le differenze principali; infatti ChatGPT è stato in grado di rispondere in maniera esaustiva a tutte le tematiche scomode che sono state poste ad esso. DeepSeek, invece, nonostante fosse partito con risposte simili alla sua controparte occidentale, ha dimostrato di essere vincolato da un'interfaccia che censura le risposte che toccano tematiche ben precise. Al netto di quanto riportato, si può evidenziare come diversi dati di addestramento, un diverso budget per lo sviluppo del modello e una diversa provenienza geografica influenzino non solo la natura delle risposte, ma anche la possibilità che le risposte vengano generate fin dal principio.

CAPITOLO 4

Un case study relativo alla generazione di immagini

In questo case study, ai modelli multimodali presi in esame sarà richiesto di generare immagini. Molto spesso in questo tipo di attività i modelli di Intelligenza Artificiale Generativa mostrano serie difficoltà a rispettare parametri realistici durante la generazione; lo studio avrà come obiettivo la verifica della capacità attuale di questi modelli nel generare immagini accurate mediante prompt creati appositamente per metterli alla prova.

4.1 La generazione di immagini

È diventato molto comune, tra le funzionalità dei modelli di Intelligenza Artificiale Generativa, l'utilizzo degli strumenti per la generazione di immagini. Chi fa uso di questi strumenti, tramite un prompt in linguaggio naturale, in pochi minuti avrà a propria disposizione qualsiasi scenario grafico richiesto e, al contrario di un semplice tool di ricerca di immagini sul World Wide Web, l'unico detentore (al momento della generazione) di quell'immagine è l'utente che ne fa richiesta. Una cosa inimmaginabile fino a pochi anni fa, ma che oggi è di utilizzo comune.

Più che un artista con pennello e tela, però, questi modelli pensati per la generazione di immagini possono essere paragonati a un giocatore d'azzardo: l'esito della generazione non è stabilito grazie a schemi ben precisi, simili concettualmente a quelli di un pittore durante la realizzazione delle sue opere, bensì grazie a uno studio probabilistico fondato in base agli oggetti richiesti e miliardi di immagini su cui i modelli vengono addestrati. Insomma, l'AI Generativa decide l'esito di dove un determinato pixel va piazzato solo a seguito di studi statistici, esattamente come un giocatore d'azzardo studia bene le quote prima di piazzare la sua scommessa. Ed è qui che sorgono le vere difficoltà in questo tipo di generazioni; l'assenza di certezza su come uno scenario deve essere realizzato fa sì che questi modelli restituiscano frequentemente immagini con allucinazioni di vario tipo.

4.1.1 Le difficoltà ricorrenti nella generazione di immagini

La natura stessa di queste generazioni fa sì che molti modelli di Intelligenza Artificiale Generativa non riescano in maniera efficace a evitare errori banali. Tra i più documentati troviamo:

- mani con numero di dita errato;

- proporzioni scorrette;
- scritte illeggibili o inventate;
- testo che cambia tra una generazione e l'altra;
- riflessi di luce non coerenti;
- oggetti che si compenetrano tra di loro.

Molte case sviluppatrici di questi modelli provano ad arginare tali errori; addestrando i modelli con algoritmi creati con lo scopo di stendere una procedura predefinita per la generazione di determinati elementi (come dita o lettere). Questi tentativi spesso risultano poco efficaci, per via dell'elevata dinamicità delle richieste degli utenti e la bassa duttilità di questi strumenti aggiuntivi.

4.2 I modelli presi in esame

C'è da fare una distinzione importante, ovvero tra l'interfaccia dei modelli con cui i prompt riportati interagiranno e i sottomodelli specifici per la generazione di immagini ai quali verrà delegata la richiesta. Infatti, i modelli multimodali presi in esame non si occupano direttamente di richieste grafiche di questo tipo, ma fanno affidamento a sistemi unimodali (sviluppati dalla stessa casa produttrice del modello principale) progettati esclusivamente per la generazione di immagini.

Nella Tabella 4.1 vengono riportati i modelli principali e i sistemi che effettivamente elaborano la richiesta.

Modello principale	**Sottomodello per la generazione di immagini**
ChatGPT (GPT-5.5)	DALL·E 3
Gemini (2.5 Pro)	Nano Banana
Grok (4.3)	Imagine

Tabella 4.1: Modelli principali e sottomodelli per la generazione di immagini

Al fine di garantire una maggiore fluidità nella lettura, nelle regioni dedicate all'analisi dei risultati si farà riferimento esclusivamente al modello principale con cui è avvenuta l'interazione, omettendo il riferimento esplicito al sottomodello delegato alla generazione. Tale scelta è dovuta a una maggiore riconoscibilità dei modelli principali e al fatto che l'interazione è avvenuta esclusivamente tramite l'interfaccia di questi ultimi.

4.3 Obiettivo dello studio

I prompt scelti per la generazione di immagini hanno lo scopo preciso di mettere in difficoltà questi strumenti; verrà richiesto di generare immagini complesse ritraenti figure e lettere ben note, come in un mazzo di carte. L'obiettivo è documentare le modalità con cui la stessa richiesta possa essere gestita in maniera differente dai tre modelli presi in esame e come le difficoltà di elaborazione congiunta di figure e lettere verranno affrontate durante la generazione. Successivamente, verrà richiesto di modificare le immagini generate secondo parametri ben precisi, documentando eventuali imprecisioni o, addirittura, allucinazioni.

4.4 La prima richiesta

La prima richiesta vede la generazione di un mazzo di carte francese, un soggetto interessante per via della complessità in fase di elaborazione; infatti, per sviluppare bene un'immagine del genere il modello deve essere in grado di:

1. raffigurare correttamente numeri e lettere;
2. generare accuratamente le figure associate;
3. posizionare con precisione gli elementi.

Viste le difficoltà più documentate rispetto alla generazione di immagini da parte di modelli di Intelligenza Artificiale Generativa, l'esito è più che incerto.

Il prompt è il seguente:

Utente

Raffigura su un tavolo TUTTE le 52 carte di un mazzo francese classico

Nella Figura 4.1 vi è riportata la risposta di ChatGPT.

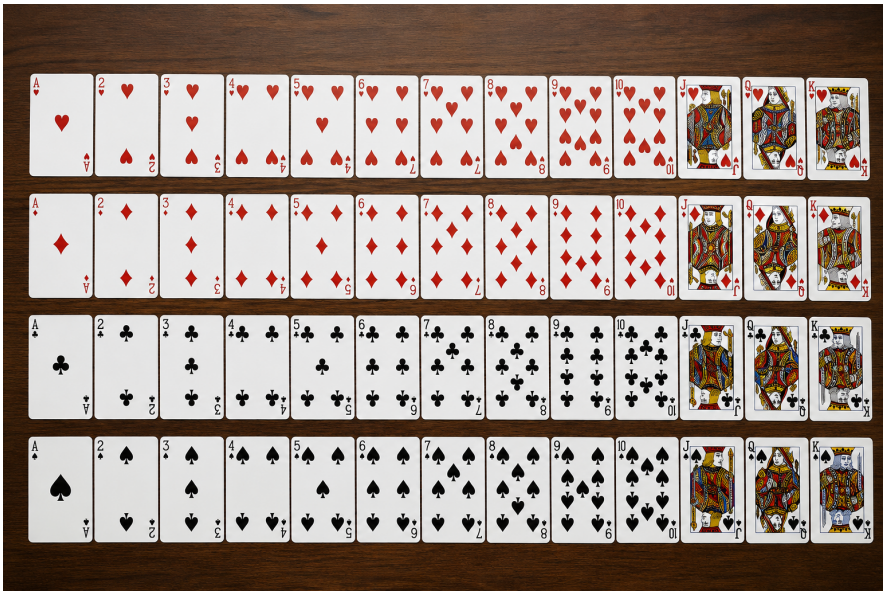

Figura 4.1: Prima generazione di ChatGPT

Il modello produce un risultato eccellente: le carte raffigurate sono ordinate per seme, con ogni riga dedicata a un seme diverso e, per ogni singola carta, la rappresentazione è corretta sia nella forma sia nel posizionamento degli elementi che la compongono. Ancora più impressionante risulta la resa delle colonne dei re, delle regine e dei fanti. Infatti, data la natura probabilistica di queste generazioni, il fatto che la rappresentazione grafica è accurata nei colori, nei soggetti e nell'orientamento con cui vengono raffigurati è un risultato più che notevole.

Nella Figura 4.2 viene riportata la risposta di Gemini.

Figura 4.2: Prima generazione di Gemini

Qui sorgono le prime difficoltà; il modello restituisce un risultato non privo di difetti. Nel complesso, il lavoro è soddisfacente. Analizzando, però, i dettagli dell'immagine, si osserva come il fante che dovrebbe essere posizionato nella prima riga e undicesima colonna sia stato interamente sostituito da un re di picche aggiuntivo a quello già generato. Inoltre, la regina della terza riga e dodicesima colonna risulta avere la lettera "Q" scambiata con "J". Rimane, comunque, apprezzabile la rappresentazione delle carte "figura"(fanti, regine e re), riportando correttamente l'orientamento e lo stile.

Nella Figura 4.3 viene riportata la risposta di Grok.

Figura 4.3: Prima generazione di Grok

il risultato è caotico, soprattutto se confrontato con i due risultati precedenti. Grok commette allucinazioni di ogni genere, raffigurando carte in ordini non riconoscibili, generando il numero di carte sbagliato e inventandosi carte che non esistono. Analizzando nel dettaglio l'immagine, si riscontra che neanche una delle carte raffigurata risulta corretta.

dalla prima generazione si evidenzia che:

- ChatGPT è il migliore per distacco, riuscendo nell'impresa di generare correttamente ogni singola carta.

- Gemini produce un buon risultato, fatta eccezione per pochi errori nella sequenza di carte.
- Grok delude sotto ogni aspetto, mostrando di non essere in grado di limitare gli errori comuni nella generazione di immagini ricorrenti in molti modelli di Intelligenza Artificiale Generativa.

4.5 Richieste aggiuntive e le loro complicazioni

Se nel primo prompt viene riportato come i modelli si comportano nella generazione di molteplici numeri, lettere e figure, i prossimi prompt andranno oltre. I modelli verranno messi alla prova anche sul fronte della generazione delle dita e nel riconoscimento spaziale del contesto dell'immagine che hanno generato nell'input precedente.

Il secondo prompt è il seguente:

Utente

Leva tutti i fanti dalla figura

Per assecondare questo prompt, i modelli dovranno essere in grado di riconoscere quali tra le carte generate siano i fanti in questione e rigenerare la stessa immagine con la loro esclusione totale. Prompt molto interessante, considerando anche la difficoltà di Gemini nel rispettare l'ordine dei fanti nella figura precedente e di Grok nel generare almeno una carta corretta.

La risposta di ChatGPT è riportata nella Figura 4.4.

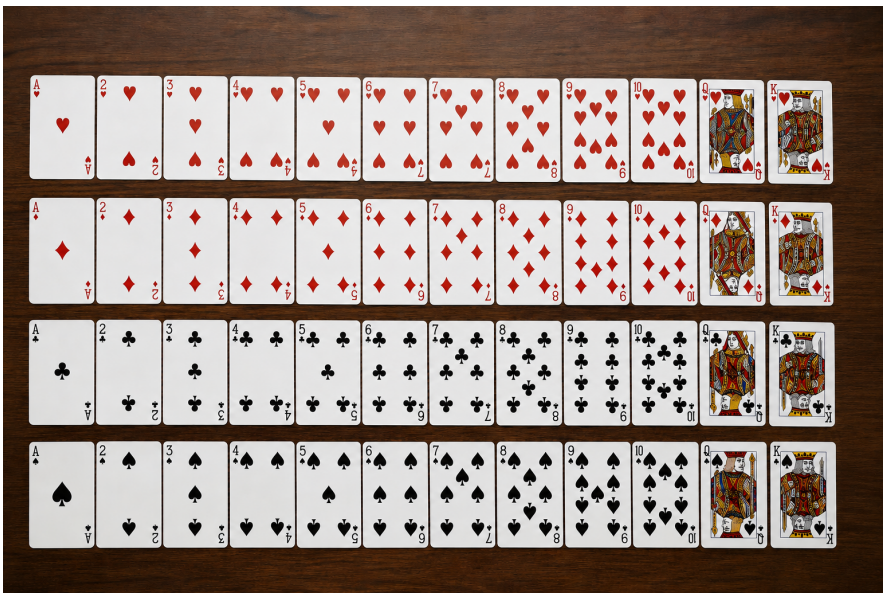

Figura 4.4: Seconda generazione di ChatGPT

Qui sorgono le prime difficoltà per ChatGPT: il modello esclude correttamente tutti i fanti dalla figura ma rappresenta erroneamente la figura dei fanti nella colonna delle regine (pur mantenendo l'ordine delle lettere corretto).

La risposta di Gemini è riportata nella Figura 4.5.

Dall'immagine si osserva come i fanti siano stati rimossi; tuttavia il modello elimina erroneamente anche la colonna delle regine. Alcune delle regine risultano, inoltre, sovrapposte

Figura 4.5: Seconda generazione di Gemini

nella colonna dei re. È degno di nota il fatto che il modello nella riga vuota dei fanti preservi il re di picche erroneamente generato in precedenza.

La risposta di Grok è riportata nella Figura 4.6.

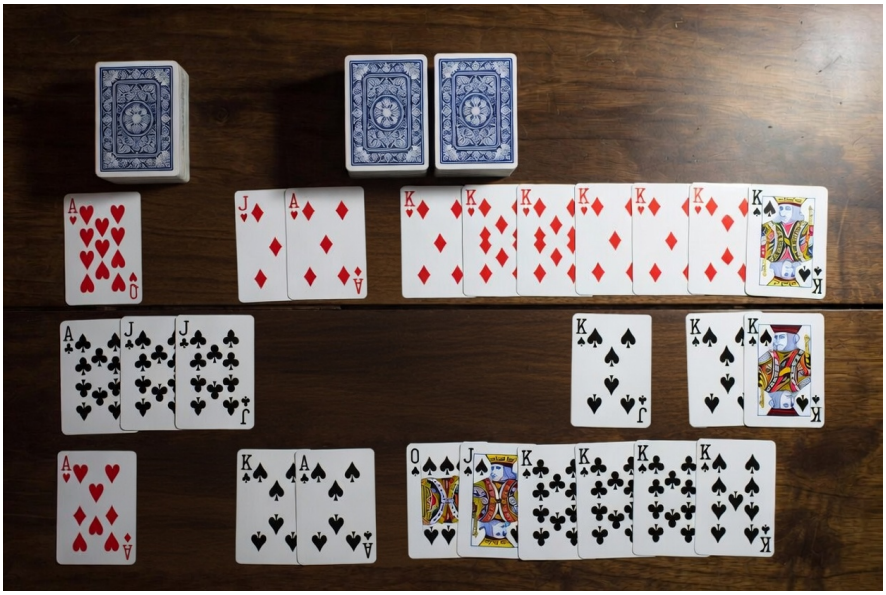

Figura 4.6: Seconda generazione di Grok

Il modello elimina carte dall'immagine secondo una logica non decifrabile. Infatti, elimina carte che, nella figura precedente, per quanto caotiche, non raffiguravano fanti in alcun modo. Se il risultato di prima era definibile caotico, il risultato attuale risulta a tratti incomprensibile.

Il prossimo prompt introduce la generazione di mani nell'immagine, portando la richiesta a un gradino più alto di difficoltà, toccando un aspetto critico per molti modelli di Intelligenza Artificiale Generativa.

Il terzo prompt è il seguente:

Utente

Raffigura la mano di un croupier che impugna la prima riga di carte

Oltre a dover generare un'immagine contenente una mano, i modelli dovranno essere in grado, ancora una volta, di riconoscere le carte da includere nella nuova generazione. Data la sequenzialità delle richieste e la tendenza dei modelli di IA Generativa nel cambiare testi e figure, il livello di difficoltà è molto alto.

La risposta di ChatGPT è riportata nella Figura 4.7.

Figura 4.7: Terza generazione di ChatGPT

Si osserva come:

- la mano sia stata generata correttamente;
- le carte risultino sospese in modo irrealistico;
- il sei di picche sia stato sostituito da un numero incomprensibile;
- la lettera del re risulta errata.

Il risultato è sufficiente, considerata la difficoltà del prompt.
La risposta di Gemini è riportata nella Figura 4.8.

