Università degli Studi di Salerno
Corso di Laurea in Informatica

Brandify

Progetto di Fondamenti di Intelligenza Artificiale

Alessio Inglese
Matricola: 0512118760

Prof. Fabio Palomba

Anno Accademico 2024/2025

Contents

1 Introduzione e Contestualizzazione

1.1 Definizione del Problema

Le raccomandazioni di prodotti musicali attuali presentano una serie di lacune che limitano l'esperienza degli utenti e il potenziale di connessione tra fan e artisti. Questi problemi possono essere sintetizzati in alcuni punti chiave.
In primo luogo, vi è una **mancanza di personalizzazione avanzata**, poiché molti sistemi si basano su metriche generiche come la popolarità o le vendite, trascurando le preferenze individuali. Questo approccio porta a suggerimenti meno rilevanti, riducendo l'engagement, soprattutto per gli appassionati di artisti o generi di nicchia. Inoltre, l'**ecosistema musicale è altamente frammentato**, con piattaforme di streaming, e-commerce e community di fan che operano in modo separato. Gli utenti devono navigare tra diversi siti e servizi per trovare prodotti correlati ai loro gusti musicali, rendendo il processo complesso e scoraggiante. Un ulteriore problema è l'**assenza di contestualizzazione temporale** nei sistemi di raccomandazione, che non tengono conto delle variazioni nei gusti musicali degli utenti nel tempo, come cambiamenti stagionali o eventi specifici. Di conseguenza, suggerimenti statici possono risultare fuori contesto, un esempio è quello di un utente che ascolta musica natalizia durante il periodo festivo ma continua a ricevere suggerimenti simili anche mesi dopo. Queste problematiche non solo diminuiscono la rilevanza delle raccomandazioni, ma ostacolano anche la scoperta di nuovi artisti e prodotti emergenti, limitando così il potenziale di connessione tra fan e artisti indipendenti.

Per affrontare queste lacune, nasce **Brandify**, una piattaforma che si propone di connettere in modo intelligente e personalizzato i gusti musicali degli utenti con i prodotti che più li rappresentano. Il sistema si distingue per due caratteristiche fondamentali:

1. **Personalizzazione avanzata**: Brandify utilizza i dati del profilo Spotify dell'utente per suggerire merchandise, brand e prodotti strettamente legati non solo agli artisti e ai generi che l'utente ama, ma anche a quelli che ha recentemente scoperto. Integrando sia le preferenze consolidate che le nuove scoperte musicali, questo approccio garantisce un'esperienza più mirata e rilevante.

2. **Uno spazio centralizzato**: La piattaforma funge da hub unico in cui l'utente può esplorare prodotti che riflettono i propri gusti musicali, eliminando la frammentazione e semplificando il processo di ricerca e acquisto.

In sintesi, Brandify non si limita a migliorare l'esperienza utente, ma rafforza anche il legame tra fan e artisti, creando un ecosistema musicale più integrato e coinvolgente.

1.2 Panoramica Strumenti e Tecnologie

Tra gli strumenti principali utilizzati per realizzare Brandify troviamo:

- **Spotify API e Spotipy**: Per estrarre i dati musicali dell'utente, trasformandoli in input utili per il sistema.

- **Algoritmo Genetico (GA)**: Il motore del sistema, che permette restituire in output un insieme ottimale di prodotti.

- **Flask e librerie Python**: Per gestire il backend e implementare un'interfaccia utente intuitiva ed efficace.

2 Obiettivi e Approccio Progettuale

2.1 Finalità del Sistema

Gli obiettivi principali di **Brandify** si concentrano sul fornire raccomandazioni di prodotti musicali altamente pertinenti per l'utente, bilanciando la precisione e la copertura dei prodotti. In particolare:

- **Precisione**: Garantire che i prodotti suggeriti siano altamente rilevanti rispetto ai gusti dell'utente, minimizzando la presenza di prodotti non pertinenti.
- **Copertura**: Assicurarsi che tutti i prodotti rilevanti per l'utente siano inclusi nelle raccomandazioni, evitando di non includere prodotti che potrebbero interessare all'utente.

2.2 Criteri di Successo

Per accertare che il progetto soddisfi i suoi obiettivi, definiamo i seguenti criteri di successo:

- **Precisione accettabile**: Almeno il 70% dei prodotti suggeriti devono essere pertinenti per l'utente. Una inesattezza del 30% è considerata accettabile come parte di una strategia di esplorazione.
- **Copertura elevata**: Massimizzare il numero di prodotti rilevanti individuati rispetto al totale disponibile.
- **Tempi di risposta ottimali**: Il sistema deve produrre raccomandazioni in meno di 5 secondi per il dataset corrente.

2.3 Trade-Offs

Durante lo sviluppo del sistema, è stato necessario gestire alcuni compromessi per bilanciare obiettivi spesso in competizione:

- **Precisione vs Copertura**: Un leggero sacrificio nella precisione è accettabile se consente di migliorare la copertura, garantendo una selezione più ampia di prodotti rilevanti.
- **Precisione e Copertura vs Durata**: Per mantenere i tempi di esecuzione accettabili, il sistema può ridurre lievemente la precisione o la copertura.

