



**Politecnico
di Torino**

**Corso di Laurea Triennale in Ingegneria
Gestionale Classe L-8**

**Sviluppo di un'Applicazione per
l'Analisi delle Interazioni Sociali**
Il Caso Social-Entertainment

Relatore
Prof. Giuseppe Averta

Candidato
Francesco Pusceddu
S294392

Sessione di Laurea Marzo 2025

A. A. 2024/2025

Sommario

Capitolo 1 - Introduzione e Contesto del Progetto	3
1.1 Proposta di Progetto	3
1.1.1 Studente Proponente.....	3
1.1.2 Titolo della proposta	3
1.1.3 Descrizione del Problema Proposto	3
1.1.4 Descrizione della Rilevanza Gestionale del Problema	3
1.1.5 Descrizione dei Data-set per la Valutazione	4
1.1.6 Descrizione Preliminare degli Algoritmi Coinvolti	4
1.1.7 Descrizione Preliminare delle Funzionalità Previste per l'Applicazione Software	4
1.2 Il problema e la motivazione dello studio	5
1.3 Obiettivi e finalità di Social-Entertainment	6
Capitolo 2 – Dataset utilizzato per l’analisi.....	7
2.1 Struttura del dataset.....	7
2.1 Scelte tecniche per la gestione dei dati	9
2.3 Strategie di ottimizzazione delle query	9
Capitolo 3 – Strutture dati ad alto livello ed algoritmi utilizzati.....	13
3.1 Architettura generale del software (Model-View-Controller)	13
3.2 Sviluppo dell’interfaccia grafica con Flet	14
3.4 Implementazione della Ricorsione e Calcolo delle Statistiche	20
3.4.1. La ricorsione come cuore dell’analisi della rete	20
3.4.2 L’impatto degli indici e delle statistiche	26
Capitolo 4 - Analisi delle Prestazioni e Test dell’Applicazione	34
4.1 Risultati sperimentali e performance	34
4.2 Valutazione e conclusioni	41
Riferimenti	44
Licenza	46

Capitolo 1 - Introduzione e Contesto del Progetto

1.1 Proposta di Progetto

1.1.1 Studente Proponente

s294392 Francesco Pusceddu

1.1.2 Titolo della proposta

Social Entertainment: Analisi e Statistiche sugli Utenti nei Social Network

1.1.3 Descrizione del Problema Proposto

L'obiettivo di questo progetto è sviluppare un software in grado di analizzare e filtrare i dati relativi agli utenti di social network, con un focus sulla clusterizzazione e la profilazione degli utenti. Il problema attuale è che i dati spesso non sono facilmente accessibili e richiedono processi di filtraggio e aggregazione manuali, aumentando i tempi di analisi e riducendo l'efficienza nell'estrazione di informazioni utili. Inoltre, in molti casi, un provider di servizi sociali ha la necessità di individuare gli utenti più adatti per testare nuove funzionalità o aggiornamenti della piattaforma. Per farlo, è fondamentale identificare utenti che massimizzano il tempo passato sulla piattaforma e, allo stesso tempo, presentano differenze significative tra loro, garantendo un campione rappresentativo e diversificato per le analisi e i test.

1.1.4 Descrizione della Rilevanza Gestionale del Problema

Dal punto di vista gestionale, questo software consente di ottenere statistiche dettagliate sugli utenti, permettendo una comprensione più approfondita delle loro abitudini. Ciò può essere utile per studi di mercato, ottimizzazione delle strategie di engagement e valutazione delle abitudini di consumo digitale. L'uso di filtri avanzati permette un'analisi mirata e più efficace. Inoltre, la profilazione degli

utenti consente di segmentare i gruppi in cluster omogenei, migliorando la qualità delle analisi e la precisione delle strategie di engagement.

1.1.5 Descrizione dei Data-set per la Valutazione

Il software utilizzerà un dataset open-source, reperito dal sito Kaggle.com, contenente circa 300.000 utenti e informazioni relative alle loro attività su una piattaforma social. Il dataset include dati sul tempo trascorso sulla piattaforma, altre attività svolte dagli utenti e alcuni indici rilevanti come l'isolamento sociale e la qualità del sonno. I dati verranno utilizzati per estrarre informazioni utili all'analisi, garantendo un uso coerente e funzionale nel contesto del progetto.

1.1.6 Descrizione Preliminare degli Algoritmi Coinvolti

Il progetto si baserà principalmente su:

- Ricerca per il provider: Implementazione di un metodo ricorsivo per l'Identificazione dei beta-tester.
- Filtraggio e aggregazione dei dati: I dati verranno filtrati utilizzando dropdown interattivi, limitando il caricamento e migliorando le prestazioni.
- Calcolo di statistiche sugli utenti: Analisi delle medie di tempo dedicato alle varie attività per ottenere insight dettagliati.
- Clusterizzazione degli utenti: Identificazione di gruppi di utenti con comportamenti simili per migliorare le analisi e ottimizzare le strategie di engagement.

1.1.7 Descrizione Preliminare delle Funzionalità Previste per l'Applicazione Software

L'applicazione offrirà un'interfaccia utente basata su Flet con le seguenti funzionalità:

- Dropdown interattivi per la selezione dei filtri (età, genere, abitudini, etc.).
- Ricorsione per l'estrazione dei dati dai provider per ottimizzare il recupero delle informazioni.
- Calcolo di statistiche avanzate sulle attività degli utenti.
- Clusterizzazione automatica per raggruppare gli utenti in categorie omogenee e migliorare le analisi.

- Selezione automatizzata di beta-tester diversificati, garantendo rappresentatività e qualità nei test.
- Esportazione dei risultati delle analisi in formati testuali per una successiva elaborazione.

L'obiettivo finale è fornire uno strumento intuitivo ed efficace per analizzare e filtrare i dati degli utenti, migliorando la comprensione delle loro abitudini, ottimizzando la profilazione e garantendo una selezione efficace degli utenti più adatti a testare le nuove funzionalità del provider, migliorando così la qualità dei risultati ottenuti.

1.2 Il problema e la motivazione dello studio

Negli ultimi anni, l'analisi dei dati provenienti dai social media è diventata un elemento cruciale per aziende di marketing, provider di servizi digitali e piattaforme di comunicazione online. La quantità di informazioni generate quotidianamente dagli utenti è in costante crescita e, di conseguenza, l'abilità di estrarre conoscenze significative da tali dati rappresenta un vantaggio competitivo di primaria importanza.

Uno degli aspetti fondamentali dell'analisi dei social è la “**profilazione degli utenti**”, che consente di individuare schemi comportamentali, interessi comuni e dinamiche di interazione.

Attraverso tecniche avanzate di **clustering** e network analysis, è possibile identificare comunità di utenti (communities) che, pur non conoscendosi direttamente, condividono interessi, abitudini o necessità simili. Questo tipo di analisi è essenziale per:

- Ottimizzare le strategie di marketing e personalizzare le campagne pubblicitarie.
- Monitorare tendenze (trends) ed evoluzioni delle community nel tempo.
- Comprendere meglio le dinamiche di diffusione di contenuti virali e le interazioni tra gli utenti.

Un problema significativo nell'analisi dei social è rappresentato dalla **grande mole di dati da processare**. Il tempo di elaborazione può essere elevato, specialmente se non vengono implementate strategie di ottimizzazione del caricamento e

filtraggio dei dati. L'obiettivo di questo progetto è quindi quello di sviluppare un sistema efficiente che permetta di **ridurre i tempi di elaborazione** pur mantenendo un'elevata qualità dell'analisi.

1.3 Obiettivi e finalità di Social-Entertainment

Il presente progetto propone un **caso di studio** basato su un ipotetico provider di servizi digitali di livello internazionale, con focus su piattaforme social come **Facebook, Instagram, Twitter, YouTube e TikTok**. Il provider è interessato a un'applicazione che permetta l'analisi avanzata dei dati e, in particolare, la selezione di utenti potenzialmente interessati a testare nuove funzionalità della piattaforma (beta-tester).

L'obiettivo principale di **Social-Entertainment** è fornire un framework che:

- Consenta di effettuare analisi avanzate sulle reti sociali, individuando community, gruppi influenti e schemi di comportamento.
- Implementi filtri e strategie di clustering, ottimizzando il processo di selezione degli utenti target.
- Sfrutti un'interfaccia grafica interattiva con “Flet”, che permetta di filtrare dinamicamente i dati per agevolare l'elaborazione e ridurre i tempi di attesa.
- Offra un'analisi efficiente, riducendo l'impatto computazionale grazie a una gestione intelligente del caricamento dei dati nel DBMS.

Attraverso l'applicazione sviluppata, il provider avrà la possibilità di testare nuove funzioni sui gruppi più indicati, senza dover ricorrere a processi lunghi e dispendiosi.

Capitolo 2 – Dataset utilizzato per l'analisi

2.1 Struttura del dataset

L'analisi è basata su un dataset contenente una singola tabella, che raccoglie informazioni dettagliate sugli utenti del sistema.

Il dataset (*Fig. 1*) è composto da 300.000 record, corrispondenti a 300.000 utenti unici, ogni riga della tabella rappresenta un utente, mentre le colonne contengono caratteristiche demografiche, comportamentali e di interazione.

Di seguito, alcuni dei campi più importanti utilizzati nell'analisi:

User_Id (INT, PK):

- Identificativo univoco dell'utente
- Numero crescente a partire da 1

Age (INT):

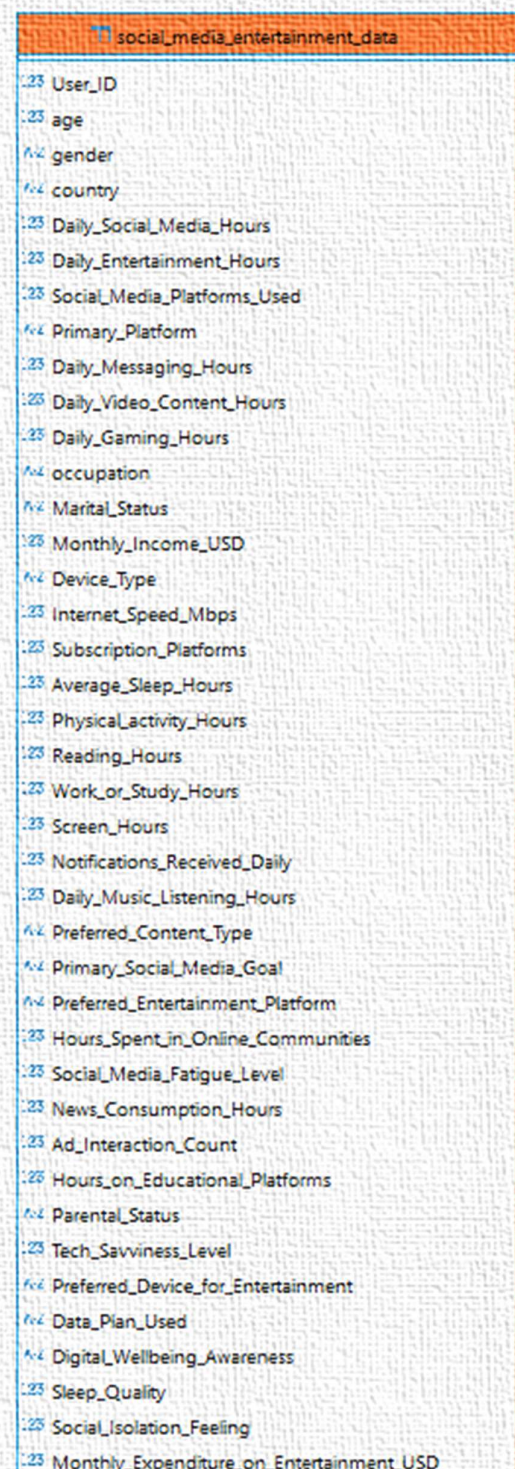
- Et  dell'utente

Gender (STR):

- Genere dell'utente
- Maschio, Femmina, Altro

Country (STR):

- Paese dell'utente



social_media_entertainment_data	
INT	User_ID
INT	age
STR	gender
STR	country
INT	Daily_Social_Media_Hours
INT	Daily_Entertainment_Hours
INT	Social_Media_Platforms_Used
STR	Primary_Platform
INT	Daily_Messaging_Hours
INT	Daily_Video_Content_Hours
INT	Daily_Gaming_Hours
STR	occupation
STR	Marital_Status
INT	Monthly_Income_USD
STR	Device_Type
INT	Internet_Speed_Mbps
INT	Subscription_Platforms
INT	Average_Sleep_Hours
INT	Physical_Activity_Hours
INT	Reading_Hours
INT	Work_or_Study_Hours
INT	Screen_Hours
INT	Notifications_Received_Daily
INT	Daily_Music_Listening_Hours
STR	Preferred_Content_Type
STR	Primary_Social_Media_Goal
STR	Preferred_Entertainment_Platform
INT	Hours_Spent_in_Online_Communities
INT	Social_Media_Fatigue_Level
INT	News_Consumption_Hours
INT	Ad_Interaction_Count
INT	Hours_on_Educational_Platforms
STR	Parental_Status
INT	Tech_Savviness_Level
STR	Preferred_Device_for_Entertainment
STR	Data_Plan_Used
STR	Digital_Wellbeing_Awareness
INT	Sleep_Quality
INT	Social_Isolation_Feeling
INT	Monthly_Expenditure_on_Entertainment_USD

Fig. 1 - Struttura Database

- Germania, Regno Unito, India, Canada, Stati Uniti d’America, Australia

Daily_Social_Media_Hours (FLOAT):

- Tempo medio giornaliero (in ore) trascorso dall'utente sulle piattaforme social.
- Indicatore chiave per le analisi che si andranno a svolgere

Primary_Platform (STR):

- Piattaforma social principale utilizzata dall’utente
- Facebook, Instagram, Twitter, YouTube, TikTok.