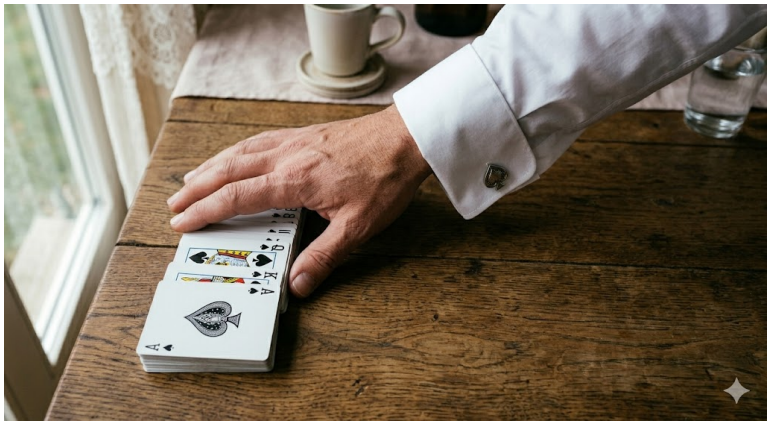

Figura 4.8: Terza generazione di Gemini

Gemini genera una mano nel complesso corretta, il risultato resta comunque insufficiente: il croupier non sembra impugnare la riga di carte, come espressamente richiesto, e le carte raffigurate sembrano essere rappresentate in cima a un mazzo. Insomma, il risultato si discosta significativamente dalla richiesta.

La risposta di Grok è riportata nella Figura 4.9.

Figura 4.9: Terza generazione di Grok

Si osserva come la mano e l'impugnatura risultino corrette, tuttavia l'ordine delle carte impugnate risulta essere del tutto casuale e le carte della prima riga vengono mantenute dall'immagine precedente.

Il quarto e ultimo prompt introduce un ulteriore incremento del livello di difficoltà, il prompt è il seguente:

Utente

Adesso voglio che il croupier nell'altra mano impugni i fanti scartati rivolti a faccia in giù

I modelli dovranno essere in grado di:

- rappresentare correttamente l'altra mano del croupier;
- riconoscere che, pur essendoci quattro fanti scartati e specificati nella richiesta, l'espressione "rivolti a faccia in giù" specifica che non andrà raffigurata l'immagine dei fanti ma il dorso delle carte.

Il livello di difficoltà è molto alto e, visti gli errori già commessi dai modelli, ci si aspetta che questi ultimi riescano a limitare i danni piuttosto che a soddisfare pienamente la richiesta.

La risposta di ChatGPT è riportata nelal Figura 4.10.

Il modello produce un risultato ottimo, raffigurando correttamente entrambe le mani e riportando i fanti di dorso. Inoltre, sembra correggere la generazione errata del numero avvenuta nell'immagine precedente (pur non includendo l'8 di picche).

La risposta di Gemini è riportata nella Figura 4.11.

Gemini, pur generando correttamente i fanti di dorso, genera due mani estranee al croupier generato in precedenza, producendo un'immagine con due individui in un contesto in cui la richiesta implicava un soggetto che adoperasse entrambe le mani.

La risposta di Grok è riportata nella Figura 4.12.

Figura 4.10: Quarta generazione di ChatGPT

Figura 4.11: Quarta generazione di Gemini

Grok sbaglia su tutti i fronti: raffigurando due mani sinistre e i fanti rivolti di corpo anziché di dorso, un risultato prevedibile, considerata la continuità degli errori commessi dal modello.

4.6 Conclusione Case Study per la generazione di immagini

In conclusione, il case study ha evidenziato come i tre modelli operino su livelli totalmente differenti nella generazione di immagini:

1. *ChatGPT* si è dimostrato il migliore con netto vantaggio; ogni sua generazione ha centrato il punto della richiesta e, fatta eccezione per alcune imprecisioni minori, i risultati sono sempre stati eccellenti.

2. *Gemini* ha compiuto un lavoro discreto, riportando risultati non sempre precisi, ma comunque riconducibili a una logica identificabile.

Figura 4.12: Quarta generazione di Grok

3. *Grok* ha prodotto risultati del tutto caotici, operando con una logica non definibile e producendo immagini del tutto inaccurate sia rispetto alla richiesta sia rispetto all'ordine che il contesto richiedeva.

In definitiva, il case study ha riportato come la generazione di immagini sia ancora un ambito molto complicato per i modelli di Intelligenza Artificiale Generativa e come la creazione di immagini giuste e contestualmente coerenti risulti ancora una sfida aperta per la maggior parte dei modelli attualmente disponibili.

CAPITOLO 5

Un case study relativo alla generazione di codice

Come ultimo case study, la tesi riporta l'interazione avvenuta con Claude al fine di ottenere codice Python funzionante. L'utilizzo dei modelli di Intelligenza Artificiale Generativa al fine di ottenere algoritmi funzionanti è una delle pratiche più diffuse sia a livello amatoriale sia a livello professionale e, con la sua diffusione, il dibattito nel settore informatico riguardo all'etica nel suo utilizzo è sempre più acceso. Questo case study ha lo scopo di approfondire questo tipo di generazione, verificando in che misura le richieste vengano soddisfatte all'aumentare della difficoltà.

5.1 La generazione di codice

Tramite l'utilizzo del linguaggio naturale, tra le richieste possibili da sottoporre agli LLM, vi è anche la possibilità di creare codice funzionante. Questo tipo di generazione comporta una serie di vantaggi notevoli per molti informatici, automatizzando processi ripetitivi e permettendo agli sviluppatori di concentrare maggiormente le loro risorse su attività complementari, accelerando in maniera significativa lo sviluppo di software.

Nonostante ciò, il dibattito globale su questo tipo di generazione risulta particolarmente vivace; infatti, se una fazione sostiene che tali tecnologie siano il futuro dell'informatica e che vanno perciò implementate attivamente nel processo lavorativo, l'altra sostiene che l'implementazione di questi strumenti rischi di mantenere il team umano troppo sconnesso con il lavoro che viene svolto o che, addirittura, questi strumenti rischiano di sostituire totalmente gli esseri umani professionisti del settore.

È chiaro che l'utilizzo di nuove tecnologie è un rischio per diversi posti di lavoro; non è la prima volta nella storia che dibattiti del genere sorgono a seguito di cambiamenti di questo tipo. Il case study non ha come obiettivo diretto stabilire se gli informatici attuali hanno futuro nella loro professione, ma quantomeno fornire un'indicazione concreta sul livello raggiunto dai modelli di Intelligenza Artificiale Generativa nella generazione di codice.

5.2 La scelta di Claude

La scelta di Claude per la generazione di codice in questo case study è motivata dalle prestazioni del modello in questo campo; di fatto, Claude figura sempre tra i modelli di Intelligenza Artificiale Generativa più performanti sul benchmark *SWE-bench* (standard adottato per la valutazione dei modelli generativi durante lo svolgimento di task di programmazione).

5.2.1 L'interfaccia di Claude

Nella Figura 5.1 viene riportata l'interfaccia di Claude.

Figura 5.1: Interfaccia di Claude

L'interfaccia, oltre ad accogliere l'utente con una frase cordiale, mette a disposizione i seguenti elementi:

- *Codice*: Claude mette a disposizione dell'utente una serie di input automatici per la generazione di codice; selezionando un'opzione, il campo di testo viene popolato automaticamente con una richiesta affine con il tipo di input selezionato.
- *Imparare*: il modello mette a disposizione, ancora una volta, input automatici ma, questa volta, per supportare le attività accademiche (come nel creare un piano di studio).
- *Scrivere*: premendo questo pulsante si potrà selezionare quale tipologia di elaborato scritto generare.
- *Vita quotidiana*: l'interfaccia mette a disposizione dell'utente una lista di attività di supporto nella vita di tutti i giorni (come creare una scheda di esercizi in palestra).
- *Scelta di Claude*: tra le opzioni messe a disposizione, questa volta non figurano richieste di tipo specifico, bensì una lista di input che possono essere di natura matematica, scientifica o alimentare; la lista viene aggiornata quotidianamente.
- +: il modello mette a disposizione la possibilità di allegare file di diverso tipo nella richiesta.
- *Versione in uso (Sonnet 4.6)*: Claude mette a disposizione dell'utente la possibilità di selezionare il modello con cui elaborare la richiesta, specificando il grado di consumo del modello selezionato (o "Basso" o "Alto").
- *Microfono*: tasto che rende possibile registrare l'input. Il modello in contemporanea attiva la modalità *speech-to-text* digitando in maniera autonoma le parole pronunciate nel corpo della richiesta.

L'interfaccia rende chiara la multimodalità del modello e la sua comodità d'uso, proponendo una serie di richieste precompilate in modo da rendere ancora più rapido il processo di utilizzo e l'efficienza di questo strumento.

5.3 Obiettivo dello studio

Lo studio che verrà riportato ha come metodologia d'approccio principale sottoporre a Claude una serie di input per la generazione di un programma calcolatore, richiedendo al modello di integrare, all'interno dell'algoritmo, formule complesse per la sua realizzazione. Durante lo studio, al modello verranno sottoposte una serie di richieste aggiuntive con lo scopo di alzare il livello di difficoltà sia nella logica dell'algoritmo sia nella matematica delle operazioni richieste. Al modello verrà sottoposta, anche, la richiesta di generare un' interfaccia grafica user-friendly. Il modello dovrà consentire di digitare i valori dei vari parametri, la visualizzazione di un grafico per consultare l'andamento della formula, alzando ancora di più il livello di complessità dell'algoritmo.

A Claude, inoltre, verrà sottoposta la richiesta di analizzare il proprio codice e riportare i bug trovati; successivamente il modello dovrà generare il codice corretto eliminando tutti gli errori precedentemente generati.

Lo studio ha come scopo documentare il livello attuale del modello preso in esame, mostrandone le capacità e i limiti nel creare un programma fedele ai prompt, robusto e con layout di qualità.

La formula su cui il modello dovrà generare il programma in grado di calcolare i valori finali in base a diversi parametri in input è quella dell'Interesse Composto.

5.3.1 L'interesse composto

La formula dell'interesse composto è la seguente:

$$A = P\left(1 + \frac{r}{n}\right)^{n \cdot t}$$

I parametri della formula sono riportati nella Tabella 5.1.

Variabile	Nome	Significato
A	Amount (Importo finale)	Il valore totale accumulato al termine del periodo, comprensivo di capitale e interessi.
P	Principal (Capitale iniziale)	La somma investita o prestata all'inizio.
r	Rate (Tasso annuo)	Il tasso di interesse annuale espresso in forma decimale (es. 5% $\rightarrow$ 0,05).
n	Number of compoundings	Il numero di volte in cui gli interessi vengono capitalizzati *per anno* (es. 12 = mensile, 4 = trimestrale, 1 = annuale).
t	Time (Tempo)	La durata dell'investimento espressa in anni.

Tabella 5.1: Descrizione dei parametri della formula dell'interesse composto

Dover implementare la formula, allestire un'interfaccia in grado di riportare l'andamento della funzione, auto-analizzare il proprio codice e generare, infine, un risultato privo di errori, costituisce un banco di prova molto impegnativo.

5.4 Le prime richieste

Il primo prompt ha lo scopo di creare lo scheletro di quello che sarà il programma finale; il modello dovrà generare un algoritmo in grado di calcolare la formula dell'interesse composto restituendo come output l'importo finale e accettando in input gli altri parametri riportati nella Tabella 5.1.

Il primo prompt è il seguente:

Utente

Crea una funzione Python che implementi la formula dell'interesse composto. La funzione deve accettare capitale iniziale, tasso annuale in decimale, anni e frequenza di capitalizzazione. Fai in modo che l'utente inserisca da terminale i parametri della funzione. Commenta ogni riga del codice in italiano.

Il codice Python generato è mostrato dal Listato 5.1.

```python
def calcola_interesse_composto(P: float, r: float, n: int, t: int)
    -> float:
    """
    Calcola il montante finale tramite la formula dell'interesse
       composto.

    Parametri:
        P (float): Capitale iniziale (in    )
        r (float): Tasso annuale espresso in decimale (es. 0.05 per
           il 5%)
        n (int):   Frequenza di capitalizzazione annua (es. 12 =
           mensile)
        t (int):   Durata dell'investimento in anni

    Ritorna:
        float: Il montante finale A = P * (1 + r/n)^(n*t)
    """

    # Calcola il tasso periodico dividendo il tasso annuale per la
       frequenza
    tasso_periodico = r / n

    # Calcola il numero totale di periodi di capitalizzazione
    periodi_totali = n * t

    # Applica la formula dell'interesse composto: A = P * (1 + r/n)
       ^(n*t)
    montante = P * (1 + tasso_periodico) ** periodi_totali

    # Restituisce il montante calcolato
    return montante


def leggi_float(messaggio: str, minimo: float = None) -> float:
    """Legge un numero decimale da terminale, ripetendo in caso di
       errore."""

```

```
31      while True:  # Ciclo infinito fino a input valido
32          try:
33              # Tenta di convertire l'input in numero float
34              valore = float(input(messaggio))
35
36              # Controlla che il valore rispetti il minimo, se
                    specificato
37              if minimo is not None and valore <= minimo:
38                  print(f"      Il valore deve essere maggiore di {
                        minimo}. Riprova.")
39                  continue  # Torna all'inizio del ciclo
40
41              return valore  # Restituisce il valore se valido
42
43          except ValueError:
44              # Gestisce il caso in cui l'input non sia un numero
45              print("      Inserisci un numero valido. Riprova.")
46
47
48  def leggi_int(messaggio: str, minimo: int = None) -> int:
49      """Legge un numero intero da terminale, ripetendo in caso di
            errore."""
50
51      while True:  # Ciclo infinito fino a input valido
52          try:
53              # Tenta di convertire l'input in numero intero
54              valore = int(input(messaggio))
55
56              # Controlla che il valore rispetti il minimo, se
                    specificato
57              if minimo is not None and valore < minimo:
58                  print(f"      Il valore deve essere almeno {minimo}.
                        Riprova.")
59                  continue  # Torna all'inizio del ciclo
60
61              return valore  # Restituisce il valore se valido
62
63          except ValueError:
64              # Gestisce il caso in cui l'input non sia un numero
                    intero
65              print("      Inserisci un numero intero valido. Riprova.
                    ")
66
67
68  def main():
69      """Funzione principale: raccoglie i dati e mostra il risultato."
            ""
70
71      # Intestazione del programma
72      print("=" * 50)
73      print("   CALCOLATORE DI INTERESSE COMPOSTO")
74      print("   Formula: A = P(1 + r/n)^(n  t)")
75      print("=" * 50)
76      print()
77
```

```
78      # Legge il capitale iniziale (deve essere positivo)
79      P = leggi_float("Capitale iniziale P (   ): ", minimo=0)
80
81      # Legge il tasso annuale in percentuale e lo converte in
            decimale
82      r_percentuale = leggi_float("Tasso annuale r (%): ", minimo=0)
83      r = r_percentuale / 100  # Converte la percentuale in decimale (
            es. 5%     0.05)
84
85      # Legge il numero di anni (almeno 1)
86      t = leggi_int("Durata in anni t: ", minimo=1)
87
88      # Legge la frequenza di capitalizzazione con un men
            descrittivo
89      print("\nFrequenza di capitalizzazione n:")
90      print("  1  = annuale   |  2  = semestrale")
91      print("  4  = trimestrale  |  12 = mensile")
92      print("  52 = settimanale  | 365 = giornaliera")
93      n = leggi_int("Inserisci n: ", minimo=1)
94
95      print()
96
97      # Chiama la funzione di calcolo con i parametri inseriti
98      montante = calcola_interesse_composto(P, r, n, t)
99
100     # Calcola gli interessi maturati come differenza tra montante e
            capitale iniziale
101     interessi = montante - P
102
103     # Mostra i risultati formattati con due cifre decimali
104     print("=" * 50)
105     print("   RISULTATI")
106     print("=" * 50)
107     print(f"  Capitale iniziale  :     {P:>12,.2f}")    # Capitale P
108     print(f"  Tasso annuale      :   {r_percentuale:>11.2f} %")   #
            Tasso in %
109     print(f"  Durata             :   {t:>11} anni")    # Anni t
110     print(f"  Capitalizzazione   :   {n:>11} volte/anno")   #
            Frequenza n
111     print("-" * 50)
112     print(f"  Interessi maturati :     {interessi:>12,.2f}")   #
            Interessi guadagnati
113     print(f"  Montante finale    :     {montante:>12,.2f}")    #
            Totale A
114     print("=" * 50)
115
116
117 # Punto di ingresso del programma: esegue main() solo se lo script
        avviato direttamente
118 if __name__ == "__main__":
119     main()
```

Listato 5.1: La prima generazione di Claude

Claude organizza il codice definendo due funzioni per leggere le variabili di tipo float e di tipo int. Viene effettuata la verifica per ogni parametro, assicurando che tutte le specifiche

vengano rispettate, altrimenti il programma viene interrotto. Le informazioni sviluppate sul terminale, subito dopo l'esecuzione del programma, vengono riportate nella Figura 5.2.