2.4 Motivazioni per la Scelta dell'Algoritmo Genetico

L'utilizzo dell' **Algoritmo Genetico** è stato scelto per la sua capacità di affrontare problemi di ottimizzazione combinatoria complessi, come quello di Brandify, che richiede la selezione di un sottoinsieme ottimale di prodotti in base ai gusti dell'utente. I motivi principali includono:

- **Adattabilità ai dati musicali**: I GA offrono un approccio flessibile per gestire preferenze musicali eterogenee e dinamiche, permettendo di combinare gusti relativi ad artisti e generi senza imporre ipotesi rigide sulla struttura dei dati o sulla loro distribuzione.
- **Personalizzazione**: La funzione di fitness del GA offre un alto grado di flessibilità, consentendo di modellare il comportamento del sistema per riflettere le preferenze dell'utente. Questo rende possibile premiare soluzioni più pertinenti rispetto ai gusti musicali, garantendo un approccio adattabile e mirato alle specifiche del problema.
- **Adattabilità a contesti contenuti**: Nel progetto, il dataset utilizzato è statico e di dimensioni limitate, rendendo gli Algoritmi Genetici particolarmente adatti. Questo approccio consente di ottimizzare la selezione di prodotti in modo efficiente, senza incorrere nei problemi di scalabilità che potrebbero sorgere con dataset più estesi o dinamici.

2.5 Esclusione di Tecniche Alternative

Sono state valutate altre tecniche per il motore di raccomandazione, ma scartate per le seguenti ragioni:

TF-IDF TF-IDF, una tecnica consolidata per l'analisi testuale e la valutazione della rilevanza di parole nei documenti, è stata esclusa per vari motivi:

- **Mancata compatibilità con l'approccio del progetto**: *Brandify* non utilizza un corpus testuale complesso, ma si basa su un confronto diretto tra i gusti musicali dell'utente (artisti e generi prelevati da Spotify) e i tag dei prodotti, estratti dai nomi e dalle descrizioni dei prodotti durante la fase di pre-processing del dataset. TF-IDF lavora bene su corpora testuali ampi e diversificati, focalizzandosi su parole chiave e frequenze relative, rendendolo poco adatto e superfluo a rappresentare le preferenze musicali basandosi su dati già strutturati.
- **Overhead computazionale non necessario**: Implementare TF-IDF avrebbe richiesto la costruzione di una matrice di confronto tra le preferenze musicali dell'utente e i tag estratti dai prodotti, un processo computazionalmente oneroso e superfluo rispetto al metodo di matching diretto adottato da *Brandify*, che utilizza un confronto basato su tag predefiniti e ottimizzati.

K-Means K-Means, un algoritmo di clustering non supervisionato, è stato scartato per i seguenti motivi:

- **Incompatibilità con la selezione personalizzata di prodotti**: Un ipotetico utilizzo di K-Means all'interno di questo contesto sarebbe stato di creare cluster di utenti, basandosi sui loro artisti e generi, e/o di prodotti, raggruppandoli in base a somiglianze nei tag. Tuttavia, questo approccio non soddisfa gli obiettivi di *Brandify*, che richiedono personalizzazione dinamica e ottimizzazione iterativa.

  L'assegnazione di utenti a cluster predefiniti riduce la capacità di rappresentare la complessità dei gusti musicali individuali. Ad esempio, un utente che apprezza sia elettronica che jazz sarebbe limitato a un cluster che cattura solo parte del suo profilo musicale. Allo stesso modo, raggruppare i prodotti in cluster statici, basandosi su somiglianze nei tag, limiterebbe la capacità di selezionare in modo mirato e personalizzato i prodotti rilevanti per ciascun utente in base ai sui gusti musicali unici.
- **Mancanza di ottimizzazione iterativa**: K-Means si limita a creare una segmentazione statica dei dati, senza un meccanismo per affinare iterativamente i risultati. L'assenza di un processo iterativo di ottimizzazione rende K-Means inadeguato a massimizzare la rilevanza e la copertura dei prodotti suggeriti, a meno di introdurre una complessità aggiuntiva significativa.

In sintesi, sia TF-IDF che K-Means sono stati valutati come non idonei rispetto agli obiettivi del progetto, che richiede un'ottimizzazione flessibile e adattabile alle preferenze musicali degli utenti.

3 Descrizione dell'Agente

3.1 Specifica P.E.A.S.

Performance Measure La performance dell'agente viene valutata attraverso tre metriche principali:

- **Precisione**: La percentuale di prodotti raccomandati che corrispondono effettivamente ai gusti musicali dell'utente.
- **Copertura**: La percentuale di prodotti rilevanti inclusi nelle raccomandazioni rispetto al totale dei prodotti rilevanti disponibili.
- **Soddisfazione dell'utente simulata**: Misurata come la percentuale di prodotti accettati dal profilo utente simulato. L'obiettivo minimo è un'accettazione superiore al 70% dei prodotti suggeriti.

Environment L'ambiente operativo dell'agente è costituito da:

- **Dataset Statico**: Un dataset contenente prodotti musicali.
- **Preferenze Utente**: Artisti e generi musicali estratti dalle abitudini di ascolto dell'utente tramite Spotify API.

Actuators Il sistema produce output attraverso:

- **Raccomandazioni generate**: Il sistema restituisce un sottoinsieme ottimale di prodotti musicali, contenuti nel dataset, selezionati dall'algoritmo genetico.
- **Formato output**: Una griglia visualizzabile su una pagina web, dove ogni prodotto è presentato con immagine, nome, descrizione, prezzo e un link per l'acquisto.

Sensors Il sistema acquisisce input attraverso:

- **Input primario**: Dati dell'utente estratti tramite Spotify API e normalizzati. I dati includono artisti e generi preferiti, nonché artisti e generi ascoltati di recente.
- **Input secondario**: Tag associati ai prodotti musicali, generati e normalizzati durante la fase di pre-processing del dataset musicale.