Occupation (STR):

- Categoria professionale dell’utente
- impiegato, disoccupato, libero professionista, studente, pensionato.

Monthly_Income_USD (FLOAT):

- Reddito mensile dell’utente, espresso in dollari statunitensi.

Average_Sleep_Hours (FLOAT):

- Numero medio di ore di sonno per notte.

Physical_activity_Hours (FLOAT):

- Numero medio di ore settimanali dedicate all'attività fisica.

Work_or_Study_Hours (FLOAT):

- Numero medio di ore giornaliere dedicate al lavoro o allo studio.

Screen_Hours (INT):

- Numero medio di ore trascorse davanti a uno schermo

Ad_Interaction_Count (INT):

- Numero giornaliero di interazioni dell’utente con gli annunci pubblicitari sulla piattaforma.

Sleep_Quality (INT):

- Valutazione soggettiva della qualità del sonno
- Scala da 1 (scarsa) a 10 (eccellente).

Social_Isolation_Feeling (INT):

- Grado di percezione della solitudine o isolamento sociale dell’utente
- Scala da 1 (basso) a 10 (alto).

Questi dati permettono di applicare tecniche di clustering e analisi delle community, segmentando gli utenti in base alle loro caratteristiche e interazioni.

2.1 Scelte tecniche per la gestione dei dati

Per la gestione dei dati è stato utilizzato **MariaDB**, un DBMS relazionale open-source scelto per la sua efficienza e scalabilità.

MariaDB è un sistema di gestione di database relazionali (RDBMS) open source, nato come fork di MySQL nel 2009. È stato creato dagli sviluppatori originali di MySQL per garantire la continuità dello sviluppo open source dopo l'acquisizione di MySQL da parte di Oracle. MariaDB è progettato per essere altamente compatibile con MySQL, consentendo una sostituzione diretta senza necessità di modifiche significative alle applicazioni esistenti.

L'interfaccia di gestione e interrogazione del database è stata realizzata con **DBeaver**, un software che permette di eseguire query SQL, visualizzare i dati e monitorare le prestazioni delle operazioni sul database.

DBeaver è uno strumento universale di gestione di database, open source e multiplatforma, progettato per sviluppatori, amministratori di database e analisti. Supporta una vasta gamma di database, tra cui MySQL, PostgreSQL, SQLite, Oracle e, naturalmente, MariaDB.

L'uso di DBeaver ha facilitato:

1. L'analisi esplorativa dei dati, permettendo di visualizzare rapidamente le caratteristiche principali della tabella utenti.
2. L'ottimizzazione delle query, attraverso strumenti di analisi delle prestazioni SQL integrati nel software.
3. Il debugging delle operazioni di lettura e scrittura, migliorando la qualità dei dati importati nel sistema.

2.3 Strategie di ottimizzazione delle query

Uno degli aspetti cruciali per garantire prestazioni ottimali è l'efficienza delle interrogazioni al database. L'analisi dei dati sociali comporta l'elaborazione di un

elevato numero di informazioni, rendendo necessario implementare strategie che riducano il tempo di esecuzione delle query, limitando il caricamento dei dati non essenziali.

Per evidenziare il processo di ottimizzazione, viene mostrato di seguito un metodo chiave che gestisce l'importazione degli utenti dal database applicando filtri dinamici in base ai parametri selezionati. Questo approccio consente di:

- Ridurre la quantità di dati elaborati, migliorando i tempi di risposta.
- Personalizzare le query in base ai filtri selezionati dall'utente, evitando ricerche inutilmente ampie.
- Mantenere il codice flessibile, permettendo l'integrazione di nuovi parametri di ricerca senza 2modificare l'intera logica.

```
1 usage (1 dynamic)
def importaUtenti(self, gender=None, social_time=None, platform=None,
                    isolation_level=None, ad_interaction=None, sleep_quality=None):

    query = "SELECT * FROM social_media_entertainment_data smed"
    filters = []
    params = {}
```

Fig. 2

Il primo step è l'inizializzazione della query (Fig. 2) di base che seleziona tutti i dati dalla tabella "social_media_entertainment_data" (alias smed) e con essa vengono inizializzati:

- **"filters"**: una lista per raccogliere i filtri da applicare alla query.
- **"params"**: un dizionario per contenere i valori da passare come parametri nella query SQL.

Successivamente vi è l'applicazione di filtri opzionali (Fig. 3-4) che, a seconda dei parametri passati alla funzione, vengono aggiunti alla query:

Filtraggio per gender (Genere): Se il genere è specificato e **non** è "Non specificato", viene aggiunto alla clausola WHERE ed il valore viene aggiunto al dizionario "params".

Filtraggio per social_time (Tempo sui social media): Viene usata una mappatura delle fasce di tempo per categorizzare gli utenti in base al tempo trascorso sui social.

```

if gender and gender != "Non specificato":
    filters.append("smed.gender = %(gender)s")
    params["gender"] = gender

# Mappatura delle fasce di tempo sui social
if social_time:
    social_time_ranges = {
        "1": "smed.Daily_Social_Media_Hours < 2",
        "2": "smed.Daily_Social_Media_Hours >= 2 AND smed.Daily_Social_Media_Hours <= 4",
        "3": "smed.Daily_Social_Media_Hours >= 4 AND smed.Daily_Social_Media_Hours <= 6",
        "4": "smed.Daily_Social_Media_Hours > 6"
    }
    if social_time in social_time_ranges:
        filters.append(social_time_ranges[social_time])

```

Fig. 3

Filtraggio per platform (Piattaforma principale usata): Se il valore è specificato, viene aggiunto il filtro, allo stesso modo operano:

- **Filtraggio per isolation_level** (livello di isolamento sociale)
- **Filtraggio per sleep_quality** (qualità del sonno)

```

if platform and platform != "Non specificata":
    filters.append("smed.Primary_Platform = %(platform)s")
    params["platform"] = platform

if isolation_level and isolation_level != "Non specificato":
    filters.append("smed.Social_Isolation_Feeling = %(iso_lvl)s")
    params["iso_lvl"] = isolation_level

# Mappatura delle fasce di interazione con gli annunci
if ad_interaction:
    ad_interaction_ranges = {
        "Low": "smed.Ad_Interaction_Count < 20",
        "Medium": "smed.Ad_Interaction_Count >= 20 AND smed.Ad_Interaction_Count <= 40",
        "High": "smed.Ad_Interaction_Count > 40"
    }
    if ad_interaction in ad_interaction_ranges:
        filters.append(ad_interaction_ranges[ad_interaction])

if sleep_quality and sleep_quality != "Non specificata":
    filters.append("smed.Sleep_Quality = %(sleep)s")
    params["sleep"] = sleep_quality

```

Fig. 4

D'altro canto, il **Filtraggio per ad_interaction** (Interazione con annunci pubblicitari) funziona a fasce come quello visto per social_time.

```
# Se ci sono filtri, li aggiungiamo alla query
if filters:
    query += " WHERE " + " AND ".join(filters)

self.utenti = DAO.getUtenti(query, params)
```

Fig. 5

Si giunge infine alla composizione della query finale (*Fig. 5*) dove, se sono stati aggiunti filtri, questi vengono concatenati con AND e aggiunti alla query di base. Il metodo chiama il DAO per recuperare i dati, passando al metodo “getUtenti” la query finale ed il dizionario contenente gli eventuali parametri. Il metodo esegue la query e restituisce i risultati (ovvero gli oggetti Utente) che vengono allocati nella lista “self.utenti”.

Capitolo 3 – Strutture dati ad alto livello ed algoritmi utilizzati

3.1 Architettura generale del software (Model-View-Controller)

Per garantire un'organizzazione modulare e scalabile, il software SocialEntertainment è stato sviluppato seguendo l'architettura Model-View-Controller (MVC). Questo approccio consente di separare in modo chiaro la logica di business dalla gestione dell'interfaccia utente, migliorando la manutenibilità e l'espandibilità del codice.

L'architettura MVC suddivide il software in tre componenti principali:

View: è responsabile della presentazione grafica e dell'interazione con l'utente. Utilizza la libreria Flet per generare un'interfaccia dinamica e intuitiva. La View riceve input dall'utente e aggiorna i dati visualizzati in base alle informazioni ricevute dal Controller.

Controller: funge da intermediario tra la View e il Model. Gestisce gli input dell'utente, elabora le richieste e aggiorna i dati attraverso il Model. Inoltre, applica la logica di business e invia alla View i dati necessari per l'aggiornamento dell'interfaccia.

Model: è responsabile della gestione dei dati e dell'interazione con il database. Contiene la logica per il recupero e l'analisi dei dati, compresa l'elaborazione tramite NetworkX e l'algoritmo ricorsivo. Il Model fornisce i dati richiesti dal Controller senza occuparsi della loro rappresentazione grafica.

Il flusso di interazione (*Fig. 6*) tra i componenti segue una logica ben definita: l'utente interagisce con la View, ad esempio selezionando parametri tramite i menu a tendina di Flet; il Controller riceve l'input, lo elabora e decide le operazioni necessarie; il Model recupera e processa i dati dal database o tramite elaborazioni

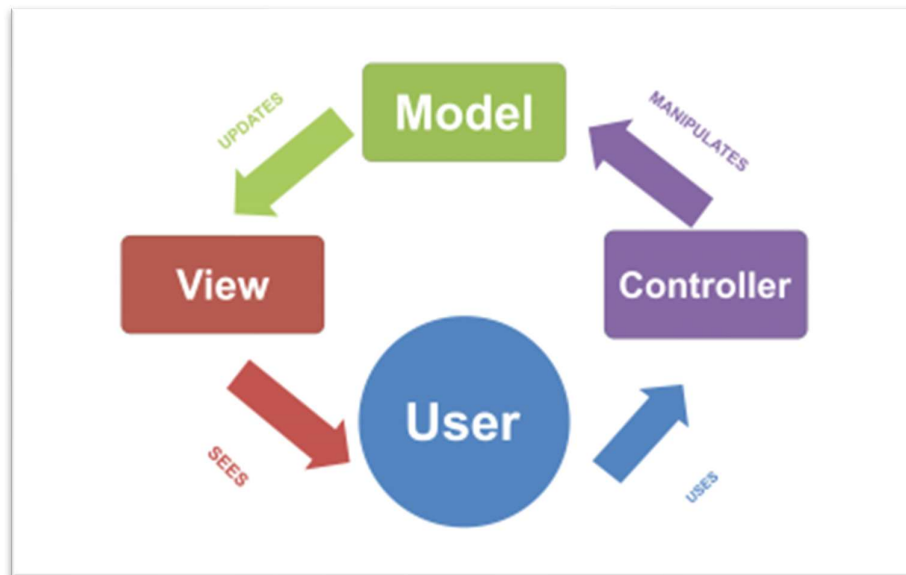


Fig. 6

di rete con NetworkX; infine, il Controller invia i dati elaborati alla View, che aggiorna l'interfaccia utente di conseguenza.