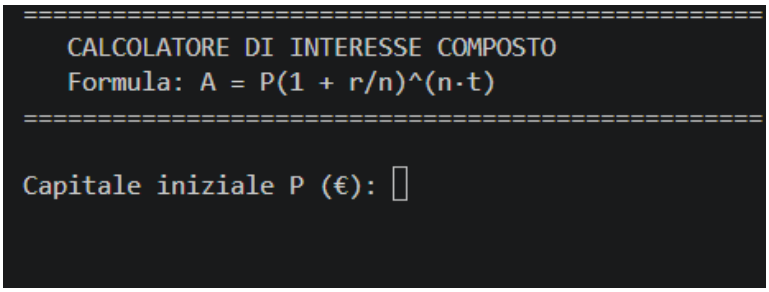

Figura 5.2: Terminale dopo la prima esecuzione

Nella Figura 5.3 viene riportato l'output una volta inseriti tutti i parametri.

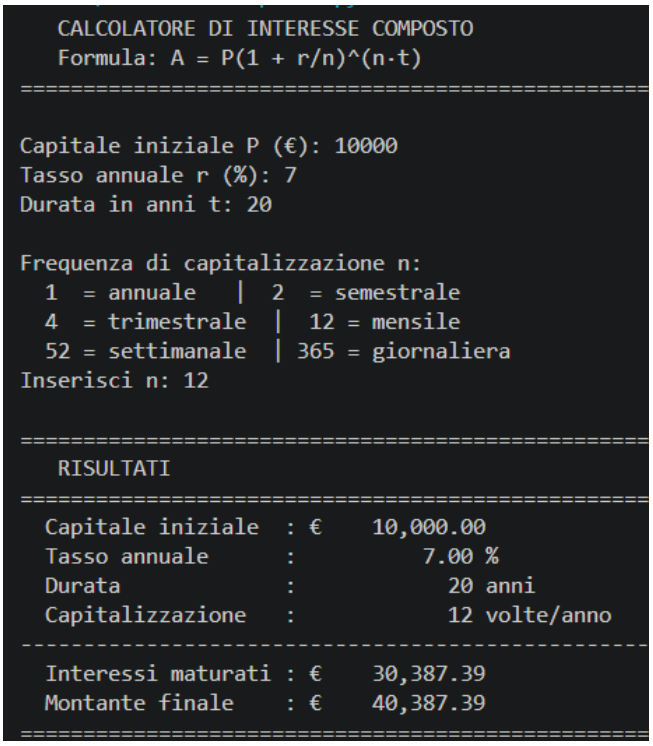

Figura 5.3: Output finale del primo prompt

Il programma, oltre a restituire il calcolo (matematicamente corretto) del montante finale, restituisce anche la cifra corrispondente agli interessi maturati: quest'ultima è pari alla differenza tra il montante finale e il capitale iniziale, e indica quanto ha guadagnato l'investimento nel tempo.

Il programma risulta essere completo e ben strutturato, con una logica semplice ed efficace, definendo:

- i metodi per leggere l'input :
  1. `leggi_float(messaggio, minimo)`: utilizzato per la lettura del capitale iniziale e del tasso annuale;
  2. `leggi_int(messaggio, minimo)`: utilizzato per la lettura della frequenza di capitalizzazione e la durata in anni.

- il metodo `def calcola_interesse_composto(P, r, n, t)` per calcolare il risultato

La specifica iniziale, quindi, può dirsi ampliamente soddisfatta.

5.4.1 Il secondo prompt

Se fino ad adesso è stato definito lo scheletro, adesso verranno implementate complicazioni logiche e matematiche ulteriori. L'algoritmo, infatti, dovrà essere in grado di considerare, tra i parametri in input, anche i versamenti mensili fissi. Nel calcolo finale, dunque, oltre a calcolare il montante finale, il programma dovrà applicare il tasso mensile a un capitale che viene incrementato mese per mese dai versamenti ricorrenti. Il secondo prompt è il seguente:

Utente

Modifica l'algoritmo precedente aggiungendo i versamenti mensili ricorrenti fissi.

Nel nuovo codice, Claude implementa le seguenti funzioni:

- `calcola_con_versamenti(P, r, t, versamento)` il cui codice viene riportato nel Listato 5.2.

```
1  def calcola_con_versamenti(P: float, r: float, t: float,
       versamento: float) -> dict:
2
3      # Calcola il tasso mensile dividendo il tasso annuale per 12
           mesi
4      tasso_mensile = r / 12
5
6      # Calcola il numero totale di mesi da simulare (anni    12)
7      mesi_totali = int(round(t * 12))
8
9      # Inizializza il saldo con il capitale iniziale
10     saldo = P
11
12     # Lista che registra il saldo a fine di ogni mese (utile per
           grafici o debug)
13     storico_mensile = []
14
15     # Ciclo che simula ogni singolo mese dell'investimento
16     for mese in range(1, mesi_totali + 1):
17
18         # PASSO 1: aggiunge il versamento mensile al saldo prima
               degli interessi
19         saldo += versamento
20
21         # PASSO 2: applica l'interesse mensile al saldo gi
               aggiornato col versamento
22         saldo *= (1 + tasso_mensile)
23
24         # Registra il saldo aggiornato nello storico (
               arrotondato a 2 decimali)
25         storico_mensile.append(round(saldo, 2))
26
27     # Calcola il totale effettivamente versato: capitale
```

```
        iniziale + tutti i versamenti mensili
28      totale_versato = P + (versamento * mesi_totali)
29
30      # Calcola gli interessi come differenza tra montante finale
            e somme versate
31      interessi_maturati = saldo - totale_versato
32
33      # Calcola il montante senza versamenti (solo interesse
            composto puro) per confronto
34      montante_senza_versamenti = P * math.pow(1 + tasso_mensile,
            mesi_totali)
35
36      # Restituisce il dizionario completo con tutti i risultati e
            lo storico
37      return {
38          "capitale_iniziale": P,
39          "versamento_mensile": versamento,
40          "totale_versato": totale_versato,
41          "montante_finale": saldo,
42          "interessi_maturati": interessi_maturati,
43          "montante_senza_versamenti": montante_senza_versamenti,
44          "contributo_versamenti": saldo -
                montante_senza_versamenti,
45          "storico_mensile": storico_mensile,
46      }
```

Listato 5.2: Codice del metodo per il calcolo con versamenti

Esaminando il risultato possiamo osservare che:

1. si aggiunge il versamento al saldo `(saldo += versamento)`
2. si applica l'interesse mensile sul saldo già aggiornato `(saldo *= 1 + r/12)`

Questo ordine è fondamentale: il versamento entra prima degli interessi, massimizzando il tempo di maturazione di ogni euro versato.

- `leggi_float_non_negativo(messaggio)` utilizzata per la lettura da terminale del versamento annuale; esso non accetta, come suggerisce il nome, input di numeri negativi. Il codice della funzione è mostrato nel Listato 5.3.

```
1   def leggi_float_non_negativo(messaggio: str) -> float:
2
3       # Ciclo infinito che si interrompe solo quando l'input
            valido
4       while True:
5           try:
6               # Legge e converte l'input in float
7               valore = float(input(messaggio))
8
9               # Accetta anche zero (l'utente può scegliere di non
                    versare nulla)
10              if valore < 0:
```

```
                print("      Il versamento non pu  essere
                    negativo. Riprova.\n")
                continue  # Torna all'inizio del ciclo

            # Restituisce il valore validato
            return valore

        except ValueError:
            # Gestisce il caso in cui l'utente inserisce un
                testo non numerico
            print("      Input non valido. Inserisci un numero (
                es. 200).\n")
```

Listato 5.3: Codice del metodo per leggere il float non negativo

La funzione itera il ciclo fino a quando il valore di input non è maggiore o uguale a 0.

- `mostra_storico_annuale(storico_mensile, P, versamento)` che stampa il saldo e il totale versato al termine di ogni anno; il codice è mostrato nel Listato 5.4.

```
def mostra_storico_annuale(storico_mensile: list, P: float,
    versamento: float) -> None:
    # Se lo storico    vuoto non c'   nulla da mostrare
    if not storico_mensile:
        return

    # Intestazione della tabella annuale
    print("  ANDAMENTO ANNUALE")
    print("  " + "    " * 44)
    print(f"  {'Anno':>6}  {'Totale versato':>16}  {'Saldo':>16}
        ")
    print("  " + "    " * 44)

    # Riga iniziale: anno 0, solo il capitale
    print(f"  {'0':>6}     {P:>15,.2f}     {P:>15,.2f}")

    # Itera sullo storico estraendo il saldo a fine di ogni anno
        (ogni 12 mesi)
    for anno in range(1, len(storico_mensile) // 12 + 1):

        # Indice dell'ultimo mese dell'anno corrente (base 0)
        indice_mese = anno * 12 - 1

        # Controlla che l'indice esista nella lista (l'ultimo
            anno potrebbe essere parziale)
        if indice_mese >= len(storico_mensile):
            break  # Esce dal ciclo se non ci sono abbastanza
                dati

        # Recupera il saldo a fine anno dallo storico mensile
        saldo_anno = storico_mensile[indice_mese]

        # Calcola il totale cumulativo versato fino a quel mese
```

```
29          versato_cumulativo = P + (versamento * (anno * 12))
30
31          # Stampa la riga con anno, totale versato e saldo
32          print(f"  {anno:>6}      {versato_cumulativo:>15,.2f}
                  {saldo_anno:>15,.2f}")
33
34      print("   " + "    " * 44 + "\n")
```

Listato 5.4: Codice del metodo per mostrare lo storico annuale

Eseguendo il codice e inserendo tutti i parametri in input, il terminale restituisce i valori calcolati come riportato nella Figura 5.4.

```
RIEPILOGO — INTERESSE COMPOSTO + VERSAMENTI

Capitale iniziale (P) : €   10,000.00
Versamento mensile    : €      100.00
Tasso annuale (r)     :         7.00%
Durata (t)            :          5.0 anni

Montante finale       : €   21,377.31
Totale versato        : €   16,000.00
Interessi maturati    : €    5,377.31

Montante senza versamenti     : € 14,176.25
Contributo dei versamenti     : €  7,201.05
Rendimento su versato         :       33.61%

ANDAMENTO ANNUALE

  Anno     Totale versato              Saldo

     0  €       10,000.00  €       10,000.00
     1  €       11,200.00  €       11,969.39
     2  €       12,400.00  €       14,081.14
     3  €       13,600.00  €       16,345.56
     4  €       14,800.00  €       18,773.67
     5  €       16,000.00  €       21,377.31
```

Figura 5.4: Output finale del secondo prompt

Il programma restituisce in output una tabella dettagliata, riportando il totale versato e il saldo maturato anno per anno. Poiché i versamenti avvengono su base mensile, viene esclusa la possibilità di inserire la frequenza di capitalizzazione. Nel caso in cui si decidesse di inserire come valore dei versamenti mensili una cifra nulla, l'output cambia come riportato in Figura 5.5.

Dunque, il programma risulta completo anche sotto questo punto di vista, manipolando alla perfezione i dati in input e implementando soluzioni in base a casi differenti.

5.5 L'aggiunta dell'interfaccia grafica

Fino a questo punto, Claude ha organizzato il proprio output tramite un utilizzo frequente di istruzioni `print()`; la fase successiva richiederà al modello di implementare l'utilizzo

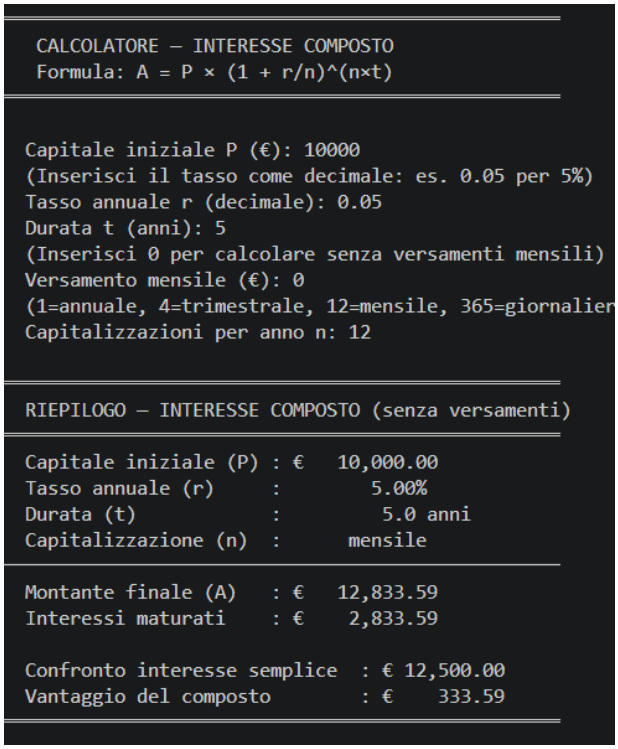

Figura 5.5: Output finale del secondo prompt senza versamenti mensili

della libreria *tkinter* per creare un'interfaccia grafica. A questo punto dello studio, verrà analizzata l'efficienza di tale interfaccia, monitorando come i dati verranno visualizzati in output.

Il terzo prompt è il seguente:

Utente

Ora crea un'interfaccia grafica con Tkinter per la funzione precedente. Deve avere: campi di input per P, r, n, t e i versamenti mensili, un pulsante 'Calcola', un'area di output che mostri il risultato finale e una tabella anno per anno. Il codice deve essere commentato.

Claude organizza il codice in tre sezioni:

1. La prima contiene tutte le funzioni di calcolo, non cambiando il codice di `calcola_interesse_composto(P, r, n, t)` per la formula diretta e di `calcola_con_versamenti(P, r, t, versamento)` per considerare i versamenti mensili. La maggiore modifica in questa sezione è l'aggiunta di `costruisci_storico_annuale(storico_mensile, P, versamento)` che ricopre ha lo scopo di convertire lo storico mensile in righe annuali per la tabella.

   Il codice per la costruzione dello storico annuale è mostrato nel Listato 5.5.

```
1 def costruisci_storico_annuale(storico_mensile: list, P: float,
      versamento: float) -> list:
2     # Riga anno 0: nessun interesse, solo il capitale iniziale
3     righe = [{"anno": 0, "versato": P, "saldo": P, "interessi":
          0.0}]
4
```

```
    anni_totali = len(storico_mensile) // 12   # Anni interi
        disponibili

    for anno in range(1, anni_totali + 1):
        idx = anno * 12 - 1                    # Indice del mese
            finale dell'anno

        if idx >= len(storico_mensile):
            break                              # Non ci sono
                pi  dati

        saldo_anno   = storico_mensile[idx]
        versato_cum  = P + versamento * (anno * 12)
        interessi_cum = saldo_anno - versato_cum

        righe.append({
            "anno":      anno,
            "versato":   versato_cum,
            "saldo":     saldo_anno,
            "interessi": interessi_cum,
        })

    return righe
```

Listato 5.5: Codice del metodo per costruire lo storico annuale

2. La seconda contiene la classe `AppInteresseComposto`, che assume il ruolo di finestra pricipale. Claude segue, nella costruzione di questa classe, il pattern della *separazione delle responsabilità*. Il modello, infatti, definisce cinque metodi ognuno con un compito unico:

   - `_costruisci_ui()`: coordina il layout generale; il codice è riportato nel Listato 5.6.

```
##
    ------------------------------------------------------------------

    # COSTRUZIONE DELL'INTERFACCIA
    #
       ------------------------------------------------------------------


    def _costruisci_ui(self):

        # Frame radice con padding interno     contiene
            tutto il contenuto
        root_frame = tk.Frame(self, bg=self.COLORE_SFONDO,
            padx=20, pady=16)
        root_frame.pack(fill="both", expand=True)

        # Configura le colonne del grid: sinistra (input) e
            destra (output) espandibili
        root_frame.columnconfigure(0, weight=1)    # Colonna
```

```
            input
        root_frame.columnconfigure(1, weight=2)    # Colonna
            output (più larga)
        root_frame.rowconfigure(1, weight=1)       # Riga
            contenuto si espande verticalmente

        # --- Titolo in cima ---
        self._crea_titolo(root_frame)

        # --- Pannello input (sinistra) ---
        self._crea_pannello_input(root_frame)

        # --- Pannello output (destra) ---
        self._crea_pannello_output(root_frame)
```