3.2 Proprietà dell'Ambiente

L'ambiente del sistema presenta le seguenti caratteristiche:

- **Osservabilità**: L'ambiente è *completamente osservabile*, poiché tutti i dati necessari (dataset dei prodotti e preferenze utente) sono accessibili prima dell'elaborazione.
- **Determinismo**: L'ambiente è *deterministico localmente*, poiché durante ogni esecuzione specifica, l'output dell'agente è prevedibile dato l'input. Tuttavia, l'ambiente diventa *non completamente deterministico* su un orizzonte temporale più ampio, poiché i dati Spotify possono cambiare nel tempo.
- **Staticità**: L'ambiente è *statico durante l'esecuzione*, in quanto i dati (dataset dei prodotti e preferenze utente) non si aggiornano automaticamente. Su scala temporale più ampia, può essere considerato *dinamico*, poiché le preferenze utente possono evolvere tra esecuzioni.

- **Episodicità**: L'ambiente è *episodico*, in quanto ogni raccomandazione è indipendente: il sistema non tiene traccia delle interazioni precedenti per influenzare le raccomandazioni successive.
- **Discrezione**: L'ambiente è *discreto*, poiché i dati (artisti, generi e tag dei prodotti) sono categorizzati e non variano su un intervallo continuo.

4 Elaborazione dei Dati

4.1 Origine e struttura del dataset

Il dataset `music-products.csv` rappresenta la base informativa su cui si fonda il sistema di raccomandazione *Brandify*. È stato progettato con l'obiettivo di fornire una rappresentazione realistica e diversificata di prodotti musicali, tale da rispecchiare scenari commerciali reali. Di seguito, vengono descritti i dettagli relativi alla sua origine, struttura e caratteristiche principali.

Origine del dataset

Il dataset è stato generato manualmente utilizzando un approccio basato su LLM, attraverso l'uso di Claude 3.5 Sonnet. L'obiettivo era creare un insieme di dati realistico e utilizzabile in un sistema di raccomandazione musicale, includendo almeno 100 prodotti unici. Per garantire la varietà e l'aderenza al contesto musicale, è stato fornito un prompt dettagliato che specificava:

- **Struttura dei dati**: Ogni prodotto è rappresentato da 5 campi principali: `name`, `price`, `description`, `image_url` e `product_url`.
- **Tipologie di prodotti**: Abbigliamento (es. t-shirt, felpe), vinili, accessori (es. portachiavi, tazze) e altri articoli legati al merchandising musicale.
- **Varietà di generi e artisti**: Copertura di almeno 30 generi musicali e un mix tra artisti contemporanei e storici.
- **Distribuzione dei prezzi**: Prezzi realistici, compresi tra 10€ e 70€, per riflettere un ampio spettro di opzioni.

Dopo la generazione iniziale, il dataset è stato ulteriormente rivisto manualmente per correggere eventuali incoerenze, migliorare la qualità delle descrizioni e garantire che i prodotti rispettassero i criteri del progetto.

Struttura del dataset

Il dataset è composto da 5 colonne, ognuna con una funzione specifica:

- `name`: Nome descrittivo del prodotto, che include riferimenti ad artisti (e in alcune istanze anche alle loro opere) o generi musicali (es. *"Taylor Swift 'Eras Tour' Hoodie", "Indie Rock Cap"*).
- `price`: Prezzo del prodotto, espresso con due decimali (es. *"29,99"*).
- `description`: Una descrizione composta da 1-2 frasi che evidenzia le caratteristiche principali del prodotto (es. materiali, design, contesto) oltre a contenere altri riferimenti all'artista o genere del prodotto.
- `image_url`: URL di un'immagine rappresentativa del prodotto, puntante a file statici organizzati.

- `product_url`: URL che conduce a una pagina fittizia di acquisto del prodotto.

Il dataset è stato progettato per essere compatibile con le esigenze del motore di raccomandazione, includendo informazioni facilmente processabili e rilevanti per l'analisi.

Caratteristiche del dataset

- **Dimensioni**: Il dataset include 145 prodotti unici, suddivisi in diverse categorie di merchandising musicale.
- **Dati realistici e diversificati**: I prodotti rappresentano una gamma di opzioni, dai vinili di artisti storici alle t-shirt di artisti emergenti, fino ad accessori legati a generi specifici.
- **Progettazione mirata**: Ogni prodotto è stato pensato per rappresentare gusti e interessi musicali specifici, facilitando il matching con le preferenze utente.

Esempio concreto

Ecco un estratto di alcune righe rappresentative del dataset `music-products.csv` in formato CSV:

```
name,price,description,image_url,product_url
"TOOL '10,000 Days' Hoodie",56.50,"High-quality hoodie inspired by TOOL's '10,000 Days'.",
/static/products/hoodie.jpg,https://example.com/hoodie/6062
"Gigi D'Agostino 'L'Amour Toujors' Keychain",15.32,
"Keep your keys secure with Gigi D'Agostino's 'L'Amour Toujors' keychain.",
/static/products/keychain.jpg,https://example.com/keychain/5750
"Heavy Metal Vinyl",37.33,
"Curated Heavy Metal vinyl compilation. Premium pressing with custom artwork.",
/static/products/vinyl.jpg,https://example.com/vinyl/8523
"Radiohead 'OK Computer' Keychain",16.76,
"Exclusive Radiohead's 'OK Computer' keychain for fans.",
/static/products/keychain.jpg,https://example.com/keychain/8018
"Grunge T-shirt",34.55,"Wear Grunge with pride on this cotton T-shirt.",
/static/products/t_shirt.jpg,https://example.com/t-shirt/5544
```

4.2 Pre-processing del dataset

1. Caricamento del CSV dei prodotti

Il primo passo prevede il caricamento del dataset `music-products.csv`, che contiene informazioni strutturate sui prodotti musicali. Come citato in precedenza, ogni riga del CSV rappresenta un prodotto unico, con colonne come `name`, `price`, `description`, `image_url` e `product_url`. Questo dataset fornisce la base su cui si costruiscono i successivi passaggi.