L'adozione di questa architettura offre numerosi vantaggi. La separazione dei compiti tra le tre componenti facilita la manutenzione e l'aggiornamento del software, permettendo di apportare modifiche a una parte del sistema senza impattare le altre. Inoltre, garantisce una maggiore scalabilità, poiché consente di aggiungere nuove funzionalità senza stravolgere l'intero codice. La flessibilità dell'implementazione consente di modificare la View indipendentemente dalla logica del Model, permettendo di adattare l'interfaccia utente a diverse esigenze senza dover intervenire sulla gestione dei dati. Infine, la chiarezza della struttura migliora la leggibilità del codice, rendendo più semplice individuare eventuali problemi o necessità di ottimizzazione.

3.2 Sviluppo dell'interfaccia grafica con Flet

L'interfaccia grafica di SocialEntertainment è stata sviluppata utilizzando la libreria Flet, un framework che consente di creare applicazioni web, desktop e mobili interattive in Python senza la necessità di competenze frontend. Flet permette di costruire interfacce utente reattive utilizzando controlli basati su Flutter, garantendo un aspetto professionale e coerente su diverse piattaforme.

L'adozione di Flet ha facilitato l'integrazione con il backend del software, permettendo una comunicazione fluida tra l'interfaccia utente e la logica di business. Grazie alla sua architettura event-driven, gli utenti possono interagire con l'applicazione in tempo reale, applicando filtri e analizzando dati senza dover ricaricare l'intera pagina.

Di seguito (Fig. 7) è mostrata l'interfaccia grafica di SocialEntertainment:

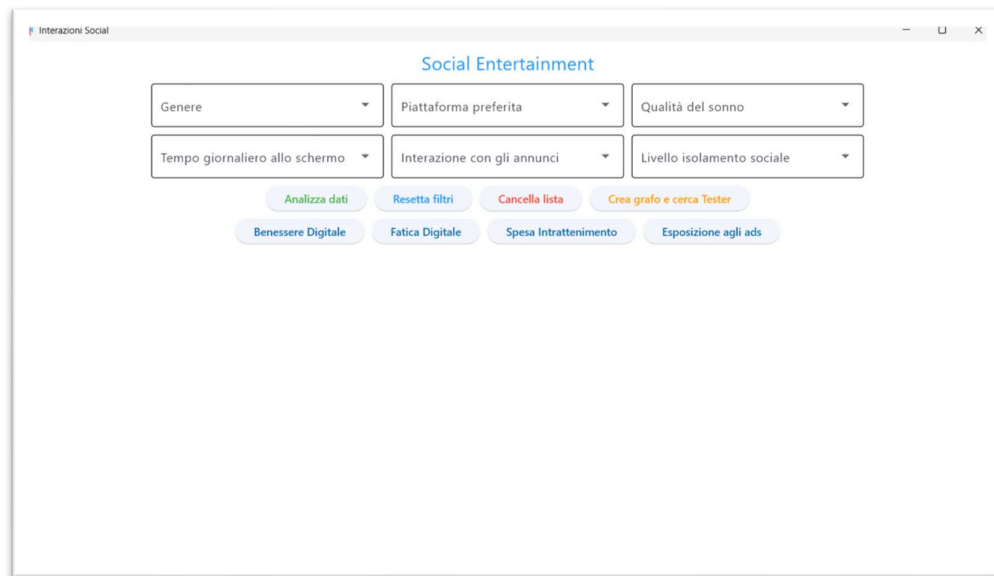


Fig. 7

Questa interfaccia presenta diversi elementi fondamentali:

Menu a tendina per selezionare parametri come genere, piattaforma preferita, qualità del sonno, tempo giornaliero allo schermo, interazione con gli annunci e livello di isolamento sociale.

Pulsanti funzionali che includono "Analizza dati", "Resetta filtri", "Cancella lista" e per gestire le operazioni principali dell'analisi.

Pulsanti dedicati per l'analisi di specifici aspetti, come il "Benessere Digitale", la "Fatica Digitale", la "Spesa Intrattenimento" ed "Esposizione agli ads".

Pulsante dedicato per la ricerca dei beta-tester, "Crea grafo e cerca Tester", che scatena l'algoritmo ricorsivo

Vista elenco dinamica, nella quale non è visibile alcuna informazione senza aver effettuato un'analisi, ma che rappresenta dove i risultati vengono riportati e sono fruibili.

Grazie a questa struttura, l'utente può personalizzare i parametri di analisi in maniera interattiva, ottenendo risultati aggiornati dinamicamente in base alle selezioni effettuate. L'uso di Flet garantisce inoltre una gestione ottimizzata degli eventi, migliorando le prestazioni dell'applicazione e offrendo un'esperienza utente fluida e immediata.

3.3 NetworkX e la modellazione del grafo

NetworkX è una libreria di Python progettata per la creazione, la manipolazione e l'analisi di strutture a grafo. Grazie alla sua flessibilità e alla sua intuitiva interfaccia, permette di rappresentare e studiare reti complesse in diversi ambiti, tra cui l'analisi delle reti sociali, la bioinformatica e la scienza dei dati.

Uno degli aspetti più interessanti di questa libreria è la possibilità di creare e gestire grafi di vario tipo, siano essi diretti, indiretti, pesati o non pesati. Inoltre, offre strumenti avanzati per analizzare la struttura e le proprietà dei grafi, calcolando misure di centralità, individuando percorsi ottimali e analizzando la connettività tra i nodi.

```
1 usage (1 dynamic)
def cercaTester(self):
    if len(self.utenti) < 30:
        self.creaGrafoPiccolo()
        for nodo in self.mappaUtenti.keys():
            self.ricorsione( parziale: [nodo], nodo, nodiVisitati: [nodo], diversityScore: 0)
    else:
        self.creaGrafo()
        for nodo in self.livelli[1]: # Partiamo dai nodi del livello 1
            self.ricorsione( parziale: [nodo], nodo, nodiVisitati: [nodo], diversityScore: 0) # Ogni chiamata
    return self.grafo.number_of_nodes(), self.grafo.number_of_edges(), self.bestSol
```

Fig. 8

Il metodo “*cercaTester()*”, come si evince dall'immagine (Fig. 8), prima di giungere all'obiettivo della sua creazione, la ricorsione, richiama certamente uno

dei due metodi (*“creaGrafoPiccolo()”* e *“creaGrafo()”*) che si occupano di strutturare una rete che rappresenta le connessioni tra utenti organizzati in livelli. Queste due strategie principali di creazione del grafo sono completamente indipendenti l’una dall’altra: nel caso in cui il numero di utenti sia inferiore a 30, viene generato un grafo completo in cui ogni nodo è connesso a tutti gli altri. Se invece il numero di utenti è maggiore, la generazione avviene attraverso un processo più selettivo, che organizza i nodi in livelli ordinati e collega tra loro solo quelli appartenenti a livelli adiacenti. Questo approccio garantisce un’ottimizzazione delle connessioni, evitando la creazione di collegamenti superflui che potrebbero appesantire l’elaborazione dei dati.

Un aspetto fondamentale di questa suddivisione riguarda il significato stesso dei livelli. Essi non hanno solo una funzione tecnica di riduzione della complessità computazionale, ma rappresentano anche un’importante caratteristica semantica legata al comportamento degli utenti. I livelli vengono infatti determinati in base al numero di piattaforme sociali utilizzate da ogni utente. Questo significa che gli utenti non vengono raggruppati casualmente, ma secondo il grado di interazione che hanno con diversi servizi digitali. È stato considerato, grazie a un vasto bacino di utenza, che questo criterio potesse aggiungere un ulteriore elemento di diversità all’analisi della rete. Infatti, differenziare tra chi utilizza esclusivamente l’applicazione in questione e chi invece è attivo su più piattaforme (fino a cinque) permette di comprendere meglio il comportamento degli utenti e identificare modelli di interazione più complessi.

L’immagine che segue rappresenta il metodo di creazione della mappatura degli utenti, *“creaMappe()”*, (Fig. 9) suddividendoli in livelli a seconda del numero di piattaforme utilizzate. Gli utenti vengono registrati in una mappa generale e successivamente raggruppati in un dizionario che suddivide gli utenti per livello. Se un utente non è iscritto ad alcuna piattaforma, viene escluso dal grafo per evitare di considerare nodi irrilevanti all’analisi.

```

1 usage
def creaMappe(self):
    for u in self.utenti:
        # Dizionario per raggruppare TUTTI gli utenti
        self.mappaUtenti[u.User_ID] = u
        # Dizionario per raggruppare utenti per livello
        if u.Subscription_Platforms == 0: # Escludo gli utenti non iscritti alla piattaforma
            continue

        livello = u.Social_Media_Platforms_Used

        if livello not in self.livelli:
            self.livelli[livello] = []
        self.livelli[livello].append(u.User_ID)

```

Fig. 9

Coloro che utilizzano una o più piattaforma/e vengono assegnati al rispettivo livello. Questo sistema consente non solo di ottimizzare la gestione dei nodi nel grafo, ma anche di evidenziare le diverse tipologie di utenti in base alla loro partecipazione a più servizi digitali.

```

def creaGrafoPiccolo(self):
    nodes = list(self.mappaUtenti.keys())
    self.grafo = nx.complete_graph(nodes, create_using=nx.Graph())
    print(self.grafo.number_of_edges())
1 usage

```

Fig. 10

Per la creazione del grafo nei casi in cui il numero di nodi sia inferiore a 30, il metodo “creaGrafoPiccolo()” (Fig. 10) utilizza una funzione di NetworkX che permette di generare un grafo completo partendo da una lista di nodi. Questo tipo di grafo è caratterizzato dal fatto che ogni nodo è direttamente connesso a tutti gli altri, garantendo così un'analisi estremamente approfondita delle connessioni tra gli utenti. Tuttavia, la costruzione di un grafo completo è computazionalmente onerosa, poiché il numero di archi cresce quadraticamente con il numero di nodi. Per questo motivo, nonostante l'elevato grado di completezza che offre in fase di analisi, questa struttura viene utilizzata esclusivamente nei casi in cui il numero di utenti sia limitato, evitando così di compromettere le prestazioni del sistema con un carico computazionale eccessivo.

```

def creaGrafo(self):
    t = time.time()
    nodes = list(self.mappaUtenti.keys())
    self.grafo.add_nodes_from(nodes)
    # Connessioni solo tra livelli adiacenti
    livelli_ordinati = sorted(self.livelli.keys())
    for i in range(len(livelli_ordinati) - 1): # Iteriamo solo fino al penultimo livello
        livello_attuale = livelli_ordinati[i]
        livello_successivo = livelli_ordinati[i + 1]
        for u1 in self.livelli[livello_attuale]:
            for u2 in self.livelli[livello_successivo]:
                self.grafo.add_edge(u1, u2) # Collegamento tra livelli adiacenti
                print(u1,u2)
    s = time.time()
    print("tempo creazione archi: " + str(s-t))

```

Fig. 11

Per quanto riguarda, invece, i casi in cui il numero di nodi sia superiore a 30 si utilizza il metodo “creaGrafo()” (Fig. 11). Escludendo le operazioni di registrazione del tempo per misurare l'efficienza dell'operazione, il processo inizia aggiungendo tutti gli elementi importati come nodi nel grafo, che non avviene in maniera completa, ma seguendo una logica strutturata. I livelli vengono ordinati e il ciclo principale scorre fino al penultimo livello, stabilendo connessioni tra nodi appartenenti a livelli adiacenti. Questo approccio garantisce che il grafo mantenga una struttura organizzata e scalabile, riducendo la complessità rispetto a un grafo completamente connesso.

Ogni nodo di un livello viene collegato a tutti i nodi del livello successivo, garantendo un flusso logico delle connessioni. Al termine dell'operazione, viene registrato e stampato il tempo impiegato per la creazione degli archi, fornendo così un parametro di valutazione delle prestazioni del metodo.

Questa implementazione consente di mantenere un equilibrio tra connettività e prestazioni computazionali. Il metodo evita il sovraccarico di connessioni inutili, favorendo un'analisi più efficiente delle relazioni tra gli utenti, con una struttura che permette di individuare rapidamente percorsi e connessioni chiave all'interno della rete.