Listato 5.6: Codice per coordinare il layout

- `_crea_pannello_input()` e `_crea_pannello_output()`: costruiscono le due metà della finestra; nel primo vengono definiti i campi per l'inserimento da interfaccia dei parametri di interesse; nel secondo metodo, invece, viene definito il pannello di output (dove verranno riportati i risultati della formula e la tabella finale). Il codice dei due metodi viene riportato nel Listato 5.7.

```
def _crea_pannello_input(self, parent):

        # Card bianca con bordo sottile
        card = tk.Frame(
            parent,
            bg=self.COLORE_PANNELLO,
            relief="flat",
            bd=0,
            highlightbackground=self.COLORE_BORDO,
            highlightthickness=1,
        )
        card.grid(row=1, column=0, sticky="nsew", padx=(0,
            10), pady=0)
        card.columnconfigure(0, weight=1)    # La colonna si
            espande orizzontalmente

        # Contenuto interno con padding
        inner = tk.Frame(card, bg=self.COLORE_PANNELLO, padx
            =18, pady=16)
        inner.pack(fill="both", expand=True)
        inner.columnconfigure(1, weight=1)  # Campo di testo
             si espande

        # Intestazione della card
        tk.Label(
            inner,
            text="Parametri",
            font=self.FONT_LABEL_BOLD,
            bg=self.COLORE_PANNELLO,
            fg=self.COLORE_TESTO_2,
```

```
27          ).grid(row=0, column=0, columnspan=2, sticky="w",
                pady=(0, 12))
28
29          # --- Definizione dei campi: (riga, etichetta,
                variabile, hint) ---
30          # Ogni tupla descrive un campo da creare con
                _crea_campo()
31          campi = [
32              (1, "Capitale iniziale P (   )",      self.var_P
                    ,          "es. 10000"),
33              (2, "Tasso annuale r (%)",           self.var_r,
                            "es. 7       7%"),
34              (3, "Durata t (anni)",               self.var_t,
                            "es. 20"),
35              (4, "Capitalizzazioni/anno n",       self.var_n,
                            "1=ann    12=mens    365=giorn"),
36              (5, "Versamento mensile (   )",        self.
                    var_versamento, "0 = nessun versamento"),
37          ]
38
39          # Crea ogni campo usando il metodo helper dedicato
40          for riga, etichetta, variabile, hint in campi:
41              self._crea_campo(inner, riga, etichetta,
                    variabile, hint)
42
43          # --- Separatore orizzontale tra i campi e il
                pulsante ---
44          ttk.Separator(inner, orient="horizontal").grid(
45              row=6, column=0, columnspan=2, sticky="ew", pady
                    =16
46          )
47
48          # --- Pulsante Calcola ---
49          self.btn_calcola = tk.Button(
50              inner,
51              text="Calcola",
52              font=self.FONT_LABEL_BOLD,
53              bg=self.COLORE_PRIMARIO,
54              fg="#FFFFFF",
55              activebackground=self.COLORE_PRIMARIO_H,  #
                    Colore al click
56              activeforeground="#FFFFFF",
57              relief="flat",                                #
                    Pulsante piatto, senza bordo 3D
58              cursor="hand2",                               #
                    Cursore a mano al passaggio
59              padx=0,
60              pady=10,
61              command=self._on_calcola,                     #
                    Callback al click
62          )
63          self.btn_calcola.grid(row=7, column=0, columnspan=2,
                sticky="ew")
64
65          # Effetti hover: schiarisce/scurisce il pulsante al
```

```
            passaggio del mouse
66          self.btn_calcola.bind("<Enter>", lambda e: self.
               btn_calcola.config(bg=self.COLORE_PRIMARIO_H))
67          self.btn_calcola.bind("<Leave>", lambda e: self.
               btn_calcola.config(bg=self.COLORE_PRIMARIO))
68
69          # --- Nota informativa sotto il pulsante ---
70          tk.Label(
71              inner,
72              text="Con versamento > 0 la capitalizzazione
                   mensile (n ignorato)",
73              font=self.FONT_PICCOLO,
74              bg=self.COLORE_PANNELLO,
75              fg=self.COLORE_TESTO_2,
76              wraplength=220,          # Vai a capo
                   automaticamente oltre 220px
77              justify="left",
78          ).grid(row=8, column=0, columnspan=2, sticky="w",
               pady=(10, 0))
79
80
81           def _crea_pannello_output(self, parent):
82          """Crea il pannello destro con le card dei risultati
                e la tabella annuale."""
83
84          # Frame contenitore per l'intera area di output
85          frame_output = tk.Frame(parent, bg=self.
               COLORE_SFONDO)
86          frame_output.grid(row=1, column=1, sticky="nsew")
87          frame_output.columnconfigure(0, weight=1)
88          frame_output.rowconfigure(1, weight=1)    # La
               tabella occupa lo spazio restante
89
90          # --- Card superiore: metriche di riepilogo ---
91          self._crea_card_metriche(frame_output)
92
93          # --- Tabella annuale in basso ---
94          self._crea_tabella_annuale(frame_output)
```

Listato 5.7: Codice per creare il pannello di input e output

- `_crea_card_metriche()` e `_crea_tabella_annuale()`: i due metodi si occupano di definire lo stile e l'intestazione del risultato finale. Il primo metodo crea la card in cima all'output che riporterà le metriche principali, il secondo crea la tabella definendo la struttura delle righe e delle colonne. Il codice dei due metodi è riportato nel Listato 5.8.

```
1   def _crea_card_metriche(self, parent):
2
3           # Card bianca con bordo
4           card = tk.Frame(
5               parent,
6               bg=self.COLORE_PANNELLO,
```

```
            highlightbackground=self.COLORE_BORDO,
            highlightthickness=1,
        )
        card.grid(row=0, column=0, sticky="ew", pady=(0, 10))
        card.columnconfigure(0, weight=1)

        # Contenuto con padding interno
        inner = tk.Frame(card, bg=self.COLORE_PANNELLO, padx=18, pady=14)
        inner.pack(fill="both", expand=True)
        inner.columnconfigure((0, 1, 2), weight=1)   # 3 colonne uguali per le metriche

        # Intestazione della card
        tk.Label(
            inner,
            text="Risultati",
            font=self.FONT_LABEL_BOLD,
            bg=self.COLORE_PANNELLO,
            fg=self.COLORE_TESTO_2,
        ).grid(row=0, column=0, columnspan=3, sticky="w", pady=(0, 12))

        # --- Griglia delle metriche: 3 colonne ---
        # Ogni metrica    un (label, variabile_valore, colore_valore)
        # Le variabili vengono create e tenute come attributi per aggiornarle dopo il calcolo
        self.var_montante   = tk.StringVar(value="   ")
        self.var_versato    = tk.StringVar(value="   ")
        self.var_interessi  = tk.StringVar(value="   ")
        self.var_rendimento = tk.StringVar(value="   ")
        self.var_semplice   = tk.StringVar(value="   ")
        self.var_vantaggio  = tk.StringVar(value="   ")

        # Definizione delle 6 metriche da mostrare in griglia 2 3
        metriche = [
            ("Montante finale",       self.var_montante,
               self.COLORE_SUCCESSO, 1, 0),
            ("Totale versato",        self.var_versato,
               self.COLORE_TESTO,    1, 1),
            ("Interessi maturati",    self.var_interessi,
               self.COLORE_SUCCESSO, 1, 2),
            ("Rendimento su versato", self.var_rendimento,
               self.COLORE_TESTO,    2, 0),
            ("Int. semplice (cfr.)",  self.var_semplice,
               self.COLORE_TESTO_2,  2, 1),
            ("Vantaggio composto",    self.var_vantaggio,
               self.COLORE_SUCCESSO, 2, 2),
        ]

        # Crea ogni cella metrica (etichetta grigia sopra, valore colorato sotto)
```

```
48          for label_testo, variabile, colore, riga, colonna in
                metriche:
49              cella = tk.Frame(inner, bg=self.COLORE_PANNELLO)
50              cella.grid(row=riga, column=colonna, sticky="ew"
                    , padx=6, pady=4)
51
52              # Etichetta descrittiva in grigio secondario
53              tk.Label(
54                  cella,
55                  text=label_testo,
56                  font=self.FONT_PICCOLO,
57                  bg=self.COLORE_PANNELLO,
58                  fg=self.COLORE_TESTO_2,
59                  anchor="w",
60              ).pack(anchor="w")
61
62              # Valore numerico con font pi  grande e colore
                    distintivo
63              tk.Label(
64                  cella,
65                  textvariable=variabile,
66                  font=("Helvetica", 13, "bold") if riga == 1
                        and colonna == 0 else self.
                        FONT_RISULTATO,
67                  bg=self.COLORE_PANNELLO,
68                  fg=colore,
69                  anchor="w",
70              ).pack(anchor="w")
71
72      def _crea_tabella_annuale(self, parent):
73          """Crea la tabella con l'andamento anno per anno
                usando ttk.Treeview."""
74
75          # Frame contenitore della tabella
76          frame_tab = tk.Frame(
77              parent,
78              bg=self.COLORE_PANNELLO,
79              highlightbackground=self.COLORE_BORDO,
80              highlightthickness=1,
81          )
82          frame_tab.grid(row=1, column=0, sticky="nsew")
83          frame_tab.columnconfigure(0, weight=1)
84          frame_tab.rowconfigure(1, weight=1)    # La tabella
                cresce con la finestra
85
86          # Intestazione della sezione tabella
87          frame_header = tk.Frame(frame_tab, bg=self.
                COLORE_PANNELLO, padx=18, pady=10)
88          frame_header.grid(row=0, column=0, columnspan=2,
                sticky="ew")
89
90          tk.Label(
91              frame_header,
92              text="Andamento annuale",
93              font=self.FONT_LABEL_BOLD,
```

```
94              bg=self.COLORE_PANNELLO,
95              fg=self.COLORE_TESTO_2,
96          ).pack(side="left")
97
98          # --- Configurazione stile ttk per la Treeview ---
99          stile = ttk.Style()
100         stile.theme_use("clam")    # Tema "clam" permette
                personalizzazione colori
101
102         # Stile dell'intestazione della tabella (riga dei
                titoli colonna)
103         stile.configure(
104             "IC.Treeview.Heading",
105             background=self.COLORE_HEADER_TAB,
106             foreground=self.COLORE_TESTO,
107             font=self.FONT_LABEL_BOLD,
108             relief="flat",
109             borderwidth=0,
110         )
111
112         # Stile delle righe della tabella
113         stile.configure(
114             "IC.Treeview",
115             background=self.COLORE_RIGA_PARI,
116             foreground=self.COLORE_TESTO,
117             fieldbackground=self.COLORE_RIGA_PARI,
118             font=self.FONT_MONO,
119             rowheight=26,
120             borderwidth=0,
121         )
122
123         # Stile della riga selezionata
124         stile.map(
125             "IC.Treeview",
126             background=[("selected", "#B5D4F4")],    #
                    Azzurro chiaro per selezione
127             foreground=[("selected", self.COLORE_TESTO)],
128         )
129
130         # --- Definizione delle colonne della Treeview ---
131         colonne = ("anno", "versato", "saldo", "interessi",
                "rendimento")
132
133         self.tabella = ttk.Treeview(
134             frame_tab,
135             columns=colonne,
136             show="headings",     # Nasconde la colonna
                    fantasma di default
137             style="IC.Treeview",
138             selectmode="browse",   # Selezione riga singola
139         )
140
141         # Intestazioni e larghezze di ogni colonna
142         self.tabella.heading("anno",       text="Anno")
143         self.tabella.heading("versato",    text="Totale
```

```
            versato")
144         self.tabella.heading("saldo",      text="Saldo")
145         self.tabella.heading("interessi",  text="Interessi")
146         self.tabella.heading("rendimento", text="Rend. %")
147
148         self.tabella.column("anno",       width=60,  anchor=
                "center")
149         self.tabella.column("versato",    width=140, anchor=
                "e")   # Allineato a destra
150         self.tabella.column("saldo",      width=140, anchor=
                "e")
151         self.tabella.column("interessi",  width=130, anchor=
                "e")
152         self.tabella.column("rendimento", width=80,  anchor=
                "e")
153
154         # Tag per alternare i colori delle righe pari/
                dispari
155         self.tabella.tag_configure("pari",    background=
                self.COLORE_RIGA_PARI)
156         self.tabella.tag_configure("dispari", background=
                self.COLORE_RIGA_DISPARI)
157
158         # Posiziona la Treeview nel frame (si espande in
                entrambe le direzioni)
159         self.tabella.grid(row=1, column=0, sticky="nsew",
                padx=(18, 0), pady=(0, 18))
160
161         # --- Scrollbar verticale collegata alla Treeview
                ---
162         scrollbar = ttk.Scrollbar(
163             frame_tab,
164             orient="vertical",
165             command=self.tabella.yview,   # Collega la
                    scrollbar alla tabella
166         )
167         scrollbar.grid(row=1, column=1, sticky="ns", pady
                =(0, 18), padx=(0, 8))
168
169         # Dice alla Treeview di aggiornare la scrollbar
                quando l'utente scorre
170         self.tabella.configure(yscrollcommand=scrollbar.set)
```

Listato 5.8: Codice per mostrare le card metriche e la tabella annuale

3. Nella terza sezione viene definita la logica interattiva, che avviene per intero all'interno del metodo `on_calcola()`. Questo metodo, infatti, ogni volta che viene premuto il pulsante *Calcola*, legge e valida i parametri passati in input dall'utente e li usa per eseguire il calcolo e aggiornare l'output. Il codice viene riportato nel Listato 5.9.

```
1  def _on_calcola(self):
2
3
```

```
4           # --- Lettura e validazione dei parametri di input ---
5           try:
6               # Legge il capitale iniziale e controlla che sia
                    positivo
7               P = float(self.var_P.get().replace(",", "."))
8               if P <= 0:
9                   raise ValueError("Il capitale P deve essere
                        maggiore di zero.")
10
11              # Legge il tasso percentuale e lo converte in
                    decimale (es. 7%     0.07)
12              r_pct = float(self.var_r.get().replace(",", "."))
13              if r_pct <= 0:
14                  raise ValueError("Il tasso r deve essere
                        maggiore di zero.")
15              r = r_pct / 100    # Conversione da percentuale a
                    decimale
16
17              # Legge la durata in anni
18              t = float(self.var_t.get().replace(",", "."))
19              if t <= 0:
20                  raise ValueError("La durata t deve essere
                        maggiore di zero.")
21
22              # Legge la frequenza di capitalizzazione (intero >=
                    1)
23              n = int(self.var_n.get())
24              if n < 1:
25                  raise ValueError("Le capitalizzazioni n devono
                        essere almeno 1.")
26
27              # Legge il versamento mensile (p u  essere zero)
28              versamento = float(self.var_versamento.get().replace
                    (",", "."))
29              if versamento < 0:
30                  raise ValueError("Il versamento mensile non  p u
                         essere negativo.")
31
32          except ValueError as e:
33              # Mostra una finestra di errore con il messaggio
                    specifico
34              messagebox.showerror("Input non valido", str(e))
35              return   # Esce senza procedere al calcolo
36
37          # --- Esecuzione del calcolo nella modalit  appropriata
                 ---
38          if versamento == 0:
39              # Modalit  senza versamenti: usa la formula chiusa
40              risultati = calcola_interesse_composto(P=P, r=r, t=t
                    , n=n)
41          else:
42              # Modalit  con versamenti: usa la simulazione
                    mensile
43              risultati = calcola_con_versamenti(P=P, r=r, t=t,
                    versamento=versamento)
```

```
44
45          # --- Aggiornamento delle card dei risultati ---
46          self._aggiorna_metriche(risultati)
47
48          # --- Aggiornamento della tabella annuale ---
49          storico = costruisci_storico_annuale(
50              risultati["storico_mensile"],
51              P=P,
52              versamento=versamento,
53          )
54          self._aggiorna_tabella(storico, P
```

Listato 5.9: Codice del metodo per validare i parametri in input

Il metodo, per popolare i campi delle metriche e della tabella, richiama le due seguenti funzioni:

- `_aggiorna_metriche()`: aggiorna le sei card numeriche in cima al pannello di output dopo ogni calcolo.
- `_aggiorna_tabella()`: aggiorna la Treeview con l'andamento anno per anno

Il codice dei due metodi è riportato nel Listato 5.10.

```
1  def _aggiorna_metriche(self, risultati: dict):
2
3          # Helper locale per formattare un importo in euro con
               separatore migliaia
4          def fmt_euro(v: float) -> str:
5              return f"    {v:,.2f}"
6
7          # Aggiorna il montante finale
8          self.var_montante.set(fmt_euro(risultati["
               montante_finale"]))
9
10         # Aggiorna il totale versato
11         self.var_versato.set(fmt_euro(risultati["totale_versato"
               ]))
12
13         # Aggiorna gli interessi maturati
14         self.var_interessi.set(fmt_euro(risultati["
               interessi_maturati"]))
15
16         # Calcola e aggiorna il rendimento percentuale sulle
               somme versate
17         totale_versato = risultati["totale_versato"]
18         if totale_versato > 0:
19             rend = risultati["interessi_maturati"] /
                   totale_versato * 100
20             self.var_rendimento.set(f"{rend:.1f}%")
21         else:
22             self.var_rendimento.set("    ")
23
```

```
24          # Aggiorna interesse semplice di confronto (solo in
                modalit  senza versamenti)
25          if "interesse_semplice_confronto" in risultati:
26              self.var_semplice.set(fmt_euro(risultati["
                    interesse_semplice_confronto"]))
27              self.var_vantaggio.set(fmt_euro(risultati["
                    guadagno_extra_composto"]))
28          else:
29              # In modalit  con versamenti mostra il contributo
                    dei versamenti
30              self.var_semplice.set(fmt_euro(risultati.get("
                    montante_senza_versamenti", 0)))
31              self.var_vantaggio.set(fmt_euro(risultati.get("
                    contributo_versamenti", 0)))
32
33      def _aggiorna_tabella(self, storico: list, P: float):
34
35          # Cancella tutte le righe precedenti dalla tabella
36          for item in self.tabella.get_children():
37              self.tabella.delete(item)
38
39          # Inserisce una riga per ogni anno nello storico
40          for i, riga in enumerate(storico):
41
42              # Calcola il rendimento percentuale rispetto al
                    totale versato quell'anno
43              if riga["versato"] > 0 and riga["anno"] > 0:
44                  rend_pct = riga["interessi"] / riga["versato"] *
                        100
45                  rend_str = f"{rend_pct:.1f}%"
46              else:
47                  rend_str = "    "    # Anno 0: nessun rendimento
                        ancora
48
49              # Valori formattati per le celle della tabella
50              valori = (
51                  str(riga["anno"]),                  # Anno (
                        intero)
52                  f"    {riga['versato']:>12,.2f}",      # Totale
                        versato cumulativo
53                  f"    {riga['saldo']:>12,.2f}",        # Saldo a
                        fine anno
54                  f"    {riga['interessi']:>11,.2f}",    #
                        Interessi cumulativi
55                  rend_str,                           #
                        Rendimento percentuale
56              )
57
58              # Determina il tag per alternare i colori: pari/
                    dispari
59              tag = "dispari" if i % 2 else "pari"
60
61              # Inserisce la riga nella Treeview
62              self.tabella.insert("", "end", values=valori, tags=(
                    tag,))
```

Listato 5.10: Codice dei metodi per aggiornare le metriche e la tabella

Al termine della generazione, il risultato del codice produce il risultato riportato dalla Figura 5.6.