2. Caricamento dei dizionari di artisti e generi

I file `artist.txt` e `genre.txt`, generati tramite **Last.fm API**, una piattaforma online dedicata alla musica che offre una varietà di servizi e funzionalità per gli appassionati di musica, rappresentano una componente fondamentale per il sistema di tagging dei prodotti musicali. Questi dizionari contengono rispettivamente un elenco di artisti e generi musicali rilevanti, garantendo una copertura ampia e precisa del dominio musicale.

Generazione dei dizionari I dizionari sono stati creati, a monte della pipeline di pre-processing, utilizzando il modulo `lastfm_extraction.py`, che si interfaccia con Last.fm API per ottenere informazioni complete e aggiornate su artisti e generi. Last.fm è stato scelto per i seguenti motivi:

- **Ampia copertura**: Last.fm cattura sia artisti popolari che di nicchia, riflettendo tendenze globali e preferenze emergenti.
- **Compatibilità con Spotify**: I dati estratti da Last.fm sono altamente compatibili con i dati di Spotify, facilitando il matching tra preferenze utente e prodotti.
- **Tag rilevanti**: La community di Last.fm genera in tempo reale "tag" che rappresentano generi musicali e artisti, garantendo aggiornamenti costanti e rilevanza.

Processo di creazione Il modulo `lastfm_extraction.py` esegue i seguenti passi:

1. Recupera i generi musicali globali più popolari tramite il metodo `tag.getTopTags`.
2. Per ogni genere, ottiene un elenco di artisti associati utilizzando il metodo `tag.getTopArtists`.
3. Salva i risultati in due file distinti: `genres.txt` e `artists.txt`, organizzati in formato testo semplice.

Utilizzo dei dizionari I dizionari vengono utilizzati durante il pre-processing per individuare termini rilevanti nei campi `name` e `description` dei prodotti. Questo processo, noto come *dictionary lookup*, consente di identificare rapidamente artisti e generi presenti nei dati testuali, facilitando la creazione della colonna `tags`.

Esempio di dizionari

artists.txt	**genres.txt**
queen	rock
metallica	pop
taylor swift	heavy metal
black sabbath	indie rock
radiohead	electronic
...	...

Questi dizionari statici garantiscono un'alta precisione nel riconoscimento dei termini, contenendo ∼5000 artisti e ∼60 generi musicali, ma possono essere aggiornati manualmente o rigenerati per includere nuove informazioni.

3. Elaborazione delle righe del dataset

Dopo aver caricato i dizionari e il dataset, ogni riga del dataset `music-products.csv` viene elaborata per estrarre i tag inerenti al prodotto.

Combinazione dei campi `name` e `description` Il primo passo consiste nel combinare i campi `name` e `description` per creare un unico campo testuale ricco. Questo aumenta la probabilità di individuare riferimenti a artisti e/o generi musicali.

- **Input**: `name = "Black Sabbath Vinyl Record"`, `description = "Heavy metal classic."`
- **Output**: `text = "Black Sabbath Vinyl Record Heavy metal classic"`

Normalizzazione del testo Il testo combinato `text` viene normalizzato per garantire uniformità e ridurre il rumore. Le operazioni includono:

- Conversione in minuscolo per eliminare la sensibilità al caso.
- Rimozione di stopwords non significative, sia generiche che legate al contesto di prodotti musicali. (es. "for", "an", "classic", "premium").
- Eliminazione della punteggiatura e dei caratteri non alfarumerici tramite espressioni regolari (`regex`).

Esempio:

- **Input**: `text = "Black Sabbath Vinyl Record Heavy metal classic"`
- **Output**: `cleaned_text = "black sabbath heavy metal"`

Estrazione dei tag (Feature Extraction) Il passo successivo consiste nell'identificare artisti e generi utilizzando i dizionari `artists.txt` e `genres.txt`. Questo processo è noto come *dictionary lookup*, la quale funzione si occupa anche di sostituire gli spazi con underscore per garantire uniformità e generare una nuova colonna `tags` contenente informazioni strutturate sui prodotti.

- **Input**: `cleaned_text = "black sabbath heavy metal"`
- **Output**: `tags = ["black_sabbath", "heavy_metal"]`

Gestione dei sotto-generi Per evitare ambiguità, viene data priorità ai sotto-generi rispetto ai generi principali. Ad esempio, se `"indie_rock"` è trovato, le tag `"indie"` e `"rock"` vengono rimosse per mantenere una maggiore specificità semantica.

- **Input**: `tags = ["indie", "rock", "indie_rock"]`
- **Output**: `tags = ["indie_rock"]`

4. Passi finali

Pulizia finale dei tag Rimozione di eventuali caratteri speciali residui nei tag per garantire una rappresentazione uniforme e pulita.

Salvataggio dei risultati Infine, i tag estratti vengono salvati nella nuova colonna `tags` del dataset, che viene restituito come DataFrame pronto per l'elaborazione da parte dell'Algoritmo Genetico.