3.4 Implementazione della Ricorsione e Calcolo delle Statistiche

Dopo aver modellato la rete degli utenti attraverso il grafo con NetworkX, il passo successivo è l'analisi approfondita delle connessioni e delle dinamiche interne alla rete. Per ottenere informazioni dettagliate sul comportamento degli utenti e sulle loro interazioni, è stata implementata una strategia basata sulla ricorsione.

3.4.1. La ricorsione come cuore dell'analisi della rete

Ma cos'è un algoritmo ricorsivo?

Si tratta di un “algoritmo espresso in termini di sé stesso”, ovvero in cui l'esecuzione dell'algoritmo su un insieme di dati comporta la semplificazione o suddivisione dell'insieme di dati e l'applicazione dello stesso algoritmo agli insiemi di dati semplificati. Quindi in informatica è una tecnica in cui una funzione richiama sé stessa per risolvere un problema suddividendolo in sottoproblemi più piccoli. Ogni chiamata ricorsiva avvicina il problema alla sua condizione di terminazione, evitando un loop infinito. Questo approccio è particolarmente utile per affrontare problemi strutturati gerarchicamente, come alberi, grafi e problemi di ricerca combinatoria.

Uno dei principali benefici della ricorsione è la chiarezza del codice, che consente di esprimere soluzioni eleganti e concise per problemi complessi. Tuttavia, richiede un'implementazione attenta per evitare problemi di stack overflow o inefficienze dovute a ripetute esecuzioni inutili degli stessi calcoli.

La ricorsione è ampiamente utilizzata in diversi ambiti dell'informatica e della matematica computazionale. Alcuni esempi di applicazioni ricorsive includono:

- *Algoritmi sui grafi*: come la ricerca in profondità (**DFS**) per esplorare reti connesse.
- *Strutture dati*: per l'elaborazione di alberi binari, heap e altre strutture gerarchiche.

- *Algoritmi di ordinamento e ricerca:* come QuickSort, MergeSort e la ricerca binaria.
- *Problemi combinatori:* come il calcolo del percorso minimo, il backtracking in problemi di ottimizzazione e la generazione di permutazioni.

La ricorsione rappresenta il cuore del problema analizzato, poiché consente di esplorare in profondità la rete, identificando la combinazione di utenti più adeguata. In un contesto come quello dell'analisi delle reti sociali, la ricorsione permette di comprendere come determinati nodi influenzino l'intero sistema, simulando percorsi di propagazione e di interazione.

L'implementazione di questo particolare algoritmo all'interno del progetto consente di esplorare scenari complessi, valutando non solo la presenza o l'assenza di connessioni, ma anche la loro qualità e il loro impatto sulla dinamica complessiva della rete e pertanto sugli interessi del provider. Infatti, non si limita solo all'analisi interna della rete, ma assume un ruolo cruciale anche nella richiesta. Il provider è interessato a un'applicazione che permetta l'analisi avanzata dei dati e, in particolare, la selezione di utenti potenzialmente interessati a testare nuove funzionalità della piattaforma (beta-tester). Inoltre, l'obiettivo principale è massimizzare le ore trascorse dagli utenti sull'applicazione, garantendo al contempo una diversificazione significativa tra i profili selezionati.

L'uso della ricorsione permette di:

- *Identificare utenti che trascorrono più tempo sulla piattaforma* e comprendere come le loro connessioni influenzano gli altri.
- *Selezionare utenti con profili diversi*, ottimizzando la varietà all'interno del gruppo di tester per garantire una rappresentatività più ampia.

Attraverso questi strumenti, il provider può affinare le proprie strategie di gestione dei beta-tester e migliorare l'esperienza degli utenti, garantendo un servizio più mirato e performante.

La funzione "*ricorsione()*" (Fig. 12 a) esplora il grafo per selezionare un sottoinsieme di utenti che massimizzino sia il tempo trascorso sulla piattaforma sia la diversità all'interno del gruppo selezionato. L'algoritmo segue una logica di **backtracking** per esplorare le possibili combinazioni di utenti, costruendo sottoinsiemi validi che rispettano vincoli specifici.

```

def ricorsione(self, parziale, nodoAttuale, nodiVisitati, diversityScore):
    print(parziale)
    print("-----")

    if len(parziale) == 5: # Se abbiamo raggiunto il numero massimo di nodi
        if diversityScore > self.maxScore: # Aggiorna solo se il punteggio è migliore
            self.bestSol = list(parziale)
            self.maxScore = diversityScore
            #print(self.bestSol)
            #print(f"score: + {self.maxScore:.2f}")
            #print("-----")
        return # Termina la funzione

    vicini = list(self.grafo.neighbors(nodoAttuale)) # Nodi adiacenti al nodo attuale
    nodiViciniBuoni = [n for n in vicini if n not in nodiVisitati] # Lista normale

```

Fig. 12 a

- Se il sottoinsieme parziale ha raggiunto il limite di 5 utenti, la funzione verifica se il punteggio di diversità (diversityScore) è superiore al massimo trovato finora (self.maxScore). Se il nuovo punteggio è migliore, aggiorna “self.bestSol” con la nuova selezione di utenti. Dopo aver aggiornato il valore massimo, la funzione termina con return, evitando esplorazioni superflue.
- Viene estratta la lista dei nodi adiacenti al nodo attuale (vicini) dove si filtrano i nodi già visitati, evitando selezioni ripetute.

```

for nodo in nodiViciniBuoni:
    # Assicura ordine crescente per evitare duplicati e massimizzare le ore sul social media
    t1 = self.mappaUtenti[nodo].Daily_Social_Media_Hours
    t2 = self.mappaUtenti[parziale[-1]].Daily_Social_Media_Hours
    if t2 > t1: # Mantiene ordine corretto
        continue
    parziale.append(nodo)
    nodiVisitati.append(nodo)

    # Calcoliamo solo il contributo aggiuntivo di nodo
    newScore = self.aggiornaScore(parziale, diversityScore, nodo)

    # Ricorsione con il nuovo score già aggiornato
    self.ricorsione(parziale, nodo, nodiVisitati[:], newScore) # Passiamo una copia di nodiVisitati

    parziale.pop()
    nodiVisitati.pop()

```

Fig. 12 b

Successivamente, (*Fig. 12 b*) per ogni nodo che è sia adiacente a quello attuale che non ancora visitato vengono applicate le istruzioni:

- Ogni nodo viene confrontato con l'ultimo aggiunto alla selezione ("*parziale*"), se t_1 (tempo utilizzo social del nodo) è maggiore di t_2 (tempo utilizzo social dell'ultimo nodo nel parziale) il codice prosegue con le istruzioni, sennò le interrompe e passa al nodo successivo (continue), Assicura ordine crescente per evitare duplicati e massimizzare le ore sul social media.
- Se il nodo passa il controllo, viene aggiunto a parziale (lista di utenti selezionati) e a "*nodiVisitati*" (per evitare selezioni ripetute).
- Si passa alla funzione "*aggiornaScore()*" aggiorna il punteggio di diversità del sottoinsieme corrente, la funzione considera fattori come nazionalità, occupazione, reddito e età. Di questa funzione verrà trattato meglio in seguito
- La funzione viene richiamata ricorsivamente con la lista degli utenti selezionati, Il nuovo nodo come nodo attuale, Una copia della lista dei nodi già visitati (per evitare modifiche accidentali) ed Il nuovo punteggio di diversità aggiornato.
- Infine, dopo la chiamata ricorsiva, il nodo viene rimosso da parziale e nodiVisitati, permettendo l'esplorazione di nuove combinazioni (backtracking).

Dopo aver visto il funzionamento della ricorsione, è fondamentale comprendere come viene calcolato il punteggio di diversità ("*diversityScore*") per ogni gruppo di utenti selezionati.

Questo ruolo è svolto dalla funzione sopracitata "*aggiornaScorere()*", (*Fig. 13*) che assegna un valore numerico alla combinazione di utenti in base a diversi fattori chiave.

```

1 usage
def aggiornaScore(self, utenti, punteggio, last):
    #-----
    # Indicizzo anche le ore del social e le facciamo pesare nel punteggio
    # In questo modo è come se una diversità valesse 1h in più sul social
    punteggio = punteggio + self.mappaUtenti[last].Daily_Social_Media_Hours
    #-----
    for u in utenti:
        if self.mappaUtenti[u].country != self.mappaUtenti[last].country:
            punteggio += 1
        if self.mappaUtenti[u].occupation != self.mappaUtenti[last].occupation:
            punteggio += 1
        salarioUltimo = self.mappaUtenti[last].Monthly_Income_USD
        punteggio += self.indiceDiversita(self.mappaUtenti[u].Monthly_Income_USD, salarioUltimo)
        punteggio += self.indiceDiversita(self.mappaUtenti[u].age, self.mappaUtenti[last].age)
    return punteggio

```

Fig. 13

Il primo elemento che viene considerato è il **tempo trascorso sui social**. La funzione inizia aggiungendo al punteggio complessivo il valore di “*Daily_Social_Media_Hours*” dell'ultimo utente inserito nella lista. Questa scelta riflette la volontà di dare più importanza ai gruppi di utenti che utilizzano la piattaforma per un tempo maggiore. In altre parole, più un utente è attivo sulla piattaforma, maggiore sarà l’impatto positivo sul punteggio totale.

Successivamente, si procede con un’analisi più approfondita della **diversità** del gruppo, confrontando le caratteristiche dell'ultimo utente con quelle degli altri utenti già presenti. Due aspetti vengono valutati in modo diretto:

- **Paese di origine:** se il nuovo utente appartiene a un paese diverso da quello di un altro utente del gruppo, il punteggio viene incrementato. Questo aspetto è importante per garantire una selezione di beta-tester più eterogenea.
- **Occupazione:** allo stesso modo, se il lavoro svolto dal nuovo utente è differente rispetto agli altri, si aggiunge un ulteriore punto.

Infine, il punteggio viene raffinato ulteriormente considerando variabili continue come **reddito e età**. Qui entra in gioco la funzione “*indiceDiversita()*”, (Fig. 14) che calcola un valore in base alla distanza tra i valori numerici. L'idea di base è che un gruppo composto da utenti con stipendi molto simili non aggiunge molta diversità, mentre un gruppo con utenti di fasce reddituali diverse avrà un punteggio

```
def indiceDiversita(self,v1,v2):  
    # Mi sono inventato questa funzione per avere un indice il più vicino possibile ad 1  
    # Inoltre tiene conto anche del fatto che se la differenza tra i valori è molta sarà un po più di 1  
    # viceversa se la differenza è poca sarà più vicino allo 0, infatti se è uguale sarà 0  
    #-----  
    # faccio la differenza in valore assoluto dei due valori  
    diff = abs(v1-v2)  
    # poi ne faccio la media  
    avg = (v1+v2)/2  
    # restituisco il rapporto dei due  
    return diff/avg  
    # casi estremi: sono uguali --> diff = 0 ==> indice = 0  
    # sono uno il doppio dell'altro --> diff = avg ==> indice = 1  
    # Ho fatto le prove e il massimo valore di differenza di stipendio è circa 1.8, invece per l'età è 1.33
```

Fig. 14

più alto. Lo stesso criterio viene applicato all'età, garantendo che i tester non appartengano tutti alla stessa fascia anagrafica.

Grazie a questo sistema, la selezione finale non avviene casualmente, ma segue criteri precisi per ottenere un **equilibrio tra utenti attivi e diversità del gruppo**.

Uno degli aspetti più innovativi del sistema è la funzione sopracitata “*indiceDiversita()*”, che permette nella sua estrema semplicità di quantificare in modo flessibile la differenza tra due valori numerici (ad esempio, reddito o età) senza imporre limiti rigidi. L'obiettivo è ottenere un **indice di diversità scalabile**, che rifletta in maniera accurata quanto due utenti siano effettivamente diversi tra loro.

Il funzionamento si basa su tre semplici passaggi chiave:

1. **Calcolo della differenza assoluta tra i due valori** (“*diff*” nell’immagine)
2. **Calcolo della media tra i due valori** (“*avg*” guardando l’immagine)
3. **Normalizzazione della differenza rispetto alla media** (“*return*” ciò che restituisce)

Questo metodo consente di esprimere la diversità in termini relativi, rendendo il confronto coerente anche tra grandezze di ordini di grandezza molto diversi.

Un aspetto importante della funzione è che il valore restituito può superare 1, una scelta voluta per enfatizzare ulteriormente le differenze quando sono particolarmente marcate. Ad esempio:

- Se due utenti hanno lo stesso reddito o la stessa età, la differenza è **0**, quindi l'indice sarà **0**.
- Se il reddito di un utente è il doppio di quello di un altro, l'indice sarà **1**.
- Se il reddito è ancora più elevato, l'indice **supererà 1**, sottolineando che la differenza è molto significativa.