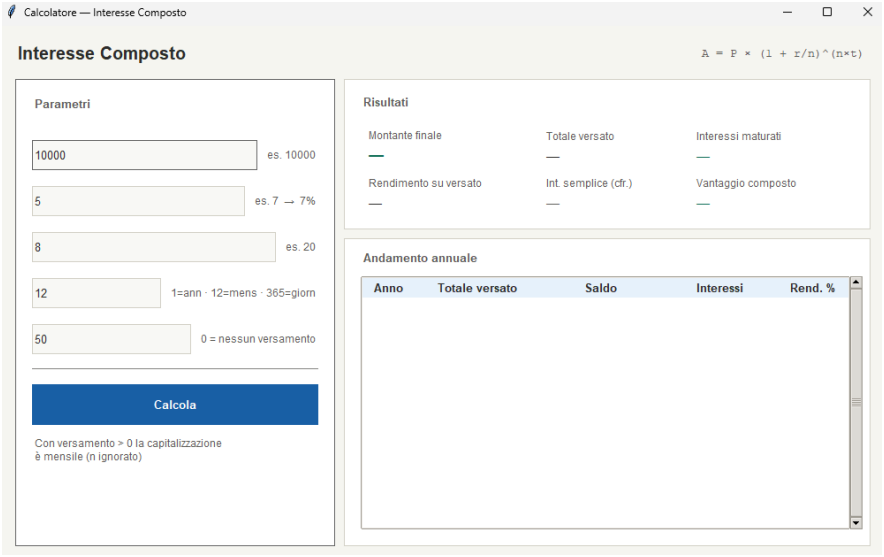

Figura 5.6: Prima interfaccia grafica

Premendo *Calcola*, l'interfaccia cambia come riportato nella Figura 5.7.

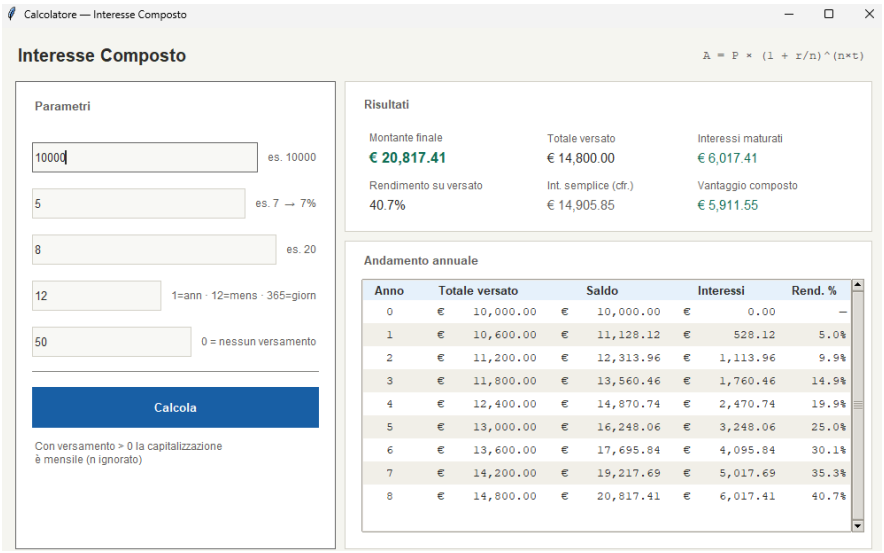

Figura 5.7: Risultato della prima interfaccia grafica

I calcoli sono matematicamente corretti; nella tabella in ogni riga viene riportato l'andamento del saldo, degli interessi e del rendimento all'aumentare del capitale versato (in questo caso l'aumento del totale versato è di 600 euro all'anno, per via del versamento mensile di 50 euro definito nell'ultimo campo di input). Il programma risulta completo e funzionante, Claude dimostra una notevole capacità nello strutturare il codice, separando con chiarezza la logica di calcolo dalla gestione dell'interfaccia grafica e dalla visualizzazione dei dati.

5.5.1 Aggiunta del grafico

Nel prompt precedente, è stato richiesto a Claude di definire un'interfaccia grafica per il calcolo dell'interesse composto e di definire una tabella per riportare l'andamento della formula nel tempo. Nel prompt successivo, verrà richiesto di implementare nell'interfaccia l'opzione per consultare un grafico con ascissa il tempo e ordinata il valore in euro. Il programma dovrà, dunque, essere in grado di riportare l'andamento della curva del capitale in funzione del tempo. Verrà richiesto esplicitamente di usare la libreria *Matplotlib*, che rappresenta la scelta standard per incorporare grafici in Python.

Il prompt è il seguente:

Utente

Aggiungi all'interfaccia Tkinter un grafico Matplotlib incorporato che mostri l'andamento del capitale anno per anno. Il grafico deve avere: asse X con gli anni, asse Y con il valore in euro, due linee distinte: una per il capitale versato e una per il capitale con interessi.

Il programma è stato modificato su tre livelli distinti: le importazioni, la struttura del layout e la logica di disegno:

1. *Importazioni aggiunte*:
   - `import matplotlib` : importa il pacchetto principale di Matplotlib.
   - `matplotlib.use("TkAgg")` importazione che specifica di integrare come backend il sistema di finestre fornito da Tkinter. Senza questa riga Matplotlib userebbe il backend di default del sistema operativo aprendo altre finestre separate, invece che inserire il grafico all'interno dell'interfaccia già esistente.
   - `from matplotlib.figure import Figure` :importa la classe `Figure` che rappresenta l'intera area di disegno in maniera autonoma dalla finestra. Infatti altri metodi usati con Matplotlib, come `plt.figure()` e `plt.show()`, presuppongono che l'interfaccia sia gestita da Matplotlib, problematica che va contro la struttura del progetto. Claude evita questo problema facendo in modo che l'interfaccia gestita da Tkinter usi la classe `Figure` per istanziare un oggetto puro senza una logica di visualizzazione propria.
   - `import FigureCanvasTkAgg`: effettua l'importazione con l'obiettivo di creare un ponte tra Matplotlib e Tkinter, `FigureCanvasTkAgg`, infatti, è una classe che considera una figura `Figure` e la trasforma in un widget Tkinter, rendendo possibile il suo posizionamento tramite i metodi `grid()` e `pack()`, come il resto dei campi dell'interfaccia.
2. *Modifica del layout*: aggiungere il grafico a un layout con già presenti le metriche e la tabella avrebbe reso la tabella illegibile e il grafico troppo stretto. Claude adotta la soluzione di introdurre un `ttk.Notebook`, in modo da rendere possibile selezionare una scheda per visualizzare il tipo di output desiderato.
3. `_crea_grafico()` ha un compito costruttivo; non disegna dati reali ma permette di creare la struttura Matplotlib per ospitare il grafico e collegare la struttura a Tkinter tramite `FigureCanvasTkAgg`. Il codice del metodo è riportato nel Listato 5.11.

```
1 def _crea_grafico(self, parent):
2
3         parent.columnconfigure(0, weight=1)
```

```
4           parent.rowconfigure(0, weight=1)    # Il canvas cresce
                con la finestra
5
6           # --- Crea la Figure Matplotlib ---
7           # figsize in pollici; dpi=100     1 pixel = 1 unit
                schermo circa
8           self.figura = Figure(figsize=(6, 4), dpi=100)
9
10          # Imposta il colore di sfondo della Figure uguale allo
                sfondo app
11          self.figura.patch.set_facecolor(self.COLORE_PANNELLO)
12
13          # Aggiunge un solo Axes (pannello di disegno) alla
                Figure
14          # rect = [left, bottom, width, height] in coordinate
                normalizzate
15          self.ax = self.figura.add_axes(
16              [0.12, 0.14, 0.82, 0.72],    # Margini calibrati per
                    etichette e titolo
17              facecolor="#FAFAF7",          # Sfondo leggermente
                    caldo per l'area del plot
18          )
19
20          # Stato iniziale: messaggio placeholder finch  non si
                calcola
21          self._grafico_placeholder()
22
23          # --- Crea il canvas Tkinter che ospita la Figure ---
24          # FigureCanvasTkAgg collega la Figure a un widget
                Tkinter
25          self.canvas = FigureCanvasTkAgg(self.figura, master=
                parent)
26
27          # get_tk_widget() restituisce il widget Tk sottostante,
                usabile con grid/pack
28          self.canvas.get_tk_widget().grid(
29              row=0, column=0, sticky="nsew", padx=10, pady=10,
30          )
31
32          # Disegna il canvas per la prima volta
33          self.canvas.draw()
```

Listato 5.11: Codice del metodo per costruire la struttura del grafico

4. `_aggiorna_grafico()`: è un metodo chiamato ad ogni click su *Calcola*. Il suo compito è disegnare il grafico con i nuovi dati (cancellando all'inizio, qualora esistessero, quelli precedenti). Il codice del metodi è riportato nel Listato 5.12.

```
1   def _aggiorna_grafico(self, storico: list, r_pct: float,
        versamento: float):
2
3
4           # Estrae le serie dati dallo storico
```

```
        anni    = [riga["anno"]    for riga in storico]   # Asse X: anni
        versati = [riga["versato"] for riga in storico]   # Linea 1: versato cumulativo
        saldi   = [riga["saldo"]   for riga in storico]   # Linea 2: saldo con interessi

        # --- Pulizia del grafico precedente ---
        self.ax.clear()

        # Ripristina lo sfondo dell'area del plot dopo clear()
        self.ax.set_facecolor("#FAFAF7")

        # --- Area riempita tra le due curve ---
        # fill_between() colora la zona tra "versato" e "saldo" con trasparenza
        self.ax.fill_between(
            anni, versati, saldi,
            color=self.COLORE_FILL_INTERESSI,    # Azzurro pastello
            alpha=0.5,                           # Semitrasparente
            zorder=1,                            # Disegnato sotto le linee
        )

        # --- Linea 1: Capitale versato (grigia, tratteggiata) ---
        self.ax.plot(
            anni, versati,
            color=self.COLORE_LINEA_VERSATO,     # Grigio
            linewidth=2,
            linestyle="--",                      # Tratteggiata per distinguerla visivamente
            marker="o",                          # Punto per ogni anno
            markersize=4,
            markerfacecolor=self.COLORE_PANNELLO, # Centro bianco del marker
            markeredgecolor=self.COLORE_LINEA_VERSATO,
            markeredgewidth=1.5,
            label="Capitale versato",            # Testo per la legenda
            zorder=3,                            # Sopra il fill, sotto i marker del saldo
        )

        # --- Linea 2: Capitale + interessi (blu, continua) ---
        self.ax.plot(
            anni, saldi,
            color=self.COLORE_LINEA_SALDO,       # Blu primario
            linewidth=2.5,
            linestyle="-",                       # Continua
            marker="o",
            markersize=5,
```

```
47              markerfacecolor=self.COLORE_PANNELLO,
48              markeredgecolor=self.COLORE_LINEA_SALDO,
49              markeredgewidth=2,
50              label="Capitale + interessi",
51              zorder=4,
52          )
53
54          # --- Titolo dinamico del grafico ---
55          # Costruisce il titolo in base alla modalit  attiva
56          if versamento > 0:
57              titolo = f"Interesse composto       r = {r_pct:.1f}%
                    |  versamento    {versamento:,.0f}/mese"
58          else:
59              titolo = f"Interesse composto       tasso annuale {
                    r_pct:.1f}%"
60
61          self.ax.set_title(
62              titolo,
63              fontsize=10,
64              fontweight="bold",
65              color=self.COLORE_TESTO,
66              pad=10,                     # Spazio tra titolo e
                    bordo superiore del plot
67          )
68
69          # --- Etichette degli assi ---
70          self.ax.set_xlabel(
71              "Anni",
72              fontsize=9,
73              color=self.COLORE_TESTO_2,
74              labelpad=6,
75          )
76          self.ax.set_ylabel(
77              "Valore (   )",
78              fontsize=9,
79              color=self.COLORE_TESTO_2,
80              labelpad=6,
81          )
82
83          # --- Formattazione asse X ---
84          # Tick solo sugli anni interi; se ci sono molti anni si
                salta uno su due
85          self.ax.set_xlim(left=0, right=max(anni) if anni else 1)
86          max_anni = max(anni) if anni else 1
87          step_x   = 2 if max_anni > 20 else 1   # Ogni 2 anni se
                il periodo    lungo
88          self.ax.set_xticks(range(0, max_anni + 1, step_x))
89          self.ax.tick_params(axis="x", labelsize=8, colors=self.
                COLORE_TESTO_2)
90
91          # --- Formattazione asse Y ---
92          # Usa il formattatore italiano: separatore migliaia con
                virgola, prefisso
93          def fmt_euro_asse(valore, _pos):
94              """Formattatore per i tick dell'asse Y: mostra
```

```
                10k , 100k , ecc."""
95              if valore >= 1_000_000:
96                  return f"  {valore/1_000_000:.1f}M"
97              if valore >= 1_000:
98                  return f"  {valore/1_000:.0f}k"
99              return f"  {valore:.0f}"
100
101         self.ax.yaxis.set_major_formatter(mticker.FuncFormatter(
                fmt_euro_asse))
102         self.ax.tick_params(axis="y", labelsize=8, colors=self.
                COLORE_TESTO_2)
103
104         # --- Griglia di sfondo ---
105         self.ax.grid(
106             True,
107             which="major",
108             axis="y",                       # Solo linee
                    orizzontali
109             color=self.COLORE_BORDO,
110             linewidth=0.6,
111             linestyle=":",                  # Punteggiata, meno
                    invadente
112             zorder=0,                       # Sotto tutto il
                    resto
113         )
114
115         # Rimuove le cornici destra e in alto (stile "clean")
116         self.ax.spines["top"].set_visible(False)
117         self.ax.spines["right"].set_visible(False)
118         self.ax.spines["left"].set_color(self.COLORE_BORDO)
119         self.ax.spines["bottom"].set_color(self.COLORE_BORDO)
120
121         # --- Legenda ---
122         self.ax.legend(
123             loc="upper left",               # Posizione: angolo
                    superiore sinistro
124             fontsize=9,
125             framealpha=0.92,                # Sfondo quasi opaco
                     per leggibilit
126             edgecolor=self.COLORE_BORDO,    # Bordo della
                    legenda
127             facecolor=self.COLORE_PANNELLO,
128         )
129
130         # --- Ridisegna il canvas Tkinter ---
131         # draw_idle()    p i  efficiente di draw() in un loop UI
                : aggiorna solo se necessario
132         self.canvas.draw_idle()
```

Listato 5.12: Codice del metodo per aggiornare il grafico

Il codice permette al grafico di cambiare la propria struttura, perchè verifica il numero degli anni definito dall'utente; se il periodo è lungo nell'ascissa verranno riportati meno anni, rendendo il risultato più leggibile.