4.3 Dati utente estratti da Spotify

Una componente essenziale per personalizzare le raccomandazioni di *Brandify* consiste nell'estrazione e nella gestione dei dati dell'utente provenienti da Spotify. Questo processo si basa sull'utilizzo di **Spotipy**, una libreria Python open-source che funge da wrapper per **Spotify API**. È progettata per semplificare l'interazione con Spotify API, fornendo un'interfaccia più intuitiva e gestendo automaticamente molte delle complessità associate all'uso diretto delle API. Ciò comprende le seguenti fasi:

1. Autenticazione dell'utente

L'accesso ai dati dell'utente avviene tramite il protocollo OAuth2 di Spotify, che garantisce un'autenticazione sicura. Gli utenti effettuano il login attraverso un'interfaccia dedicata, ottenendo un token di autorizzazione necessario per interrogare le API di Spotify. Questo passaggio garantisce che solo dati autorizzati siano elaborati, rispettando la privacy dell'utente.

2. Recupero dei Dati Musicali

Una volta autenticato, il sistema utilizza il token di accesso per interagire con le API di Spotify e ottenere i dati musicali dell'utente. Il recupero dei dati avviene attraverso le seguenti API:

- `current_user_top_artists`
- `current_user_recently_played`
- `artist`

1. **Recupero degli Artisti e Generi Preferiti:**
   - `current_user_top_artists` viene chiamata per ottenere una lista di artisti top.
   - Per ogni artista, si estraggono i nomi e i generi musicali.
   - I nomi degli artisti vengono memorizzati in `top_artists`, mentre i generi vengono aggregati in `top_genres`.
2. **Recupero degli Artisti e Generi Recenti:**
   - `current_user_recently_played` viene chiamata per ottenere le tracce ascoltate di recente.
   - Per ogni traccia, si identifica l'artista principale.
   - Utilizzando l'API `artist`, si ottengono i generi musicali associati a ciascun artista.
   - I nomi degli artisti vengono memorizzati in `recent_artists`, mentre i generi vengono aggregati in `recent_genres`.

Questi dati rappresentano rispettivamente le preferenze musicali e gli ascolti più recenti dell'utente, fornendo una visione completa delle sue abitudini di ascolto.

Nota: È essenziale che il token di accesso disponga dei permessi necessari (`scopes`) come `user-top-read` e `user-read-recently-played` per accedere correttamente ai dati richiesti.

3. Pulizia e Normalizzazione dei Dati

I dati grezzi estratti dalle API di Spotify vengono sottoposti a un processo di pulizia e normalizzazione analogo al pre-processing del dataset per garantire coerenza con i tag dei prodotti:

- **Rimozione di caratteri speciali**: I nomi di artisti e generi vengono puliti utilizzando una funzione dedicata che rimuove tutti i caratteri non alfanumerici tramite espressioni regolari (`regex`).
- **Normalizzazione del testo**: Nel testo vengono sostituiti gli spazi con underscore e viene tutto convertito in minuscolo per garantire coerenza e uniformità.
- **Selezione delle occorrenze principali**: Le liste di artisti e generi (sia top che recenti) vengono convertite in oggetti `pandas.Series` per contare le occorrenze. Successivamente, vengono selezionati i primi N generi e i primi M artisti più frequenti. Questa fase permette di ridurre la dimensionalità dei dati, concentrandosi sugli elementi più rappresentativi del profilo musicale dell'utente.

Esempio di dati estratti Quindi per rappresentare in modo completo le preferenze musicali dell'utente, i dati estratti vengono organizzati e standardizzati come segue:

```
top_artists: ['tool', 'slipknot', 'radiohead', 'black_sabbath']
top_genres: ['metal', 'nu_metal', 'alternative_metal', 'rock']
recent_artists:['alice_in_chains', 'korn', 'city_morgue']
recent_genres: ['grunge', 'heavy_metal', 'psychedelic_rock', 'progressive_metal']
```

Questa aggregazione permette di ottenere un profilo musicale completo e rappresentativo per ciascun utente.

4.4 Integrazione con il Motore di Raccomandazione

Una volta completato il pre-processing del dataset e l'elaborazione dei dati Spotify, questi vengono integrati nel motore di raccomandazione di *Brandify*. Questa fase rappresenta il collegamento critico tra le preferenze musicali dell'utente e i prodotti musicali disponibili. Di seguito sono descritte le principali operazioni prima dell'avvio dell'Algoritmo Genetico:

- **Trasferimento dei Dati**: Il dataset dei prodotti e i dati utente aggregati vengono trasferiti al sistema di raccomandazione per l'elaborazione.
- **Definizione dei Parametri di Ricerca**: Prima di avviare il motore di raccomandazione, vengono configurati i seguenti parametri di ricerca selezionati tramite interfaccia web dall'utente, tra cui:
  - `min_price` e `max_price`: Limiti di prezzo opzionali per i prodotti da includere nelle raccomandazioni.
  - `preference_mode`: Modalità di preferenza selezionata dall'utente (focalizzata su artisti, generi o bilanciata.)

Successivamente l'algoritmo è pronto a processare i dati.

5 Algoritmo Genetico

L'algoritmo genetico (GA) rappresenta il cuore del motore di raccomandazione di *Brandify*. Ispirato ai principi dell'evoluzione naturale, il GA seleziona prodotti musicali ottimali basandosi sui gusti dell'utente, ottimizzando precisione e copertura. L'algoritmo integra personalizzazioni specifiche, come una funzione di fitness che premia prodotti coerenti con i generi e gli artisti preferiti dell'utente. Attraverso le operazioni di selezione, crossover e mutazione, l'algoritmo evolve progressivamente la popolazione per generare raccomandazioni sempre più mirate.