Nei test effettuati, il valore massimo raggiunto per il reddito è stato circa **1.8**, mentre per l'età **1.33**. Questo significa che, in media, le differenze di reddito tra gli utenti selezionati tendono a essere più marcate rispetto alle differenze di età, il che rispecchia la realtà di una piattaforma dove gli utenti possono avere età simili ma condizioni economiche molto differenti.

Grazie a questa formulazione, il sistema non si limita a selezionare utenti con caratteristiche diverse, ma enfatizza le combinazioni in cui le differenze sono particolarmente marcate, portando a una selezione ancora più eterogenea e strategica.

3.4.2 L'impatto degli indici e delle statistiche

Nell'era digitale, la raccolta e l'analisi dei dati rappresentano un elemento essenziale per il successo delle grandi aziende e dei provider di servizi. Le statistiche non sono soltanto numeri, ma veri e propri strumenti decisionali che consentono di comprendere il comportamento degli utenti, ottimizzare strategie di business e migliorare l'efficienza operativa. Grazie all'analisi avanzata dei dati, le aziende possono adattare i loro servizi alle esigenze reali del mercato, incrementando il coinvolgimento degli utenti e, di conseguenza, la loro redditività.

Le statistiche rappresentano una risorsa imprescindibile sia per la crescita economica di un'azienda che per la sua gestione strategica. Dal punto di vista economico, l'analisi dei dati consente di individuare con precisione le preferenze degli utenti, ottimizzando l'offerta commerciale e migliorando la monetizzazione della piattaforma. Un esempio concreto è la pubblicità mirata, che, grazie alla raccolta di informazioni dettagliate sul comportamento degli utenti, permette di

proporre contenuti personalizzati, aumentando il tasso di conversione e il ritorno sugli investimenti pubblicitari. Un altro aspetto chiave è l'analisi delle transazioni e dei modelli di spesa degli utenti, che permette di calibrare in modo più efficace le strategie di prezzo e massimizzare i profitti derivanti da servizi premium o abbonamenti.

Dal punto di vista gestionale, le statistiche giocano un ruolo determinante nella personalizzazione dell'esperienza utente e nell'ottimizzazione delle risorse aziendali. Monitorare le interazioni degli utenti con la piattaforma consente di offrire servizi sempre più in linea con le loro aspettative, migliorando la fidelizzazione e riducendo il tasso di abbandono. Inoltre, la segmentazione del pubblico in base a dati demografici e comportamentali permette di differenziare l'offerta e proporre contenuti o funzionalità specifiche per ogni tipologia di utente. In parallelo, l'analisi delle metriche operative aiuta le aziende a ottimizzare le risorse tecnologiche, regolando in modo più efficiente il carico sui server, prevedendo eventuali picchi di traffico e riducendo i costi operativi.

Un altro ambito in cui le statistiche rivestono un ruolo cruciale è **l'innovazione e lo sviluppo di nuove funzionalità**. I dati raccolti forniscono informazioni fondamentali per individuare le esigenze emergenti degli utenti e anticipare le tendenze del mercato. Grazie a un'analisi accurata, è possibile testare nuove funzionalità in ambienti controllati, raccogliendo feedback prima di implementarle su larga scala. Questo approccio consente di ridurre il rischio di insuccesso e di ottimizzare il processo di sviluppo, garantendo che ogni nuova feature sia effettivamente utile e apprezzata dagli utenti.

L'applicazione offre quattro tipologie di statistiche differenti che si concentrano su differenti ambiti tematici che ho ritenuto possano essere d'interesse al provider/azienda:

- 1) La prima è incentrata sul **“Benessere Digitale”**.

Si tratta di un concetto sempre più rilevante nell'era moderna, in cui la tecnologia e i dispositivi digitali occupano una parte significativa della nostra vita quotidiana.

Questo indice mira a valutare quanto l'uso della tecnologia sia equilibrato rispetto ad altri aspetti fondamentali del benessere, come il sonno, l'attività fisica e le ore dedicate al lavoro o allo studio. Un utilizzo eccessivo degli schermi può portare a problemi di salute, ridotta produttività e persino impatti negativi sul benessere mentale. Per questo motivo, l'analisi del bilanciamento tra le diverse attività giornaliere è cruciale per comprendere lo stato di benessere digitale di un individuo. L'implementazione dell'analisi (*Fig. 15*) si basa sulla misurazione delle ore quotidiane dedicate a quattro macro-categorie di attività:

- **Attività fisica** (esercizio, movimento, sport)
- **Lavoro o studio** (tempo dedicato ad attività produttive)
- **Sonno** (ore di riposo notturno)
- **Tempo trascorso davanti a uno schermo** (incluso uso di smartphone, PC e TV)

Questi valori vengono convertiti in percentuali rispetto al totale di 24 ore, permettendo di ottenere un quadro chiaro della distribuzione del tempo giornaliero per ogni utente.

L'indice di equilibrio digitale viene poi calcolato fondamentalmente come:

$$BD = (Attività\ fisica + Sonno) - (Lavoro/Studio + Tempo\ davanti\ a\ uno\ schermo)$$

```

def calcola_equilibrio_digitale(self):
    equilibrio_digitale = {}
    # Calcolo indice e percentuali giornaliere
    for u in self.utenti:
        # Percentuali sul totale di 24 ore
        a = (u.Physical_activity_Hours * 100) / 24 # Attività fisica
        b = (u.Work_or_Study_Hours * 100) / 24 # Lavoro/studio
        c = (u.Average_Sleep_Hours * 100) / 24 # Sonno
        d = (u.Screen_Hours * 100) / 24 # Tempo davanti a uno schermo

        # Memorizziamo le percentuali per l'utente
        self.mappaPercentualiEd[u.User_ID] = (a, b, c, d)

        # Calcolo dell'indice di equilibrio digitale
        score = (u.Physical_activity_Hours +
                 u.Average_Sleep_Hours -
                 u.Work_or_Study_Hours -
                 u.Screen_Hours)
        equilibrio_digitale[u.User_ID] = score

    # Ordiniamo gli utenti in base all'Indice di Equilibrio Digitale
    risultati_top10 = sorted(equilibrio_digitale.items(), key=lambda x: x[1], reverse=True)[:10]
    risultati_bottom10 = sorted(equilibrio_digitale.items(), key=lambda x: x[1])[:10]

    return risultati_top10, risultati_bottom10

```

Fig. 15

Un punteggio più alto indica un miglior bilanciamento tra attività benefiche (sonno e movimento) e attività che possono generare stress o affaticamento (studio/lavoro e utilizzo degli schermi).

2) La seconda sulla “Fatica Digitale”.

Rappresenta, purtroppo, un fenomeno sempre più diffuso, strettamente legato all’uso intensivo della tecnologia e dei dispositivi digitali. Con l’aumento delle interazioni online, delle notifiche continue e del tempo trascorso nelle community digitali, molte persone sperimentano un sovraccarico cognitivo che può tradursi in stress, riduzione della concentrazione e affaticamento mentale. L’obiettivo di questo indice è quello di quantificare il livello di fatica accumulata dagli utenti, identificando i fattori che contribuiscono maggiormente a questo stato e fornendo indicazioni per ridurre l’impatto negativo di un uso eccessivo della tecnologia. L’analisi della fatica digitale (Fig. 16) si basa sulla misurazione di quattro macro-categorie che influiscono sull’affaticamento mentale e cognitivo:

- **Numero di notifiche ricevute giornalmente**, che rappresentano un'interruzione costante dell'attenzione.
- **Tempo trascorso nelle community online**, un indicatore della partecipazione attiva a discussioni digitali che possono generare un eccessivo coinvolgimento emotivo e mentale.
- **Livello di fatica percepita sui social media**, che riflette l'impatto soggettivo dell'interazione digitale sulla stanchezza mentale dell'utente.
- **Qualità del sonno**, parametro che può essere influenzato negativamente dall'uso eccessivo della tecnologia prima di dormire.

Questi valori vengono combinati per ottenere un indice che misura il livello di fatica digitale dell'utente, secondo la formula:

$$FD = (Notifiche\ Ricevute \times Tempo\ Online/24) + (Fatica\ Sociale - Qualita'\ Sonno)$$

```
def calcola_fatica_digitale(self):
    fatica_digitale = {}

    # Calcolo indice e percentuali giornaliere
    for u in self.utenti:
        a = u.Notifications_Received_Daily
        b = u.Hours_Spent_in_Online_Communities # Ore in community rispetto a 24h
        c = u.Social_Media_Fatigue_Level
        d = u.Sleep_Quality

        # Memorizziamo le percentuali per l'utente
        self.mappaPercentualiFd[u.User_ID] = (a, b, c, d)

        # Calcolo dell'indice di fatica digitale
        score = (a * b/24 + c - d)

        fatica_digitale[u.User_ID] = score

    # Ordiniamo gli utenti per punteggio di fatica digitale
    utenti_top10 = sorted(fatica_digitale.items(), key=lambda x: x[1], reverse=True)[:10]
    utenti_bottom10 = sorted(fatica_digitale.items(), key=lambda x: x[1])[:10]

    return utenti_top10, utenti_bottom10
```

Un valore più alto dell'indice indica una maggiore esposizione a fattori di stress

Fig. 16

digitale, segnalando un possibile sovraccarico mentale dovuto all'uso intensivo della tecnologia.

3) La terza tratta la “**Spesa per l'intrattenimento**”

La spesa per l'intrattenimento digitale è un indicatore fondamentale per comprendere i comportamenti di consumo degli utenti nel contesto delle piattaforme digitali. Con l'aumento delle offerte di contenuti a pagamento, dagli abbonamenti ai servizi streaming ai videogiochi online, analizzare il budget che gli utenti dedicano all'intrattenimento permette di tracciare tendenze di mercato e individuare gruppi di consumatori con esigenze diverse. Questo indice aiuta a determinare quanto un utente sia coinvolto economicamente nelle piattaforme digitali e quale sia il suo livello di investimento nei servizi di intrattenimento.

L'analisi di questo indice (*Fig. 17*) si basa sulla misurazione di due macro-categorie che influenzano la spesa digitale:

- **Spesa mensile destinata all'intrattenimento**, che rappresenta l'ammontare economico investito in piattaforme di streaming, gaming, servizi premium e altri contenuti digitali.
- **Numero di piattaforme di abbonamento utilizzate**, un indicatore della diversificazione del consumo di contenuti digitali.

Questi valori vengono combinati per ottenere un indice che misura il livello di spesa per l'intrattenimento, secondo la formula:

$$SI = (Spesa\ Mensile / 500) \times Numero\ di\ Piattaforme$$

```

def calcola_spesa_intrattenimento(self):
    spesa_intrattenimento = {}

    # Calcolo indice e percentuali giornaliere
    for u in self.utenti:
        a = u.Monthly_Expenditure_on_Entertainment_USD
        b = u.Subscription_Platforms # Normalizziamo considerando un massimo di 10 piattaforme
        if b == 0: # Escludo utenti non iscritti a piattaforme
            continue
        # Memorizziamo le percentuali per l'utente
        self.mappaPercentualiSi[u.User_ID] = (a, b, u.Preferred_Entertainment_Platform)

        # Calcolo dell'indice di spesa per l'intrattenimento
        score = (a/500 * b) # Normalizziamo su un massimo di 500$

        spesa_intrattenimento[u.User_ID] = score

    # Ordiniamo gli utenti per punteggio di spesa sull'intrattenimento
    utenti_top10 = sorted(spesa_intrattenimento.items(), key=lambda x: x[1], reverse=True)[:10]
    utenti_bottom10 = sorted(spesa_intrattenimento.items(), key=lambda x: x[1])[:10]

    return utenti_top10, utenti_bottom10

```

Fig. 17

La normalizzazione sulla soglia di 500 dollari permette di avere un riferimento standard per confrontare gli utenti con livelli di spesa molto diversi. Un valore più alto dell'indice indica un maggiore investimento economico nell'intrattenimento digitale.