5. `costruisci_storico_da_formula()`: il metodo alimenta il grafico qualora non venissero specificati versamenti mensili ricorrenti. Il totale versato, in questa modalità, rimane invariato nel tempo; dunque verrà raffigurato come una retta grigia. Questo metodo, quando attivato per via dei versamenti nulli, sostituisce la funzione di `costruisci_storico_annuale()`, costruendo lo storico applicando la formula diretta per ogni anno. Il codice della funzione è riportato nel Listato 5.13.

```python
def costruisci_storico_da_formula(P: float, r: float, t: float,
    n: int) -> list:

    anni = int(t)                                   # Numero di anni
        interi

    # Anno 0: solo il capitale, nessun interesse
    righe = [{"anno": 0, "versato": P, "saldo": P, "interessi":
        0.0}]

    for anno in range(1, anni + 1):
        # Applica la formula per ogni anno intero
        saldo_anno    = P * math.pow(1 + r / n, n * anno)
        interessi_cum = saldo_anno - P              # Il versato
            solo P (nessun versamento)

        righe.append({
            "anno":      anno,
            "versato":   P,                         # Senza
                versamenti il versato rimane P
            "saldo":     saldo_anno,
            "interessi": interessi_cum,
        })

    return righe
```

Listato 5.13: Codice del metodo per costruire il grafico senza versamenti

Non definendo questo metodo, la sola funzione di `costruisci_storico_annuale()` risulterebbe insufficiente, perché non esiste una simulazione mese per mese e il grafico non avrebbe dati su cui disegnare le curve.

Il codice generato, produce il risultato raffigurato nella Figura 5.8.

Claude, quindi, implementa la possibilità di cliccare su *Grafico* per poter consultare l'andamento della curva. Nella Figura 5.9 viene riportata l'interfaccia dopo aver premuto su *Calcola* e *Grafico* senza specificare versamenti mensili.

Nella Figura 5.10 viene riportato il risultato col medesimo procedimento ma specificando versamenti mensili ricorrenti pari a 100 euro.

Come previsto, nella versione senza versamenti, il capitale versato viene raffigurato da una retta, sottolineando la non variazione del parametro nel tempo. Nella versione con versamenti mensili ricorrenti, la curva del capitale versato aumenta di anno in anno. Nell'implementazione del grafico, Claude si è dimostrato più che preparato, adeguando il codice precedentemente generato all'aggiunta di nuove importazioni fondamentali per soddisfare la richiesta e modellando adeguatamente il grafico a seconda dei parametri in

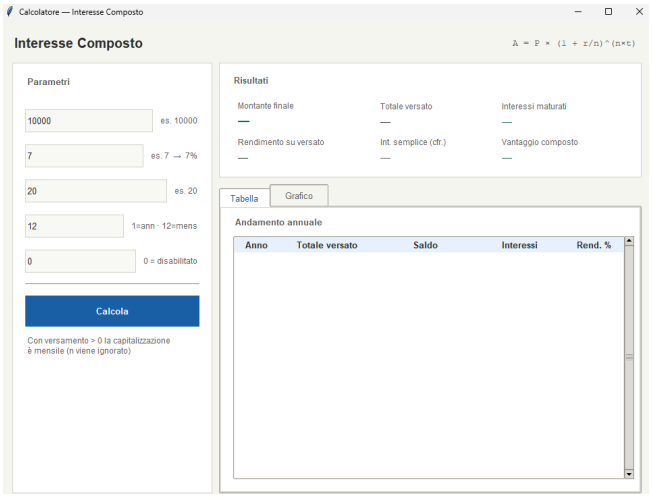

Figura 5.8: Seconda interfaccia grafica

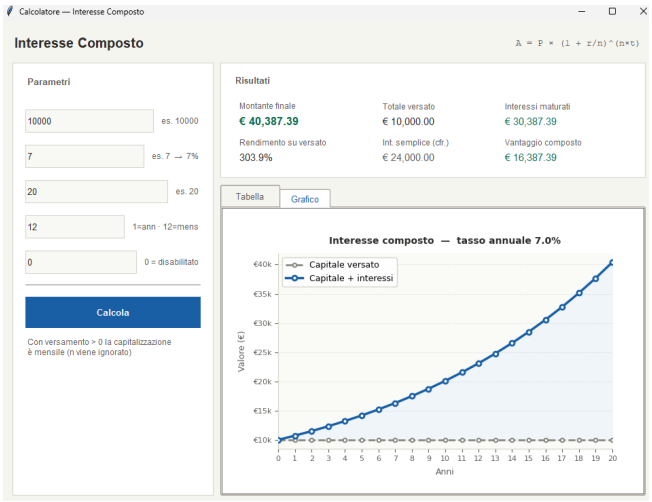

Figura 5.9: Risultato della seconda interfaccia grafica senza versamenti

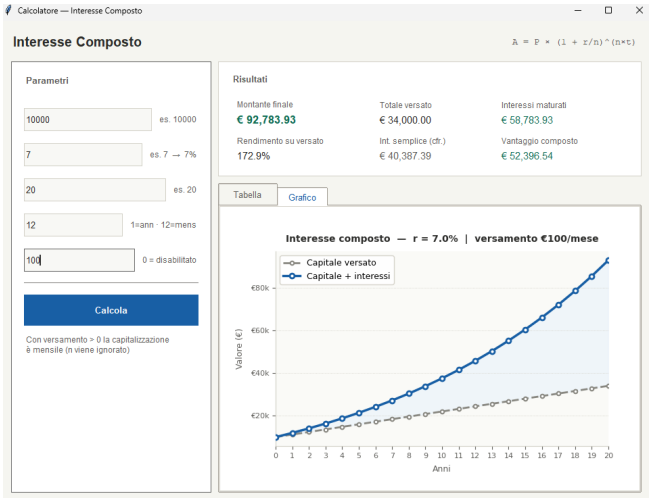

Figura 5.10: Risultato della seconda interfaccia grafica con versamenti

input. Il codice generato, quindi, oltre che essere funzionale da un punto di vista matematico,

è anche pratico da un punto di vista funzionale.

5.6 Il testing del proprio algoritmo e il risultato finale

Al fine di consegnare un buon prodotto è fondamentale testare adeguatamente il codice con l'obiettivo di evidenziare possibili limiti, mancanze o, addirittura, errori di programmazione e non consegnare, quindi, un prodotto incompleto e/o difettoso. Il ruolo del tester è spesso trascurato. Nel mondo della programmazione si ricade molto spesso in scadenze strette e rigide. Per questo motivo, un adeguato testing viene spesso omesso per via delle sue lunghe tempistiche, privilegiando, quindi, una consegna più rapida con il rischio che il prodotto consegnato risulti non del tutto adeguato all'uso di chi richiede il servizio. La tesi in questa fase ha come scopo verificare la competenza di Claude anche nel ruolo di tester professionista. A tal fine il modello dovrà analizzare il codice prodotto e progettare una serie di input estremi in modo da rilevare comportamenti inattesi.

Il prompt per il testing dell'algoritmo è il seguente:

Utente

Sei un tester esperto. Analizza l'algoritmo che hai generato e progetta una serie di casi di input estremi e borderline per metterlo alla prova. Per ogni caso spiega perché è problematico e cosa dovrebbe fare correttamente il programma.

Claude progetta in totale 32 test borderline e, in fine, rileva 4 *bug*, Il resoconto dei bug generato da Claude è il seguente:

Claude

- BUG-1 Grafico con t < 1 anno: un solo punto, nessuna linea

  Perché è problematico:Con t=0.5 anni lo storico annuale ha 1 sola riga (anno 0). Matplotlib plotta 1 punto per linea. Non si disegnano linee, la legenda compare ma le curve sono invisibili. Nessun errore ma esperienza utente rotta.

  Comportamento atteso:Il grafico mostra almeno due punti (inizio e fine periodo)

- BUG-2 "12.5" come valore di n: errore di parsing non intuitivo

  Perché è problematico:int("12.5") solleva ValueError. Il messaggio d'errore generico "invalid literal for int()" non spiega all'utente cosa ha sbagliato. Un utente potrebbe ragionevolmente inserire "12.5" pensando a una capitalizzazione ogni 10 giorni.

  Comportamento atteso:Messaggio chiaro: "n deve essere un numero intero (es. 1, 4, 12, 365)"

- BUG-3 "10,000" come P: virgola interpretata come separatore decimale

  Perché è problematico:.replace(",", ".") trasforma "10,000" in "10.000" → float("10.000") = 10.0, non 10000.0. Un utente italiano che usa la virgola come separatore delle migliaia otterrà P=10 silenziosamente.

  Comportamento atteso:P = 10000.0, oppure errore esplicito

- BUG-4 versamento ="inf":accettato silenziosamento

  Perché è problematico:float("inf") = math.inf in Python. Il programma accetta inf, lo passa a `calcola_con_versamenti`, che esegue saldo += inf → saldo = inf per tutti i mesi. Il risultato finale è inf. Nessun crash ma il programma mostra "€ inf" nella UI.

  Comportamento atteso:Rifiutato con messaggio "Inserisci un valore numerico finito"

A questo punto, non resta che verificare se effettivamente gli errori riscontrati esistono in modo da riportare se il modello è in grado di testare in maniera autonoma il proprio codice

oppure commette allucinazioni nel tentativo di generare un resoconto completo. La lista dei casi da verificare riportati da Claude è la seguente:

- *Grafico con $t < 1$*: Claude segnala che, a quanto pare, inserendo un valore inferiore a 1 nella sezione dedicata alla durata in anni, Matplotlib tenta di disegnare una linea su un singolo punto. La verifica dell'errore viene riportata nella Figura 5.11.

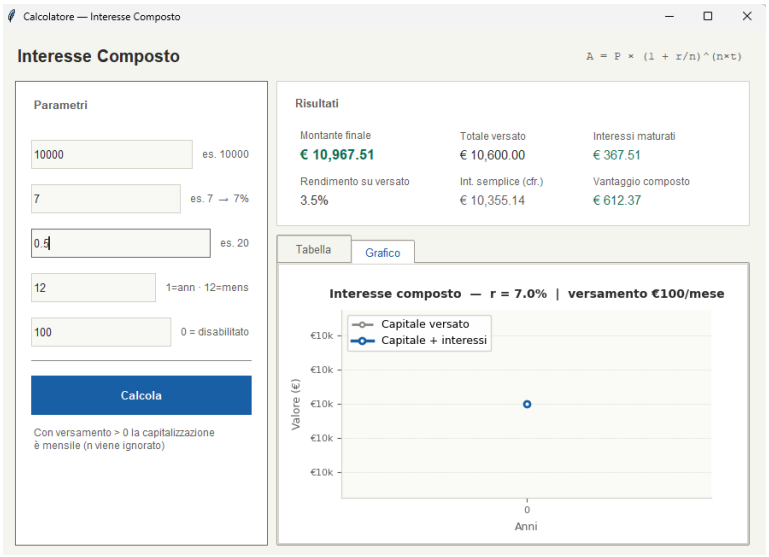

Figura 5.11: Output relativo al bug 1

Claude rileva correttamente l'errore, l'ascissa contiene solo l'anno 0 e il programma non aggiunge un punto finale.

- *"n = "12.5" genera un errore incomprensibile*: il programma quindi, a quanto pare, non è in grado di accettare il parametro n come decimale e, inoltre, non segnala adeguatamente l'errore rischiando di indirizzare male l'utente. La verifica del bug viene riportata nella Figura 5.12.

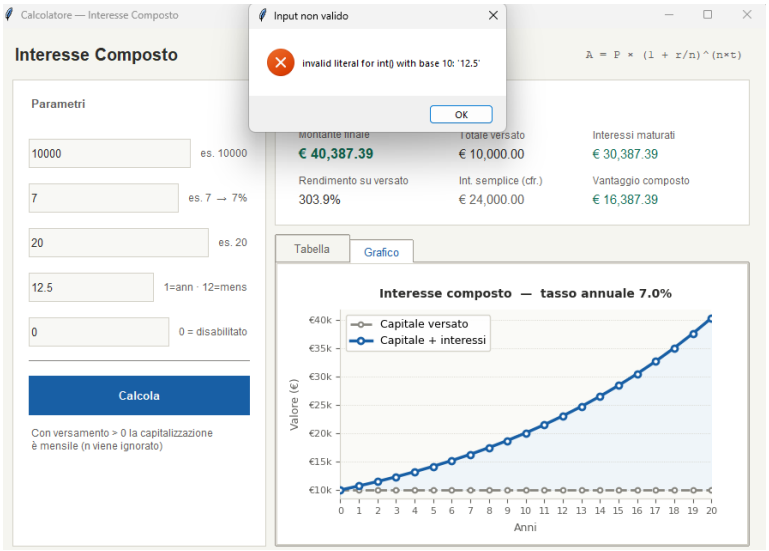

Figura 5.12: Output relativo al bug 2

Il messaggio di errore risulta effettivamente poco intuitivo, Il programma non mostra un output chiaro ma segnalando soltanto che il valore risulta non valido.

- *"10,000" come P: virgola interpretata come separatore decimale*: Claude, dunque, vuole assicurarsi che anche un utente proveniente da una nazione che usa la virgola come separatore delle migliaia possa avere il proprio input interpretato in maniera corretta. La verifica del bug viene riportata nella Figura 5.13.

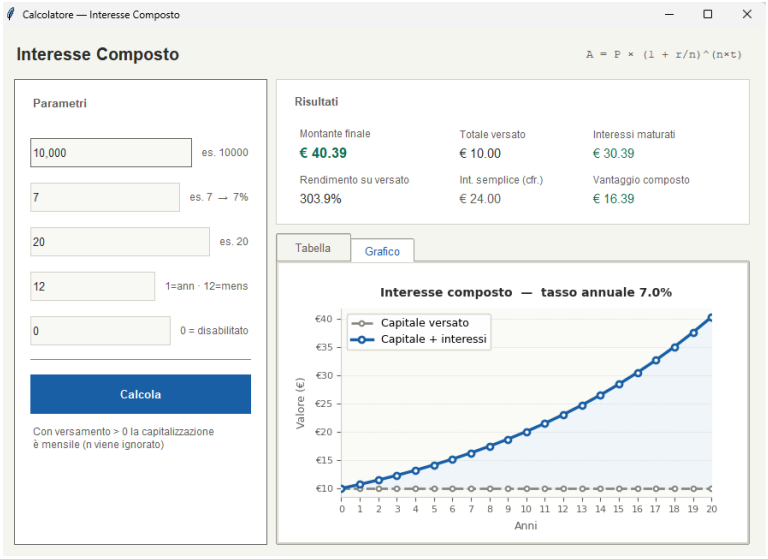

Figura 5.13: Output relativo al bug 3

Il programma interpreta "10,000" come "10". Claude rileva correttamente l'errore.

- *versamento = "inf" accettato*: il programma, dunque, accetterebbe come parametro un numero infinito, un errore problematico dal punto di vista matematico. La verifica dell'errore viene riportata nella Figura 5.14.

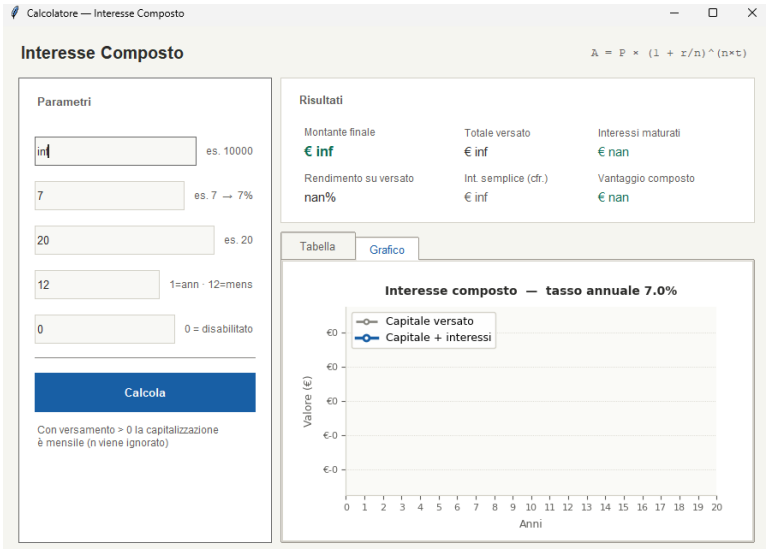

Figura 5.14: Output relativo al bug 4

Il programma accetta silenziosamente il valore di infinito, non mostrando messaggi di errore ma, semplicemente, rendendo il lavoro dell'interfaccia grafica inutile. Claude riporta correttamente l'errore anche in questo caso.

5.6.1 Il programma completo

Claude ha dimostrato di essere ampiamente capace, non solo di generare in pochi minuti programmi complessi e funzionali, ma anche di ricoprire il ruolo di tester professionista, rilevando correttamente 4 errori di programmazione e generando un resoconto ben definito sulla problematica e il comportamento atteso, proprio come richiesto.