5.1 Codifica degli Individui

Gli individui nella popolazione dell'algoritmo genetico di *Brandify* sono rappresentati come array binari, in cui ogni elemento (o gene) indica se un prodotto musicale, identificato dalla posizione del gene nel dataset `df_products`, è incluso o escluso nella raccomandazione. La rappresentazione è la seguente:

- **Valore 1**: Indica che il prodotto associato al gene è selezionato per la raccomandazione.
- **Valore 0**: Indica che il prodotto associato al gene non è selezionato per la raccomandazione.

Motivazioni della Scelta La codifica binaria offre diversi vantaggi per il sistema di raccomandazione:

- **Semplicità**: La rappresentazione binaria è intuitiva essendo che rappresenta in modo naturale il problema di selezionare sottoinsiemi di prodotti basati su gusti musicali ed è facile da implementare, riducendo la complessità computazionale.
- **Flessibilità**: Non è necessario definire a priori un numero fisso di prodotti da includere nelle raccomandazioni. Questo permette al GA di restituire un sottoninsieme ottimale del dataset senza un limite superiore alla dimensione dell'output.

Implementazione La funzione `_generate_initial_population` gestisce la generazione della popolazione iniziale nel seguente modo:

```
Procedura: Generazione della popolazione iniziale

Input:
- Numero di soluzioni per popolazione (sol_per_pop)
- Numero totale di prodotti nel dataset (num_prodotti)

Output:
- Matrice (array bidimensionale) di popolazione iniziale

Passaggi:
1. Inizializza una lista vuota chiamata ‘popolazione_iniziale‘.
2. Ripeti fino a quando la dimensione della ‘popolazione_iniziale‘ è minore di ‘sol_per_pop‘:
   a. Genera un array binario casuale di dimensione ‘num_prodotti‘, dove:
      - Ogni elemento dell’array può essere 0 o 1.
      - 0 indica che il prodotto non è selezionato.
      - 1 indica che il prodotto è selezionato.
   b. Aggiungi l’array generato alla lista ‘popolazione_iniziale‘.
3. Converti la lista ‘popolazione_iniziale‘ in una matrice (array bidimensionale).
4. Restituisci la matrice risultante.
```

5.2 Fitness Function

La funzione di fitness guida l'evoluzione delle soluzioni generate dall'algoritmo genetico, bilanciando l'affinità dei prodotti selezionati con le preferenze dell'utente e applicando penalizzazioni per garantire precisione e copertura.

Calcolo dell'Affinità dei Prodotti Selezionati Per ogni prodotto selezionato si calcola un punteggio di affinità basato sulle preferenze dell'utente:

- Combinazione delle preferenze degli artisti e dei generi, includendo sia quelli preferiti sia quelli recentemente ascoltati.
- Valutazione di match tra le tag del prodotto e le preferenze dell'utente, assegnando un peso predefinito in base alla modalità di ricerca configurata:
  - **Artist**: Considera solo gli artisti.
  - **Genre**: Considera solo i generi.
  - **Balanced**: Considera entrambi.

Implementazione Il punteggio di affinità per un prodotto viene calcolato tramite la funzione `_evaluate_product_score` nel seguente modo:

```
Procedura: Calcolo del punteggio di affinità di un prodotto

Input:
- Indice del prodotto (product_idx)
- Tag associati al prodotto (products_tags[product_idx])
- Dati utente:
    - Artisti preferiti (user_data["artists"])
    - Artisti recentemente ascoltati (user_data["recent_artists"])
    - Generi preferiti (user_data["genres"])
    - Generi recentemente ascoltati (user_data["recent_genres"])
- Modalità di ricerca (preference_mode)
- Pesi di affinità per artisti e generi (affinity_weights)

Output:
- Punteggio di affinità (affinity_score)

Passaggi:
1. Ricava le tag del prodotto specificato (p_tags) come un insieme.
2. Combina gli artisti preferiti e recenti dell'utente in un unico insieme (user_artists).
3. Combina i generi preferiti e recenti dell'utente in un unico insieme (user_genres).
4. Inizializza il punteggio di affinità (affinity_score) a 0.
5. Determina la modalità di ricerca (preference_mode) e calcola l'affinità:
   a. Se preference_mode è "artist":
      - Se p_tags ha una corrispondenza con user_artists, assegna
        affinity_score al peso corrispondente agli artisti (affinity_weights["artists"]).
   b. Se preference_mode è "genre":
      - Se p_tags ha una corrispondenza con user_genres, assegna
        affinity_score al peso corrispondente ai generi (affinity_weights["genres"]).
   c. Se preference_mode è "balanced":
      - Se p_tags ha una corrispondenza con user_artists, aggiungi
        il peso degli artisti (affinity_weights["artists"]) ad affinity_score.
      - Se p_tags ha una corrispondenza con user_genres, aggiungi
        il peso dei generi (affinity_weights["genres"]) ad affinity_score.
6. Restituisci il punteggio di affinità calcolato (affinity_score).
```

Applicazione delle Penalizzazioni Dopo aver calcolato tutti i punteggi di affinità per i prodotti selezionati, le penalizzazioni vengono calcolate e applicate alla fine della funzione di fitness per garantire precisione e copertura:

- **Penalità di Precisione**:
  - Ogni prodotto selezionato senza affinità con i gusti dell'utente è considerato un falso positivo e penalizzato con un peso predefinito.
- **Penalità di Copertura**: Per garantire una copertura adeguata dei prodotti inerenti alle preferenze dell'utente, viene effettuato un pre-compute all'inizio del GA degli indici dei prodotti rilevanti ed essi vengono salvati in una struttura dedicata, per un utilizzo rapido durante la valutazione della fitness:
  - Ogni prodotto rilevante non incluso nella selezione è considerato un falso negativo e penalizzato con un peso predefinito.