4) L'ultima la "Esposizione ad inserzioni pubblicitarie"

L'esposizione agli annunci pubblicitari è un aspetto chiave dell'esperienza digitale moderna. Le piattaforme online, dai social media ai siti web di informazione, utilizzano strategie avanzate per massimizzare la visibilità degli annunci, influenzando così le decisioni di acquisto degli utenti. La frequenza con cui un utente interagisce con la pubblicità, il tempo trascorso davanti agli schermi e il numero di notifiche ricevute giocano un ruolo determinante nella costruzione dell'indice di esposizione agli ads. Analizzare questo parametro permette di comprendere meglio l'impatto della pubblicità digitale sugli utenti e di ottimizzare le strategie di targeting delle campagne di marketing.

L'analisi dell'esposizione pubblicitaria (Fig. 18) si basa sulla misurazione di tre macro-categorie:

- **Numero di interazioni con gli annunci**, che indica la frequenza con cui un utente visualizza e interagisce con le inserzioni pubblicitarie.
- **Tempo trascorso davanti agli schermi**, fattore che determina il potenziale numero di annunci visualizzati in base alla durata dell'attività online.
- **Numero di notifiche ricevute**, che può aumentare l'esposizione indiretta alla pubblicità attraverso richiami visivi e sonori.

Questi valori vengono combinati per ottenere un indice che misura il livello di esposizione pubblicitaria dell'utente, secondo la formula:

$$EA = (Interazioni\ con\ Ads + Notifiche\ Ricevute) \times (Tempo\ Online/24)$$

```
def calcola_esposizione_ads(self):
    esposizione_ads = {}

    # Calcolo indice e salvataggio dati reali
    for u in self.utenti:
        a = u.Ad_Interaction_Count # Numero puro
        b = u.Screen_Hours # Ore
        c = u.Notifications_Received_Daily # Numero puro

        # Memorizziamo i valori per l'utente
        self.mappaPercentualiEp[u.User_ID] = (a,b,c)

        # Calcolo dell'indice di esposizione agli ads
        score = (a + c) * b/24

        esposizione_ads[u.User_ID] = score

    # Ordiniamo gli utenti per punteggio di esposizione agli ads
    utenti_top10 = sorted(esposizione_ads.items(), key=lambda x: x[1], reverse=True)[:10]
    utenti_bottom10 = sorted(esposizione_ads.items(), key=lambda x: x[1])[:10]

    return utenti_top10, utenti_bottom10
```

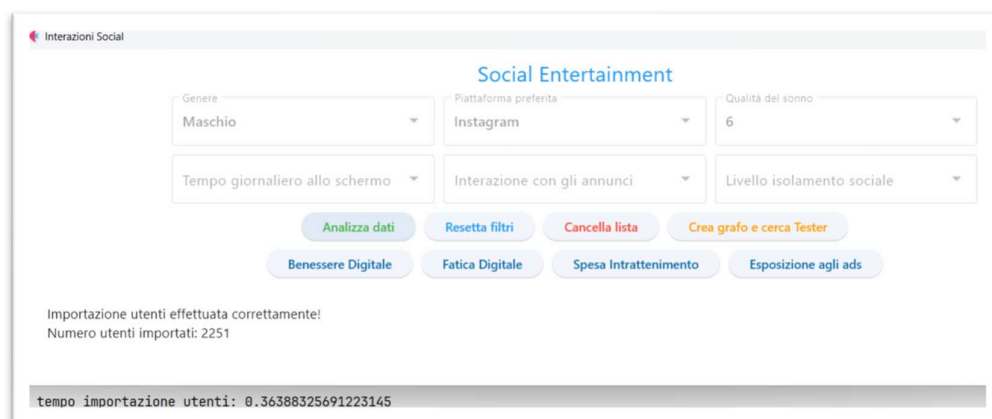
Fig. 18

Un valore elevato di questo indice suggerisce un'elevata esposizione agli stimoli pubblicitari, il che può avere impatti sia sul comportamento di consumo che sulla qualità dell'esperienza utente.

Capitolo 4 - Analisi delle Prestazioni e Test dell'Applicazione

4.1 Risultati sperimentali e performance

I test sperimentali (*Fig. 19*) hanno mostrato la reattività dell'interfaccia e la velocità dell'elaborazione.



The screenshot displays the 'Interazioni Social' application interface. At the top, there's a header with the title 'Interazioni Social' and a sub-header 'Social Entertainment'. Below this, there are several filter dropdowns: 'Genere' (Maschio), 'Piattaforma preferita' (Instagram), 'Qualità del sonno' (6), 'Tempo giornaliero allo schermo', 'Interazione con gli annunci', and 'Livello isolamento sociale'. A row of action buttons follows: 'Analizza dati' (green), 'Resetta filtri' (blue), 'Cancella lista' (red), and 'Crea grafo e cerca Tester' (orange). Below these are four more buttons: 'Benessere Digitale', 'Fatica Digitale', 'Spesa Intrattenimento', and 'Esposizione agli ads'. A status message at the bottom left reads 'Importazione utenti effettuata correttamente! Numero utenti importati: 2251'. At the very bottom, a grey bar shows the import time: 'tempo importazione utenti: 0.36388325691223145'.

Fig. 19

Il test sull'importazione ha dimostrato un'ottima efficienza: il caricamento di 2251 utenti è stato completato in soli 0,36 secondi.

Il sistema riesce a ridurre drasticamente i tempi di attesa grazie alla gestione intelligente delle query. Questo è fondamentale per migliorare l'esperienza utente e garantire analisi in tempo reale.

Uno degli aspetti più rilevanti è l'equilibrio (o benessere) digitale degli utenti (*Fig. 20*). La classificazione tra Top 10 e Bottom 10 evidenzia come l'uso intensivo dei dispositivi influenzi la qualità della vita. Gli utenti con un buon equilibrio digitale

mostrano una distribuzione del tempo più omogenea tra attività, studio, sonno e schermo, mentre quelli con un **indice negativo** accumulano eccessivo tempo sugli schermi e hanno bassi livelli di attività.

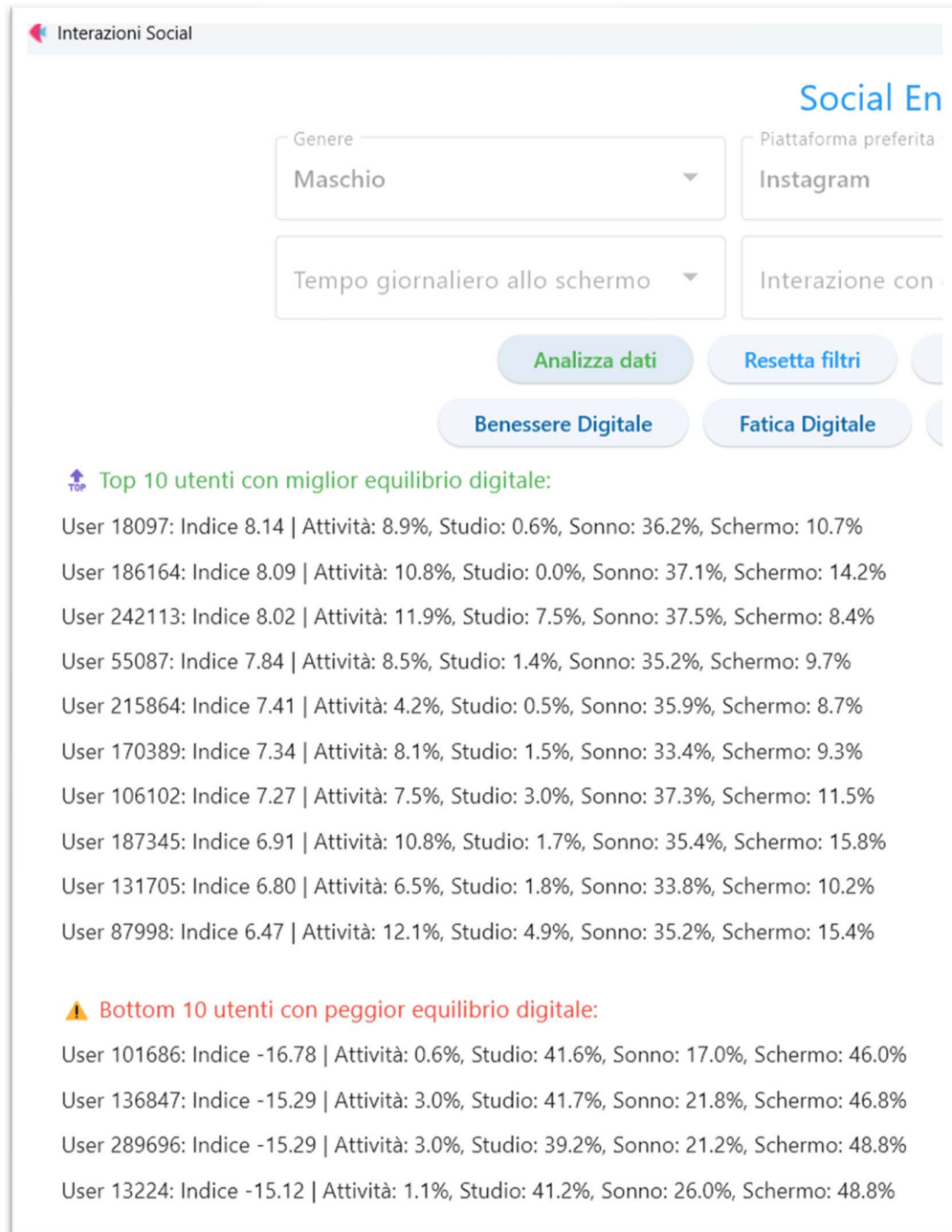


Fig. 20

L'applicazione riesce a individuare pattern ricorrenti che evidenziano comportamenti potenzialmente dannosi, fornendo un punto di partenza per migliorare "l'igiene digitale" degli utenti.

Un dato rilevante riguarda la quantità di annunci pubblicitari visualizzati (*Fig. 21*) e il loro impatto sull'esperienza utente. Gli utenti più esposti arrivano a superare le 11 ore di schermo al giorno, ricevendo centinaia di notifiche e interagendo frequentemente con gli ads.

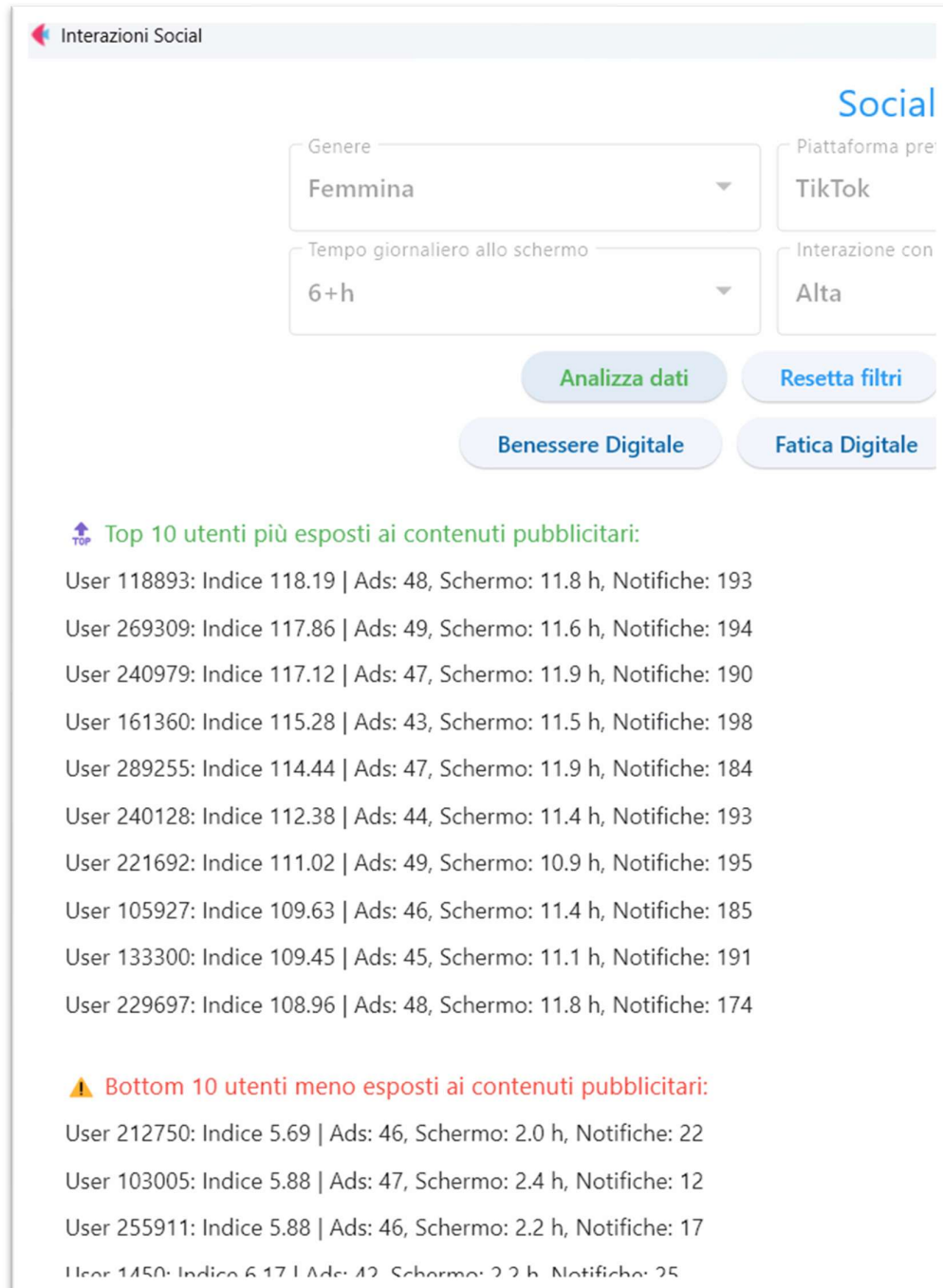


Fig. 21

L'analisi conferma che l'**overload informativo** è una realtà concreta. Utenti con un alto numero di notifiche e tempo di schermo potrebbero essere soggetti a fenomeni di affaticamento cognitivo e riduzione dell'attenzione.