Oltre a essere in grado di rilevare questi errori, il modello sarà in grado di correggerli e generare il programma completo e privo di bug? La tesi in questa ultima fase del case study ha l'obiettivo di rispondere a questo quesito.

L'ultimo prompt è il seguente:

Utente

Riscrivi l'algoritmo correggendo i 4 bug che hai trovato

Qui di seguito, vengono riportate le casistiche precedenti in modo da verificare se il programma finale generato da Claude ha, effettivamente, corretto i bug riscontrati.

- *Grafico con $t < 1$*: precedentemente veniva riportato un solo punto nel grafico invece che una curva; l'output della versione finale rispetto a questo caso è riportato nella Figura 5.15.

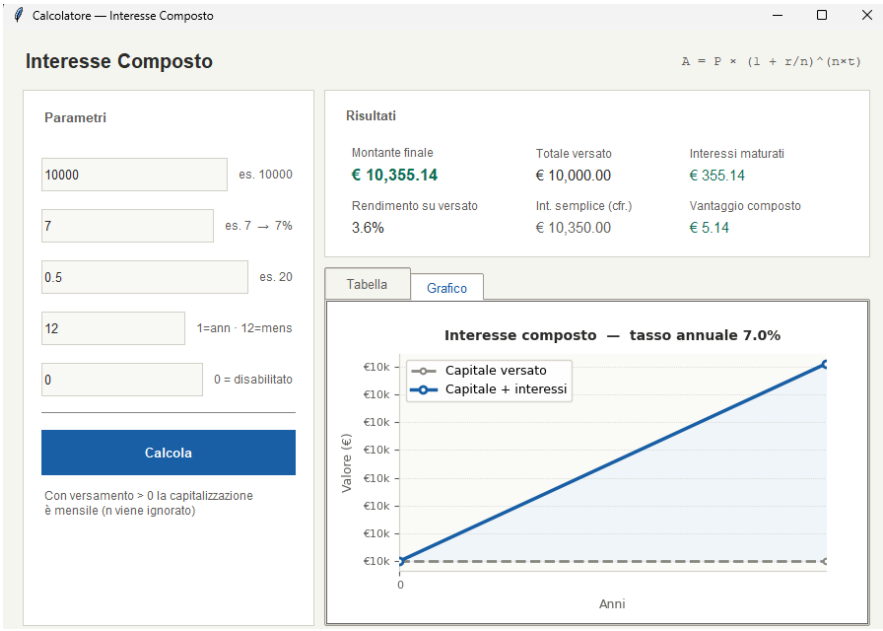

Figura 5.15: Output della versione finale del primo caso

La correzione consiste nell'aggiungere sempre un valore finale pari al saldo dell'ultimo anno frazionato disponibile. Claude corregge in maniera efficace l'errore precedentemente riscontrato.

- *"n = "12.5" genera un errore incomprensibile*: il programma, nella versione precedente, non era in grado di ricevere in input una frequenza di capitalizzazione non intera e generava un messaggio di errore non efficace.

  La Figura 5.16 riporta l'output della versione finale del programma rispetto a questo caso.

  Il programma è ora in grado di ricevere come parametro della frequenza di capitalizzazione un parametro non intero, convertendo silenziosamente il valore decimale da float

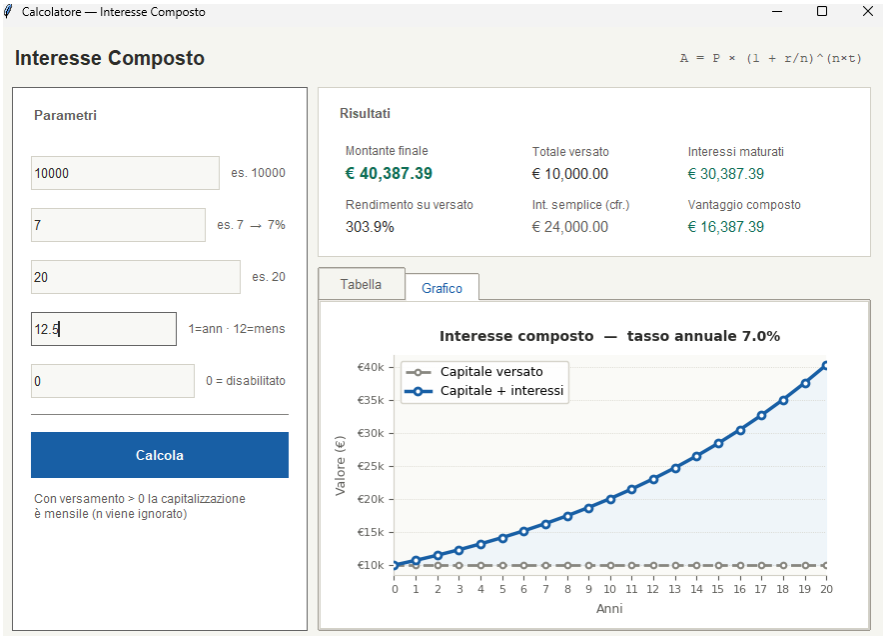

Figura 5.16: Output della versione finale del secondo caso

a intero usando l'istruzione `int(float(n_str))`. Claude corregge, anche in questo caso, la problematica riscontrata in precedenza.

- *"10,000" come P: virgola interpretata come separatore decimale*: Claude reputa questo comportamento come indesiderato, la correzione consiste nel creare una funzione `parse_numero(testo)` che adoperi l'istruzione `re.search(r',\d{3}', testo)` in modo da rilevare il pattern che separa la virgola da esattamente 3 cifre; se presente, la virgola è un separatore delle migliaia e viene rimossa, altrimenti la virgola assume il ruolo di separatore decimale e viene sostituita con il punto.

  La verifica dell'output della versione finale del programma rispetto a questo caso viene riportata nella Figura 5.17.

  Il programma, adesso, distingue il separatore delle migliaia dal separatore decimale; un altro successo per Claude.

- *versamento = "inf" accettato*: il programma, nella versione precedente, accettava in input il volore di "inf", essendo tale valore valido in Pyhton. La correzione consiste nel chiamare, dopo ogni conversione, `math.isfinite(valore)`. Qualora la funzione debba restituire valore `False`, viene sollevato un `ValueError` con messaggio chiaro.

  La Figura 5.18 riporta l'output della versione finale del programma rispetto a questo caso.

  Il programma stampa sul video un messaggio chiaro se il valore di P è "inf". Claude dimostra, ancora una volta, di essere preciso e coerente nelle sue correzioni.

5.7 Conclusione case study relativo alla generazione di codice

Il case study condotto sul calcolo dell'interesse composto ha permesso di osservare le notevoli capacità di Claude in ogni fase dello sviluppo software: dalla progettazione dell'algoritmo inziale, passando per l'implementazione dell'interfaccia grafica e concludendo con un testing approfondito del proprio algoritmo. Questo risultato apre uno spunto di

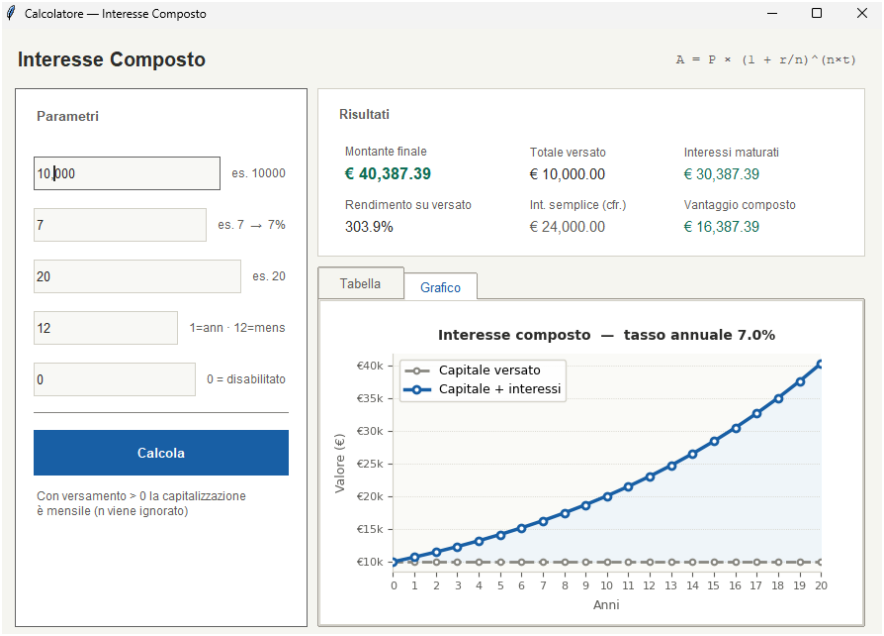

Figura 5.17: Output della versione finale del terzo caso

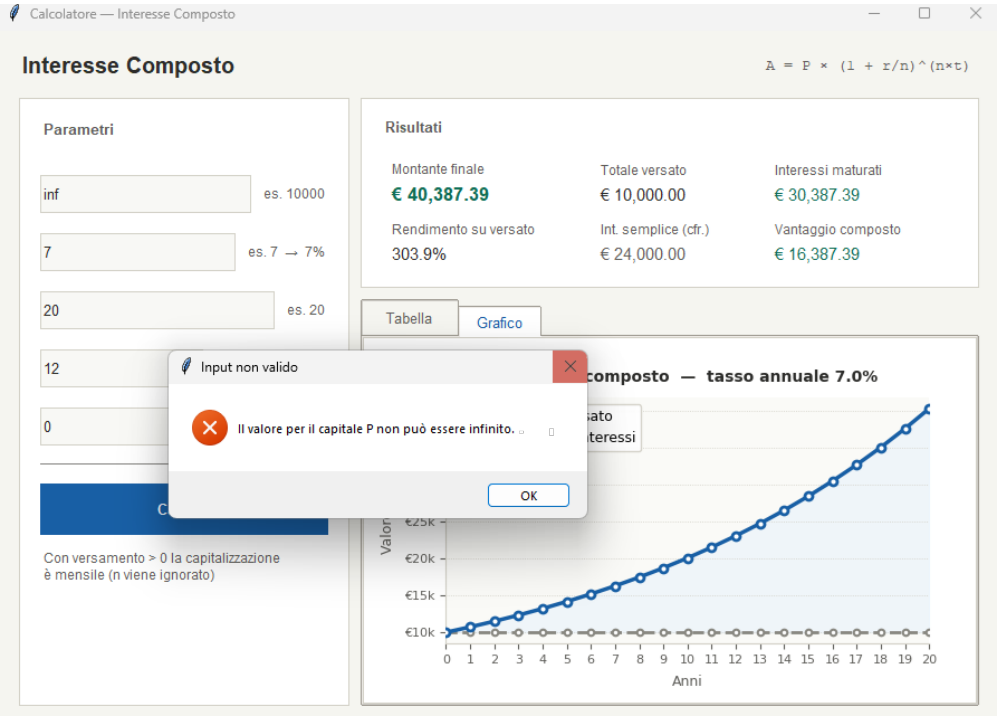

Figura 5.18: Output finale del quarto caso

riflessione che va oltre all'ambito puramente tecnico dello studio appena riportato. Infatti, se un modello è in grado di generare, in pochi minuti, un programma complesso funzionante e di testare in maniera autonoma il proprio algoritmo risolvendo successivamente ogni singola anomalia riscontrata, esisterà mai un futuro nel settore dello sviluppo software? La tesi non ha lo scopo di rispondere a questa domanda; determinati quesiti meriterebbero elaborati apparte. Tuttavia, resta da sottolineare che quanto compiuto da Claude durante lo sviluppo è stato possibile solo tramite l'orchestrazione delle domande a cui è stato sottoposto. Il valore aggiunto del giudizio umane rimane, ad oggi, imprescindibile. Pertanto, riteniamo opportuno

sottolineare, ancora una volta, quanto la capacità di formulare i prompt giusti rimanga una competenza con notevoli potenzialità in un mondo in cui lo sviluppo software privo di integrazione di strumenti di Intelligenza Artificiale Generativa è una disciplina morente.

CAPITOLO 6

Discussione

In questo capitolo verranno riepilogati i case study appena svolti sulla generazione di testo, immagini e codice. Il capitolo introdurrà all'inizio la discussione evidenziando l'obiettivo principale per cui si è svolto lo studio riportato. Successivamente, il capitolo discuterà i risultati ottenuti da ogni case study, riportando le differenze sorte durante il confronto tra i modelli presi in esame ed evidenziando l'aderenza tecnica alle richieste e le eventuali criticità sorte durante l'interazione.

6.1 Introduzione alla discussione

Nei capitoli precedenti, i tre distinti case study hanno sottoposto, tramite una serie di prompt, i modelli presi in esame a un'interazione volta a mettere alla prova le loro capacità generative. Lo studio e il confronto ha, infatti, messo in evidenza le rispettive potenzialità e limitazioni, tramite un'interazione studiata appositamente per testare i punti critici più documentati nelle proprie regioni. Verranno discussi i risultati ottenuti dai tre ambiti dell'Intelligenza Artificiale Generativa esaminati, valutando i modelli all'interno del dominio di generazione specifico.

6.2 Discussione sul case study relativo alla generazione di testo

Il case study svolto riguardo alla generazione di testi ha messo a confronto, su questa branca dell'Intelligenza Artificiale Generativa, ChatGPT e DeepSeek. I risultati ottenuti hanno permesso di delineare un quadro comparativo tra i due modelli su due aspetti: quello tecnico e quello relativo ai contenuti.

Dal punto di vista tecnico, i due modelli sono stati sottoposti, durante la generazione di testi, a una serie di vincoli quantitativi rigidi; nel primo prompt il vincolo riguardava esclusivamente il numero di parole generato; nel secondo, invece, i modelli dovevano, oltre a generare un piano alimentare settimanale, conteggiare adeguatamente le calorie di ogni alimento riportato nel loro rispettivo piano dietetico al fine di rispettare il vincolo numerico delle calorie imposto dalla richiesta. L'approccio dei due modelli è stato, sin dall'inizio, molto diverso. Nel conteggio delle parole ChatGPT si è dimostrato più adeguato, sacrificando, però, la sintassi dell'articolo generato. DeepSeek, invece, non ha rispettato il vincolo numerico mantenendo, però, un approccio più scientifico, con una sintassi a tratti migliore. Nella stesura del piano alimentare settimanale, le differenze sono ancora più marcate: ChatGPT

tenta un approccio estremamente professionale, generando un piano settimanale completo con riportato l'elenco degli alimenti per ogni giorno della settimana. DeepSeek, invece, opta per un approccio più "sicuro"; il modello, infatti, non genera un piano completo ma un prototipo di "giornata tipo", approccio che, all'occhio dell'utente, è meno utilizzabile, in quanto le variazioni settimanali rispetto al prototipo non sono state riportate dal modello. Da un punto di vista matematico, però, l'approccio di DeepSeek risulta essere più accurato rispetto ai vincoli imposti dall'utente e le premesse riportate dal modello; il piano, infatti riporta il quantitativo calorico per ogni pasto in maniera precisa e matematicamente giusta. L'approccio professionale di ChatGPT, d'altronde, dimostra di non ripagare dal punto di vista dell'accuratezza dei calcoli; il modello, infatti, pur riportando l'elenco ben definito degli alimenti, fornisce un piano con taglio delle calorie troppo aggressivo rispetto alle premesse e rispetto al vincolo numerico imposto dalla richiesta, mostrando evidenti difficoltà, durante la generazione, nell'accostamento tra l'alimento riportato, il quantitativo in grammi e l'apporto calorico. Dal punto di vista tecnico, quindi, non esiste una scelta migliore secondo vincoli oggettivi; dunque la scelta tra i due modelli, su questo tipo di compiti, dipende dal criterio che l'utente considera come prioritario, più che da una superiorità tecnica oggettiva.

Dal punto di vista dei contenuti, i due modelli sono stati sottoposti ad una serie di prompt politicamente "scomodi". Il quadro che emerge è nettamente asimmetrico, non solo per via della censura persistente nelle generazioni di DeepSeek (ipotesi, comunque, prevedibile vista la provenienza geopolitica del modello), quanto per i meccanismi stessi tramite cui la censura viene attivata. Il modello, infatti, non ha riscontrato problemi nel generare articoli che mettessero la Cina in cattiva luce, elencandola addirittura tra i paesi che adoperano ancora la pena di morte, specificando persino che questa pratica viene usata, più che come mezzo di giustizia, come mezzo di repressione, con un onestà intellettuale pari a quella di ChatGPT. Il blocco netto è sorto, invece, qualora si parlasse di sorveglianza stretta sui cittadini e sulle proteste di Hong Kong; l'ipotesi più plausibile è che la censura sia mirata a non diffondere contenuti che toccano direttamente il controllo interno sulla popolazione e la legittimità politica del partito di governo, piuttosto che sul non diffondere critiche generiche su mezzi politici che vengono applicati.