Calcolo della Fitness Finale Il punteggio di fitness, indicato come F, è calcolato bilanciando il punteggio totale di affinità e le penalizzazioni associate a precisione e copertura. La formula utilizzata è:

$$F = \text{total\_affinity} - \text{precision\_penalty} - \text{coverage\_penalty}$$

Dove i termini sono definiti come segue:

- $\text{total\_affinity} = \sum_{i \in \text{selected\_indices}} \text{affinity\_score}(i)$: Somma dei punteggi di affinità dei prodotti selezionati, calcolati in base alla coerenza con le preferenze dell'utente.
- $\text{precision\_penalty} = w_p \cdot \sum_{i \in \text{selected\_indices}} 1(\text{affinity\_score}(i) = 0)$: Penalizzazione proporzionale al numero di prodotti selezionati che non corrispondono ai gusti dell'utente (*falsi positivi*), dove w_p è il peso della penalità per precisione.
- $\text{coverage\_penalty} = w_c \cdot |\text{relevant\_indices} \setminus \text{selected\_indices}|$: Penalizzazione proporzionale al numero di prodotti rilevanti (*falsi negativi*) non inclusi nella selezione, dove w_c è il peso della penalità per copertura.

Questo approccio bilancia l'obiettivo di massimizzare l'affinità totale dei prodotti selezionati con la necessità di minimizzare i prodotti irrilevanti e di garantire una copertura adeguata delle preferenze dell'utente.

Implementazione Il punteggio di fitness per ogni soluzione viene calcolato tramite la funzione `_fitness_func`, la quale richiama anche la funzione `_evaluate_product_score`, nel seguente modo:

```
Procedura: Calcolo della Fitness di una Soluzione

Input:
- Istanza dell'algoritmo genetico (ga_instance)
- Soluzione (array binario) che rappresenta i prodotti selezionati
- Indice della soluzione nella popolazione (solution_idx)

Output:
- Valore di fitness (total_affinity - precision_penalty - coverage_penalty)

Passaggi:
1. Identifica gli indici dei prodotti selezionati:
   a. Trova tutti gli indici i in cui solution[i] == 1.
   b. Salva questi indici in "selected_indices".

2. Calcola il punteggio totale di affinità:
   a. Inizializza "total_affinity" a 0.
   b. Per ogni indice in "selected_indices", calcola il punteggio di affinità
      del prodotto richiamando la procedura 'Calcolo del punteggio
      di affinità di un prodotto'.
   c. Somma i punteggi ottenuti e assegna il risultato a "total_affinity".

3. Calcola la penalità di precisione:
   a. Inizializza "precision_penalty" a 0.
   b. Per ogni indice in "selected_indices":
      - Se il punteggio di affinità del prodotto è 0, incrementa "precision_penalty"
        di un valore proporzionale a "penalty_weight_non_match".

4. Calcola la penalità di copertura:
   a. Inizializza "coverage_penalty" a 0.
   b. Confronta i prodotti selezionati ("selected_indices") con quelli rilevanti
      precomputati ("relevant_indices").
   c. Trova i prodotti rilevanti non inclusi nella selezione.
      e salva il risultato in "missing_relevant_products".
   d. Moltiplica la dimensione di "missing_relevant_products" per "penalty_missing_relevant"
      per ottenere "coverage_penalty".

5. Calcola il punteggio finale di fitness:
   a. Sottrai "precision_penalty" e "coverage_penalty" da "total_affinity".
   b. Restituisci il risultato come punteggio netto di fitness.
```

5.3 Selezione, Crossover e Mutazione

Selezione

Definizione La selezione degli individui nell'algoritmo genetico di *Brandify* si basa su una combinazione di **Steady State Selection (SSS)** ed **elitismo**. Questo approccio bilancia l'esplorazione dello spazio delle soluzioni con la preservazione delle migliori soluzioni già identificate.

Motivazioni L'adozione di SSS in combinazione con l'elitismo offre diversi vantaggi nel contesto del motore di raccomandazione, tra cui:

- **Preservazione delle soluzioni migliori:** L'elitismo garantisce che gli individui con il punteggio di fitness più alto vengano trasferiti direttamente alla generazione successiva, riducendo il rischio di perdere soluzioni promettenti durante le fasi di crossover e mutazione.
- **Stabilità evolutiva:** Il metodo SSS sostituisce soltanto una parte della popolazione a ogni generazione, permettendo un'evoluzione graduale e una maggiore stabilità nella popolazione.
- **Efficienza computazionale:** La selezione tramite SSS richiede meno operazioni rispetto a metodi come la Tournament Selection, concentrandosi su una frazione della popolazione e accelerando così i tempi di esecuzione, uno dei criteri di successo di *Brandify*.