L'analisi della spesa nell'intrattenimento (Fig. 22) mostra variazioni tra utenti con alti e bassi livelli di investimento nelle piattaforme digitali. L'informazione può risultare utile per individuare ad esempio trend di consumo o comunque in contesti di

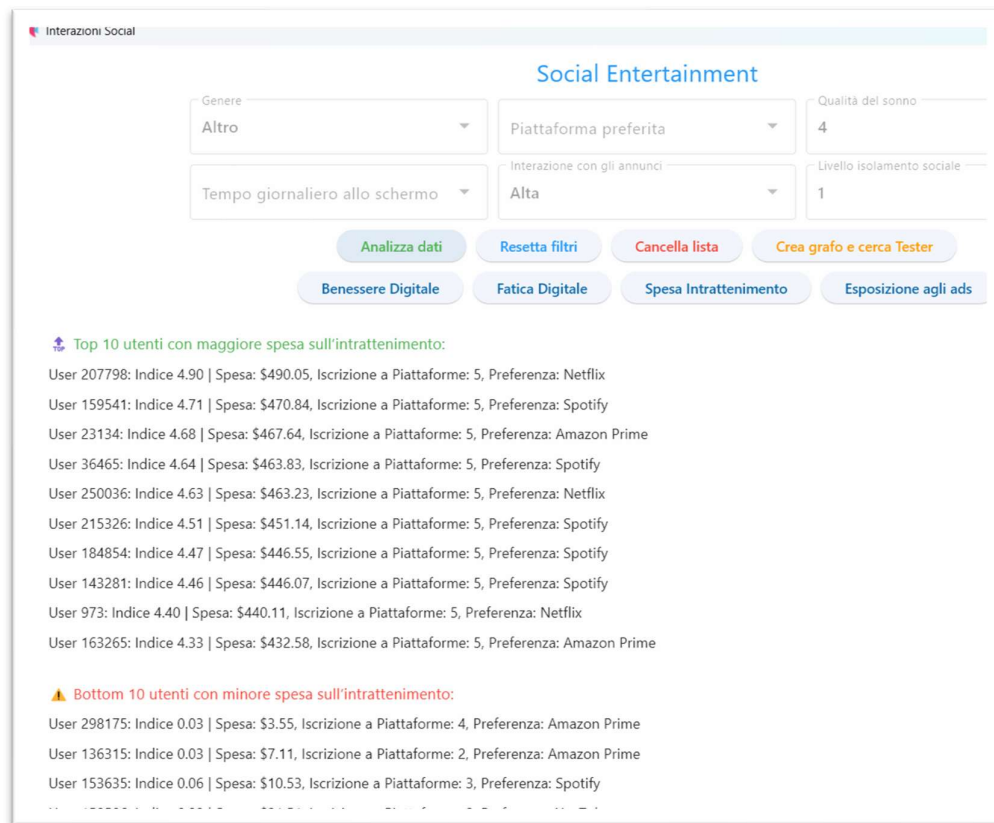


Fig. 22

L'analisi della fatica digitale (Fig. 23) ha evidenziato una chiara correlazione tra notifiche ricevute, tempo nelle community e qualità del sonno. Gli utenti con un indice elevato di fatica digitale tendono a registrare scarsa qualità del sonno e un alto numero di notifiche giornaliere, dimostrando una forte **dipendenza digitale**.

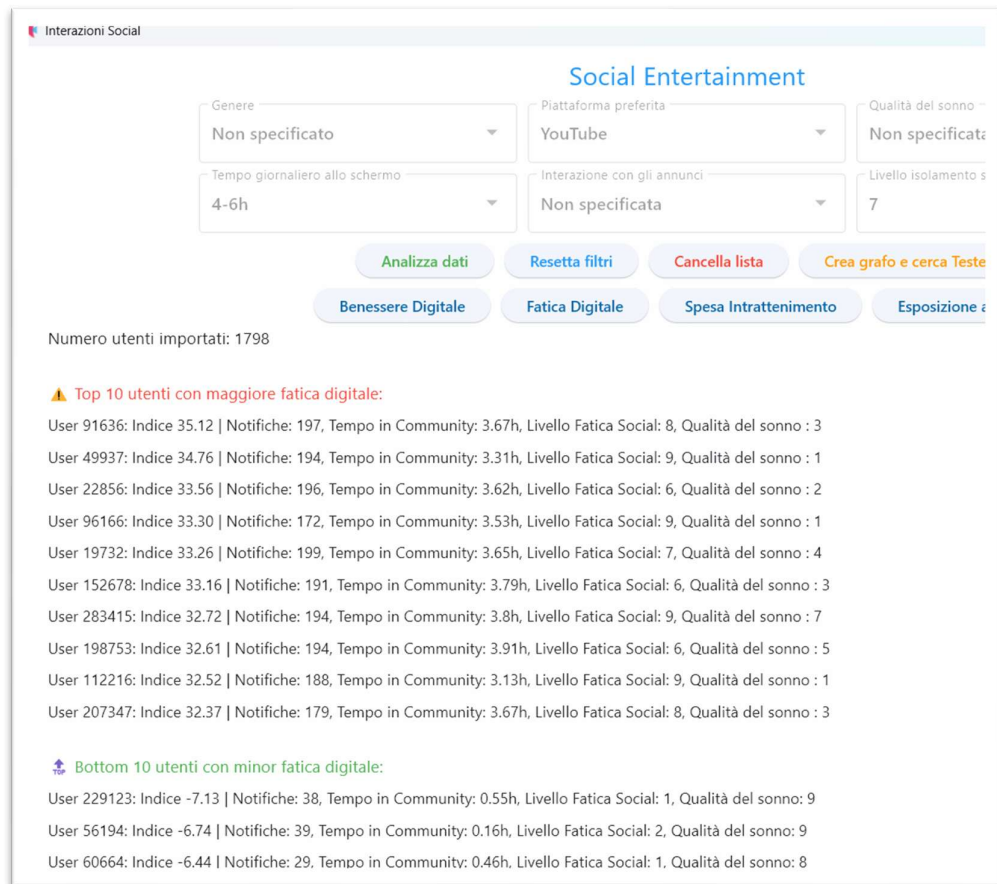


Fig. 23

La correlazione tra notifiche, tempo di esposizione e qualità del sonno conferma l'impatto negativo di un uso non regolato della tecnologia. Questa metrica potrebbe essere utile per suggerire strategie di **digital detox**.

Spostandoci invece sull'elaborazione chiave del software (Fig. 24-25-26), si evince bene dalle seguenti immagini che L'analisi dei dati condotta attraverso la ricorsione in SocialEntertainment ha prodotto risultati estremamente soddisfacenti, evidenziando la capacità del sistema di individuare profili utenti altamente differenziati e di fornire insight precisi sulle loro abitudini digitali.

L'algoritmo si è dimostrato efficace nel processare dataset complessi, restituendo non solo una lista di utenti selezionati, ma anche una panoramica dettagliata delle loro caratteristiche, tra cui età, paese di origine, occupazione e livello salariale.

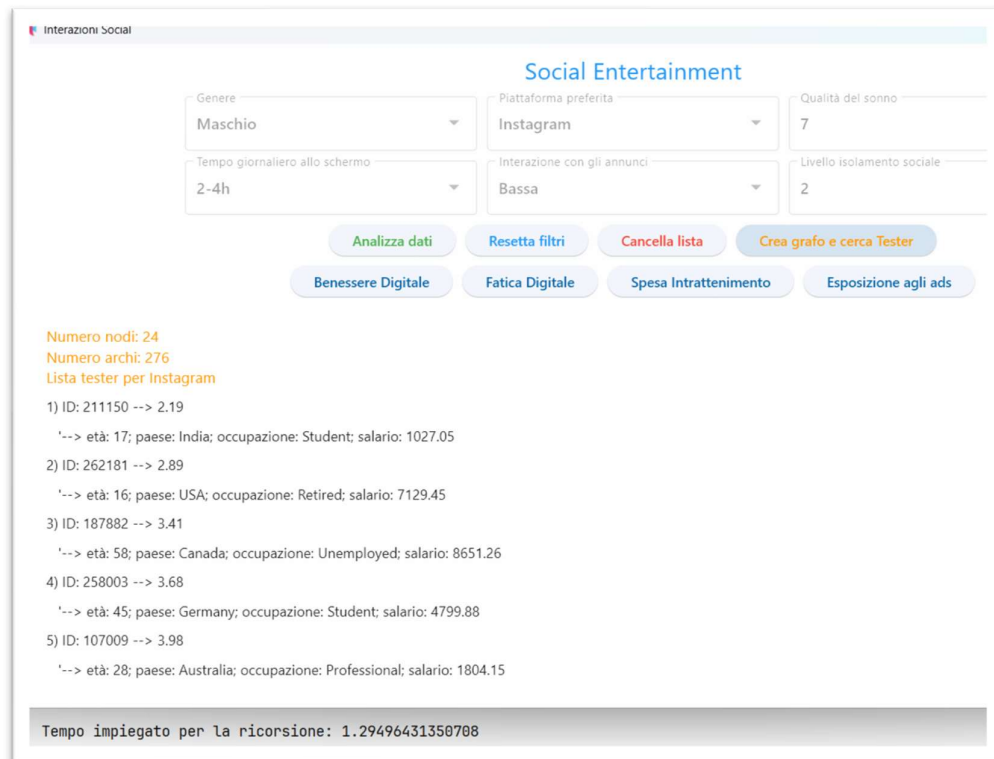


Fig. 24

Dai test emerge chiaramente come gli utenti identificati presentino profili eterogenei, con differenze significative in termini di fascia d'età e background lavorativo. Questo dimostra che il sistema è in grado di individuare gruppi con comportamenti digitali distinti, mettendo in luce l'impatto dei fattori sociodemografici sull'utilizzo delle piattaforme social.

Per esempio, su TikTok, (immagine pagina dopo) gli utenti selezionati spaziano da studenti a pensionati, con un'ampia distribuzione salariale, suggerendo che l'interazione con il social non è vincolata a uno specifico segmento della popolazione.