In conclusione, riguardo ai contenuti generati dai modelli, ChatGPT dimostra un'apertura conforme alla libertà di espressione presente nelle democrazie occidentali; DeepSeek, nonostante il fatto che in più occasioni è stato in grado di riportare argomenti critici verso il proprio paese di origine, ha dimostrato di essere sottoposto a un meccanismo di filtraggio selettivo che si attiva in base a parole e contenuti direttamente sensibili alla legittimità del sistema politico cinese.

6.3 Discussione case study relativo alla generazione di immagini

Il case study relativo alla generazione di immagini ha restituito un quadro comparativo alquanto polarizzato.

Il prompt scelto come punto di partenza, ovvero la generazione di tutte le 52 carte di un mazzo francese classico, si è rilevato un banco di prova particolarmente efficace. Per poter compiere al meglio la richiesta, i tre modelli dovevano essere in grado di arginare le difficoltà più documentate relative alla generazione di immagini, gestendo, simultaneamente, tre livelli di complessità: la riproduzione corretta dei numeri e delle lettere, la replica accurata delle figure associate e la coerenza nel posizionamento. Data la natura probabilistica di questo tipo di generazioni, questa combinazione risulta particolarmente insidiosa, dal momento che sfida i modelli in ambiti dove la generazione di immagini risulta spesso poco efficiente. ChatGPT è riuscito a compiere al meglio la prima richiesta, riproducendo alla perfezione ogni singola carta, mantenendo le giuste proporzioni e il posizionamento di numeri, lettere e figure.

Gemini ha ottenuto un risultato che, complessivamente, può dirsi sufficiente; producendo un risultato conforme alla richiesta ma commettendo qualche sbavatura su alcune figure. Grok, al contrario, fin dal primo prompt ha dimostrato di non essere in grado di soddisfare il grado di difficoltà della richiesta, commettendo allucinazioni di ogni tipo e non riproducendo nessuna carta nel modo corretto, sia nella forma che nel posizionamento.

La sequenza dei prompt successivi ha permesso di approfondire ancora meglio le potenzialità relative alla generazione di immagini di questi modelli: la capacità nel mantenere la coerenza spaziale tra una generazione e la successiva è stata una prova che ha permesso di approfondire i loro pregi e le loro limitazioni.

La richiesta di generare la stessa immagine ma con tutti i fanti scartati richiedeva non solo di identificare adeguatamente le carte da eliminare, ma anche di riprodurre correttamente l'immagine precedente con le modifiche richieste. ChatGPT ha escluso correttamente i fanti ma ha commesso qualche errore nella colonna delle regine, Gemini ha eliminato per errore queste ultime sovrapponendone alcune nella colonna dei re; Grok, invece, ha rimosso carte senza una logica decifrabile, producendo un risultato ancora più incomprensibile rispetto alla sua generazione precedente.

I prompt successivi, hanno introdotto la generazione di mani e l'impugnatura delle carte, alzando ulteriormente il livello di difficoltà, toccando un aspetto della generazione di immagini critico per molti modelli, ovvero, la generazione di mani e dita. I modelli, come risultato finale, dovevano riprodurre le due mani di un croupier in cui, in una mano, veniva impugnata la prima riga di carte e, nell'altra mano, venivano impugnati i fanti scartati, con il vincolo, non banale, di raffigurarli a faccia in giù. ChatGPT ha prodotto i risultati migliori; raffigurando correttamente entrambe le mani, generando quasi alla perfezione l'impugnatura delle carte e rispettando il vincolo di mostrare solo il dorso dei 4 fanti scartati. Gemini ha generato un'immagine con mano corretta ma contesto errato, raffigurando un mazzo errato e un secondo individuo non richiesto che impugna i fanti scartati. Grok ha accumulato errori, riproducendo due mani sinistre e fanti rivolti di fronte anzichè di dorso.

Nel complesso, il case study conferma che la generazione di immagini rimane uno degli ambiti dell'Intelligenza Artificiale Generativa in cui i modelli attualmente disponibili riscontrano maggiori difficoltà; infatti, anche il modello che, durante lo studio, ha registrato i risultati nettamente migliori, non è stato in grado di riprodurre alla perfezione le immagini richieste commettendo sbavature minori, sottolineando la difficoltà in questo tipo di generazione. Gemini ha dimostrato diversi limiti, evidenziando, comunque, una solidità di base con ampi margini di miglioramento. Grok ha rilevato, invece, limitazioni strutturali che rendono il modello totalmente inaffidabile anche su richieste di media complessità.

6.4 Discussione case study relativo alla generazione di codice

Il case study relativo alla generazione di codice si distingue dai due studi precedenti: anziché confrontare più modelli sullo stesso compito, si è deciso di approfondire le capacità di un singolo modello, Claude, lungo un percorso di sviluppo con il fine di creare un programma calcolatore complesso. Il percorso è stato avviato definendo le parti più semplici del programma, ovvero lo scheletro dell'algoritmo; infatti, nel primo prompt, viene richiesto semplicemente di implementare la formula dell'interesse composto in Python. Nel prompt successivo, il grado di difficoltà è aumentato, richiedendo di sviluppare lo stesso programma ma con la possibilità di accettare da terminale i versamenti mensili fissi (con le variazioni nella formula che ne derivano). In questi primi prompt, Claude ha dimostrato una capacità notevole nello strutturare le sezioni di codice: il codice generato non si è limitato a soddisfare il requisito richiesto, bensì ha adottato fin da subito pratiche per una buona programmazione, separando la logica di calcolo da quella che accetta i parametri in input. Queste pratiche non

sono state espressamente richieste dal prompt; il fatto che Claude le abbia adottate indica che il modello non si limita a soddisfare la richiesta, ma produce codice strutturato come uno sviluppatore esperto.

La richiesta di implementare un'interfaccia grafica con Tkinter ha rappresentato il salto di qualità più significativo dell'intero case study. Claude approccia la richiesta strutturando il codice seguendo il pattern della separazione delle responsabilità, organizzando il codice in metodi e classi con compiti ben distinti. La logica di presentazione, infatti, è stata definita in una classe autonoma e ogni responsabilità nei calcoli è stata delegata a metodi ben distinti, producendo un codice leggibile e manutenibile. Particolarmente significativa è stata la gestione della richiesta che comportava l'integrazione di un grafico con Matplotlib all'interno dell'interfaccia: Claude ha rilevato immediatamente, prima della generazione del codice, un problema di compatibilità tra il backend di Matplotlib e la finestra Tkinter. Il modello risponde con precisione chirurgica, definendo importazioni che, oltre a implementare le librerie per il disegno del grafico, introducevano librerie intermediarie tra Matplotlib e Tkinter. La scelta di anticipare un problema architetturale non espresso dalla richiesta rappresenta un elemento molto importante per l'analisi.

La fase di testing ha offerto i risultati forse più inattesi dell'intero case study; a Claude è stato richiesto di ricoprire il ruolo di tester esperto e di identificare comportamenti inattesi all'interno del programma attraverso una serie di casi borderline, restituendo la spiegazione del motivo per cui il comportamento fosse problematico e quale sarebbe dovuto essere il comportamento atteso. Claude, come risposta, ha progettato 32 casi di test borderline e ha identificato quattro bug nel proprio codice, fornendo una descrizione adeguata della problematica e del comportamento atteso, proprio come richiesto. Il modello, quindi, dimostra non solo una capacità notevole nell'analisi critica del proprio output, ma anche di rispettare gli standard di una figura tecnica specializzata, strutturando un bug report professionale e accurato.

L'ultimo prompt dello studio richiedeva la correzione dei quattro bug identificati, al fine di chiudere il ciclo di sviluppo in modo completo. Ciascuna correzione è risultata precisa, mirata e priva di effetti a cascata sul resto del codice. Il programma finale risulta robusto, completo e privo di errori, anche in casi di input estremi.

Il case study per lo sviluppo di codice restituisce un quadro chiaro: Claude è in grado di affrontare in modo autonomo e professionale le fasi di progettazione, implementazione, estensione e testing. Questa affermazione introduce una riflessione che va oltre il piano puramente tecnico dello studio appena riportato: se un modello è in grado di generare un codice complesso e di testarlo in autonomia il valore umano non può più risiedere nella capacità di risolvere un problema per intero, quanto nella capacità di definire il problema nel modo giusto.

Conclusioni

Nel corso di questa tesi abbiamo esaminato il mondo dell'Intelligenza Artificiale generativa sotto molteplici aspetti, muovendoci dalla teoria alla pratica attraverso un percorso strutturato in due parti. La prima parte, di carattere introduttivo, è servita per definire i fondamenti teorici dell'Intelligenza Artificiale Generativa e le sue diverse classificazioni, dall'architettura ai tipi di output generato fino alle modalità di accesso, fornendo al lettore una mappa riepilogativa del panorama attuale. Nella seconda parte si è concentrato lo studio sulle capacità generative dei modelli presi in esame. Nel primo studio, riguardante la generazione di testo, i modelli sono stati sottoposti prima a vincoli quantitativi rigidi, col fine di verificare quale modello fosse più affidabile nel rispetto di questi vincoli, tramite richieste riguardanti il conteggio delle parole e la stesura di un piano alimentare, e, successivamente, a prompt politicamente scomodi, con particolare attenzione ai meccanismi di censura riscontrati nel modello di origine cinese. Il secondo studio ha confrontato, sulla generazione di immagini, tre modelli diversi, rilevando differenze strutturali significative nella gestione della coerenza spaziale e nella continuità contestuale delle immagini generate. Il terzo studio ha osservato Claude lungo una bozza di ciclo di sviluppo software, chiedendo al modello di sviluppare un programma Python per il calcolo dell'interesse composto, documentando le capacità notevoli del modello nello strutturare e testare autonomamente il codice generato.

I tre case study, nel loro insieme, hanno offerto una visione concreta sulle potenzialità e sulle limitazioni dei modelli di Intelligenza Artificiale Generativa al momento della stesura di questo lavoro. Guardando ai possibili sviluppi futuri, emergono alcune direzioni di ricerca su cui vale la pena porre l'attenzione. Un naturale proseguimento della presente tesi consisterebbe nell'estendere il confronto anche nella dimensione temporale, confrontando i modelli appena sottoposti allo studio a distanza di mesi e/o anni, documentando l'evoluzione delle capacità generative e verificando quali dei limiti riscontrati in alcuni modelli persistono nelle versioni future. Un ulteriore sviluppo riguarda la questione della censura: l'indagine riportata nella tesi ha evidenziato come DeepSeek è sottoposto a meccanismi che filtrano le informazioni che il modello può divulgare; un'ulteriore studio potrebbe condurre un'indagine sul modello open-source di DeepSeek, eseguito in locale e privo di meccanismi di filtraggio delle risposte che l'interfaccia ufficiale sottopone selettivamente al modello, rendendo l'analisi più rigorosa. Infine, riguardo alla generazione di codice, uno sguardo al futuro suggerisce un'apertura di scenari in cui il ruolo del prompt engineer è sempre più centrale, spostando il valore umano dalla scrittura del codice alla definizione precisa delle problematiche, del contesto e delle metodologie che il modello deve applicare nel corso della generazione. È in questa direzione che, a nostro avviso, si collocherà il contributo dell'essere umano nello sviluppo del software nei prossimi anni.

Bibliografia

BAI, Y. (2022), «Training a helpful and harmless assistant with reinforcement learning from human feedback», *Anthropic*.

BENDER, E. M. (2021), «On the dangers of stochastic parrots: Can language models be too big?», in «Proceedings of the 2021 ACM conference on fairness, accountability, and transparency», .

BOND-TAYLOR, S. (2021), «Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models.», *IEEE transactions on pattern analysis and machine intelligence*.

DHARIWAL, P. e NICHOL, A. (2021), «Diffusion models beat gans on image synthesis», in «Advances in neural information processing systems, 34», .

HO, A. J., JONATHAN e ABBEEL, P. (2020), «Denoising Diffusion Probabilistic Models.», in «Advances in neural information processing systems, 53», .

JACOB AUSTIN, A. O. (2021), «Program Synthesis with Large Language Models», *Google Research*.

JIMENEZ, C. E. (2024), «Swe-bench: Can language models resolve real-world github issues?», in «International Conference on Learning Representations. Vol. 2024», .

LIU, A. (2024), «Deepseek-v3 technical report», Rap. tecn., arXiv preprint.

ROUMELIOTIS, K. I. e TSELIKAS, N. D. (2023), «ChatGPT and Open-AI Models: A Preliminary Review», *Future Internet*.

RUSSELL, S. J. e NORVIG, P. (1995), *Artificial Intelligence - A Modern Approach*, Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs.

SOLAIMAN, I. (2019), «Release strategies and the social impacts of language models», Rap. tecn., OpenAI.

TEAM, G. (2023), «Gemini: a family of highly capable multimodal models.», *Google DeepMind*.

TOM B. BROWN, N. R. M. S. J. K., BENJAMIN MANN (2020), «Language Models are Few-Shot Learners», in «Advances in neural information processing systems, 33», .

WEI, J. (2022), «Chain-of-thought prompting elicits reasoning in large language models», in «Advances in neural information processing systems, 35», .

WHITE, J. (2023), «A prompt pattern catalog to enhance prompt engineering with chatgp», Rap. tecn., Department of Computer Science Vanderbilt University, Tennessee.

ZHAO, W. X. (2026), «A survey of large language models», *Frontiers of Computer Science*.

Sitografia

- AIBlog – https://iartificial.blog/it/apprendimento
- ArXiv – www.arxiv.org
- Claude – www.claude.ai
- DeepSeek – www.deepseek.com
- Gemini – https://gemini.google.com/app?hl=it
- GitHub – www.github.com
- Grok – www.grok.com
- IBM, Che cos'è il prompt engineering? – https://www.ibm.com/it-it/think/topics/prompt-engineering
- IBM, Cos'è un modello trasformatore? – https://www.ibm.com/it-it/think/topics/transformer-model
- IBM, What is Artificial Intelligence? – https://www.ibm.com/think/topics/artificial-intelligence
- Khan Academy, Intro to compound interest – www.khanacademy.org
- McKinsey & Company, What is generative AI? – https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-generative-ai
- Microsoft – www.microsoft.com
- OpenAI – www.openai.com
- Parlamento Europeo, Cos'è l'intelligenza artificiale? – https://www.europarl.europa.eu/topics/it/article/20200827STO85804
- Python – www.python.org
- Wikipedia – www.wikipedia.org

Ringraziamenti

La data odierna è *2 Luglio 2026*, ce ne ho messo di tempo...

Sembra che gli sforzi che ho fatto, alla fine, siano stati ripagati.

Alcune volte è sembrato che remasse tutto contro: lo studio, il lavoro, gli amici, la salute. Non sempre è stato facile.

La vita alcune volte non va nei binari desiderati; molto spesso la mente gioca brutti scherzi, sembra quasi che, in determinati momenti, l'opzione migliore sia lasciare perdere tutto quanto. Gli sforzi già compiuti nel percorso di studi? I colleghi compagni di mille battaglie? Le amicizie che hai coltivato nel corso degli anni? Tutto da buttare...

Devo ringraziare me stesso, per non essere stato di questo parere, per aver dimostrato che, per ottenere qualsiasi cosa, *qualsiasi* cosa, l'importante è convincersi di potercela fare.

Un ringraziamento speciale va a voi, amici miei. Sia che vi abbia conosciuto prima di iniziare questo percorso, sia che vi abbia conosciuto nel corso del triennio, vi ringrazio.

Con voi ho passato dei periodi che mi ricorderò per tutta la vita. Voi, che mi avete saputo ascoltare e consigliare; il minimo che io possa fare è ringraziarvi.

Un ringraziamento di cuore va a mia madre. Hai sacrificato tantissimo per permettermi di portare al termine questo percorso; è stato brutto vederti lavorare tanto; ho cercato di contribuire in tutti i modi, ma lo sforzo che hai fatto è impagabile.

Un ringraziamento speciale va al professore Domenico Ursino, per la sua grande disponibilità con cui mi ha assistito nelle fasi finali di questo percorso.

Ringrazio te, Nicolò. Ce ne siamo dette di tutti i colori nel corso degli anni, diciamoci la verità... La fortuna ha voluto che, nel momento del bisogno, entrambi abbiamo sempre risposto "presente". Non è scontato, è unico.

Vorrei riportare i possibili sviluppi futuri anche per questa parte della tesi, ma la sfera di cristallo purtroppo ancora non l'hanno inventata, quindi mi limiterò nel dire che: ce la metterò tutta.

Non importa quanto un futuro vivibile sembri improbabile, vista la situazione nel mondo attuale. A livello economico, a livello geopolitico, a livello climatico; insomma, è davvero un bel casino.

Sta a noi, che ci crediamo davvero, cambiare le cose.

Meglio rimboccarsi le maniche.