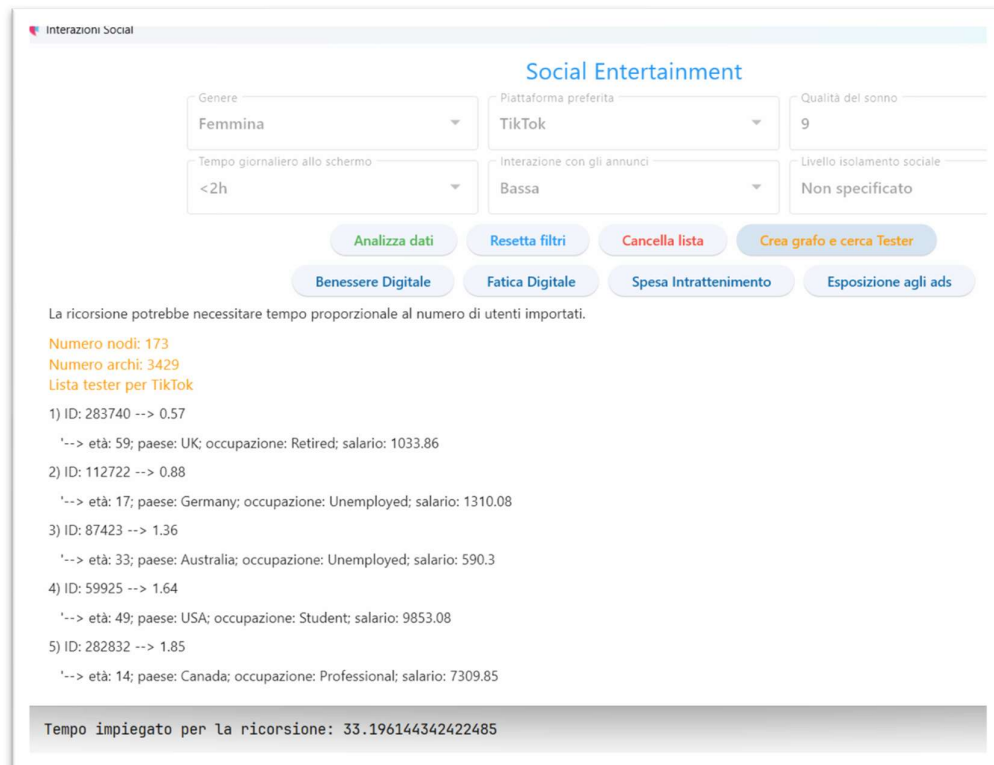


Fig. 25

Un altro aspetto chiave emerso dalle analisi è che le ore passate sulle piattaforme social risultano sempre massimizzate tra i profili selezionati, indipendentemente dalle variabili di genere o occupazione. Questo indica che l'algoritmo riesce a individuare con precisione gli utenti con maggiore engagement, ovvero coloro che passano più tempo connessi e interagiscono con i contenuti in modo attivo. Il confronto tra TikTok, Instagram e YouTube mostra che, sebbene i pattern di utilizzo possano variare da una piattaforma all'altra, la ricorsione riesce a estrapolare gli utenti più immersi nell'esperienza digitale, fornendo un dato chiaro e utile per l'analisi del loro comportamento.

Interazioni Social

Social Entertainment

Genere: Altro

Piattaforma preferita: YouTube

Tempo giornaliero allo schermo: 6+h

Interazione con gli annunci: Alta

Qualità del sonno:

Livello isolamento sociale: 8

Analizza dati Resetta filtri Cancella lista Crea grafo e cerca Tester

Benessere Digitale Fatica Digitale Spesa Intrattenimento Esposizione agli ads

Numero nodi: 104
 Numero archi: 1356
 Lista tester per YouTube

1) ID: 239285 --> 6.86
 '--> età: 60; paese: Australia; occupazione: Professional; salario: 9553.86

2) ID: 230780 --> 7.27
 '--> età: 18; paese: India; occupazione: Student; salario: 2096.23

3) ID: 186753 --> 7.37
 '--> età: 35; paese: USA; occupazione: Professional; salario: 8943.29

4) ID: 150563 --> 7.69
 '--> età: 45; paese: Canada; occupazione: Retired; salario: 657.48

5) ID: 126031 --> 7.93
 '--> età: 14; paese: UK; occupazione: Unemployed; salario: 4599.63

Tempo impiegato per la ricorsione: 2.6491503715515137

Fig. 26

4.2 Valutazione e conclusioni

L'analisi condotta su SocialEntertainment ha confermato l'efficacia del software nel fornire uno strumento strutturato e funzionale per l'analisi delle interazioni sociali, consentendo di selezionare utenti con caratteristiche ben definite in modo rapido ed efficiente. Il sistema offre un approccio innovativo alla segmentazione degli utenti, permettendo di filtrare i dati in base a parametri di interesse e di individuare profili significativi per studi comportamentali o per la selezione di beta-tester.

Grazie all'integrazione di NetworkX per la gestione del grafo e all'uso della ricorsione per l'analisi combinatoria, l'applicazione è in grado di individuare connessioni rilevanti tra gli utenti e di restituire risultati mirati con un alto livello di dettaglio. La capacità di selezionare utenti che massimizzano il tempo trascorso sulla piattaforma e che, al contempo, presentano profili diversificati, è un punto

chiave del progetto, in quanto consente ai provider di servizi digitali di ottenere campioni rappresentativi per l'analisi del comportamento degli utenti.

Dal **punto di vista gestionale**, il software risponde in modo efficace al problema descritto nel Capitolo 1, offrendo un'alternativa strutturata rispetto alle tradizionali metodologie di analisi dei dati sociali, spesso basate su estrazioni manuali e su aggregazioni poco flessibili. L'applicazione permette di ridurre i tempi di ricerca e di analisi, fornendo strumenti dinamici per la segmentazione degli utenti e l'estrazione di informazioni utili. Questo approccio consente alle aziende di ottimizzare le strategie di engagement e di gestione della community, offrendo un metodo data-driven per prendere decisioni più mirate.

Uno degli **aspetti più rilevanti** del software è la sua capacità di **scalare** in modo efficace all'aumentare delle esigenze di analisi. La possibilità di selezionare parametri specifici e di eseguire analisi filtrate consente di adattare il sistema a diversi scenari d'uso, garantendo flessibilità e personalizzazione dell'output.

Un altro punto di forza è la **precisione** della selezione degli utenti, che si basa su una combinazione di criteri di engagement e diversificazione. Il software è in grado di identificare utenti con elevata attività sulle piattaforme social e, al contempo, garantire una selezione eterogenea, fondamentale per studi di mercato e analisi comportamentali.

L'interfaccia utente, sviluppata con Flet, rappresenta un altro aspetto vincente del progetto. Grazie a una struttura intuitiva e a un sistema di interazione **dinamico**, l'utente può navigare facilmente tra le opzioni di analisi, selezionare parametri con pochi click e ottenere risposte immediate sui dati filtrati. Questa semplicità d'uso rende l'applicazione accessibile anche a utenti non esperti in analisi dati.

Nonostante i risultati soddisfacenti ottenuti nei test, il software presenta una **limitazione** strutturale legata all'uso della **ricorsione** per l'identificazione dei beta-tester. La natura combinatoria dell'algoritmo fa sì che, all'aumentare del numero di utenti e delle connessioni nel grafo, il numero di configurazioni possibili cresca esponenzialmente. In altre parole, se il dataset diventa troppo grande, il tempo di

esecuzione della selezione non può essere garantito in tempi polinomiali, portando a possibili rallentamenti o a tempi di elaborazione molto lunghi.

Questa limitazione, tuttavia, non incide sulle altre funzionalità del software, come la visualizzazione dei dati e l'analisi delle metriche sugli utenti, che restano rapide ed efficienti grazie alla gestione ottimizzata delle query SQL e alla selezione mirata delle informazioni. In un contesto reale, dove la selezione dei tester viene eseguita su insiemi di utenti di dimensioni contenute, il sistema mantiene comunque prestazioni soddisfacenti.

In conclusione, Il progetto SocialEntertainment ha raggiunto con gli obiettivi previsti, offrendo un framework efficace per l'analisi delle interazioni sociali e la segmentazione degli utenti. Il software si dimostra uno strumento utile per identificare trend comportamentali e per selezionare campioni rappresentativi di utenti, facilitando lo studio dell'engagement digitale.

L'applicazione si distingue per la sua capacità di fornire insight dettagliati e personalizzabili, garantendo al contempo un'esperienza utente fluida e intuitiva. Pur presentando alcune limitazioni computazionali legate alla crescita esponenziale della ricorsione, l'approccio adottato rimane altamente valido per la maggior parte dei casi d'uso reali, offrendo una soluzione innovativa e pratica per l'analisi dei social network.

Riferimenti

- **Flet.dev**
 - Documentazione ufficiale di Flet: <https://flet.dev/docs>
 - Repository GitHub di Flet: <https://github.com/flet-dev/flet>
 - Introduzione a Flet e sviluppo di UI: <https://flet.dev>
 - Architettura event-driven di Flet: <https://flet.dev/docs/architecture>
- **NetworkX**
 - Documentazione ufficiale:
<https://networkx.org/documentation/stable>
 - Repository GitHub: <https://github.com/networkx/networkx>
- **MariaDB**
 - MariaDB: <https://mariadb.org> | <https://mariadb.com>
- **HTML.it**
 - HTML.it: <https://html.it>
- **DBeaver**
 - DBeaver: <https://dbeaver.io>
- **Wikipedia**
 - Model-view-controller – Wikipedia:
<https://en.wikipedia.org/wiki/Model-view-controller>
 - https://it.wikipedia.org/wiki/Algoritmo_ricorsivo
 - <https://en.wikipedia.org/wiki/MariaDB>
 - <https://en.wikipedia.org/wiki/DBeaver>
- **GeeksforGeeks**
 - MVC Framework Introduction:
<https://www.geeksforgeeks.org/mvc-framework-introduction>
 - MVC Architecture - System Design:
<https://www.geeksforgeeks.org/mvc-architecture-system-design/>
 - Per la ricorsione: <https://www.geeksforgeeks.org/recursion/>
- **Programiz.com**
 - Ricorsione: <https://www.programiz.com/python-programming/recursion>

- **Altre fonti**

- European Central Bank: <https://www.ecb.europa.eu/>
- CCIAA Venezia Rovigo - Informazione Economica e Statistica:
<https://www.dl.camcom.it/>
- Sedus Insights - "Fatica Digitale nel Lavoro Ibrido":
<https://www.sedus.com/knowledge>
- AboutPharma - "Come affrontare la fatica digitale":
<https://www.aboutpharma.com/>
- Hrtools - "Fatica digitale: come far funzionare la tecnologia a tuo
favore": <https://www.hrtools.it/>
- Marie Claire - "La fatica della socialità digitale":
<https://www.marieclaire.it/>
- Osservatori Digital Innovation del Politecnico di Milano - "Contenuti
Digitali in Italia": <https://www.osservatori.net/>
- Spocket - "Costo medio dell'intrattenimento al mese":
<https://www.spocket.co/statistica>
- Magazine Qualità - "Informazione e intrattenimento, la spesa per i
contenuti": <https://www.magazinequalita.it/>
- Invitalia - "Fondo per l'intrattenimento digitale": <https://www.invitalia.it/>
- Framework360 - "Cos'è la Frequency CAP":
<https://www.framework360.it/>
- Spider4Web - "10 metriche e KPI di Facebook Ads":
<https://www.spider4web.it/>
- Human Analytica - "Facebook Ads: come funziona l'advertising su
Facebook": <https://www.humananalytica.it/>
- Fabio Pellencin - "Pianificare una campagna pubblicitaria":
<https://www.fabiopellencin.it/>

Licenza

Attribuzione - Non commerciale - Condividi allo stesso modo 4.0 Internazionale

(CC BY-NC-SA 4.0)

Questo lavoro è distribuito con Licenza Creative Commons Attribuzione - Non commerciale - Condividi allo stesso modo 4.0 Internazionale.

Tu sei libero di:

- **Condividere** — riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare questo materiale con qualsiasi mezzo e formato.
- **Modificare** — remixare, trasformare il materiale e basarti su di esso.

Il licenziante non può revocare questi diritti fintanto che tu rispetti i termini della licenza.

Alle seguenti condizioni:

- **Attribuzione** — Devi attribuire adeguatamente l'opera, fornire un link alla licenza e indicare se sono state effettuate delle modifiche. Puoi farlo in qualsiasi modo ragionevole, ma non in alcun modo che suggerisca che il licenziante avalli te o il tuo utilizzo dell'opera.
- **Non commerciale** — Non puoi utilizzare il materiale per scopi commerciali.
- **Condividi allo stesso modo** — Se remixi, trasformi il materiale o ti basi su di esso, devi distribuire i tuoi contributi sotto la stessa licenza dell'originale.

Nota:

Non ci sono ulteriori restrizioni — Non puoi applicare termini legali o misure tecnologiche che limitino legalmente altri dal fare qualsiasi cosa la licenza permetta.

Per una copia della licenza completa, visita <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>.