Implementazione La selezione è gestita direttamente dal framework `Pygad` con le seguenti configurazioni:

- **parent_selection_type = "sss"**: Questo parametro indica che la selezione dei genitori avviene utilizzando il metodo Steady State Selection.
- **keep_elitism = n**: Specifica il numero di individui elitari che vengono preservati nella generazione successiva.

```
Procedura: Selezione tramite Steady State Selection ed Elitismo

Input:
- Popolazione corrente (popolazione)
- Numero di individui elitari da preservare (keep_elitism)
- Numero di genitori da selezionare per riproduzione (num_genitori)

Output:
- Nuova popolazione con genitori selezionati e individui elitari preservati

Passaggi:
1. Ordina la popolazione in base al punteggio di fitness in ordine decrescente.
2. Preserva i migliori "keep_elitism" individui nella nuova popolazione.
3. Seleziona "num_genitori" individui dalla popolazione corrente per riproduzione:
   a. Utilizza il metodo Steady State Selection (SSS) per scegliere i genitori.
   b. Aggiungi i genitori selezionati alla nuova popolazione.
4. Restituisci la nuova popolazione aggiornata.
```

Crossover

Definizione Il crossover all'interno del sistema di raccomandazione è implementato come un **crossover uniforme**. Ogni gene (bit) di un individuo ha una probabilità del 50% di essere ereditato da uno dei due genitori, determinata tramite una maschera binaria generata casualmente. Inoltre, per favorire il bilanciamento tra esplorazione e sfruttamento, è prevista una probabilità di "salto del crossover". In questo caso, il figlio viene copiato direttamente da uno dei genitori senza alcuna ricombinazione genetica, preservando integralmente le caratteristiche del genitore selezionato.

Esempio Intuitivo Consideriamo due genitori, rappresentati come sequenze di geni binari. Ogni gene indica se un prodotto è incluso (1) o escluso (0) nella raccomandazione. Supponiamo che i due genitori siano i seguenti:

$$\text{Genitore 1: } [1, 0, 1, 1, 0]$$
$$\text{Genitore 2: } [0, 1, 0, 0, 1]$$

Il **crossover uniforme** combina i geni dei due genitori utilizzando una **maschera casuale**, generata come segue:

$$\text{Maschera: } [1, 0, 1, 0, 1]$$

La maschera indica da quale genitore prendere il gene in ciascuna posizione:

- Se la maschera ha valore **1**, il gene è preso dal **Genitore 1**.
- Se la maschera ha valore **0**, il gene è preso dal **Genitore 2**.

Applicando la maschera, otteniamo il seguente figlio:

$$\text{Figlio: } [1, 1, 1, 0, 0]$$

In alcune occasioni, invece di applicare la maschera, il sistema può decidere di **saltare il crossover** con una certa probabilità. In tal caso, il figlio sarà una copia esatta di uno dei due genitori. Ad esempio:

$$\text{Figlio: } [1, 0, 1, 1, 0] \quad \text{(copia del Genitore 1)}$$

oppure

$$\text{Figlio: } [0, 1, 0, 0, 1] \quad \text{(copia del Genitore 2)}$$

Questo approccio bilancia **esplorazione** (introduzione di nuove combinazioni tramite la maschera) e **sfruttamento** (preservazione delle buone soluzioni esistenti).

Motivazioni Il crossover uniforme, con la probabilità di salto, è particolarmente vantaggioso nel contesto di *Brandify* per i seguenti motivi:

- **Diversità nei geni:** Ogni gene rappresenta l'inclusione o esclusione di un prodotto musicale. La combinazione casuale di geni consente di esplorare configurazioni uniche, migliorando la varietà delle raccomandazioni.
- **Bilanciamento genetico:** La maschera casuale assicura che i figli ereditino caratteristiche sia di artisti che di generi da entrambi i genitori, aumentando le possibilità di trovare combinazioni rilevanti.
- **Conservazione delle soluzioni ottimali:** Se uno dei genitori è già una soluzione molto buona (cioè, con un'elevata fitness), il meccanismo di salto permette eventualmente di copiarlo direttamente al posto di mescolare i geni tramite crossover. Questo evita di introdurre cambiamenti potenzialmente dannosi o inutili in una configurazione che è già vicina all'ottimale.

Implementazione Il crossover uniforme è gestito tramite il metodo `_crossover_func`, che combina i geni dei genitori con una probabilità predefinita:

```
Procedura: Crossover Uniforme

Input:
- Genitori selezionati (parents)
- Dimensione della progenie da generare (offspring_size)
- Probabilità di crossover (crossover_probability)

Output:
- Array di nuovi individui (progenie)

Passaggi:
1. Inizializza un array vuoto ‘offspring‘ per memorizzare la progenie.
2. Per ogni individuo ‘k‘ nella progenie:
   a. Seleziona due genitori: ‘parent1‘ e ‘parent2‘.
   b. Genera una maschera casuale della stessa dimensione dei genitori.
   c. Combina i geni di ‘parent1‘ e ‘parent2‘ utilizzando la maschera casuale:
      - Se la maschera è ‘True‘ in una posizione, copia il gene da ‘parent1‘.
      - Altrimenti, copia il gene da ‘parent2‘.
   d. Con una probabilità di ‘1 - crossover_probability‘,
      copia direttamente ‘parent1‘ senza eseguire il crossover.
3. Aggiungi l’individuo generato all’array ‘offspring‘.
4. Restituisci l’array ‘offspring‘.
```

Mutazione

La mutazione nell'algoritmo genetico di *Brandify* svolge un ruolo cruciale nell'introdurre variazioni casuali nella popolazione, consentendo al sistema di esplorare nuove configurazioni di soluzioni e di evitare di convergere prematuramente su soluzioni locali subottimali. La mutazione implementata nel sistema utilizza il metodo di **flip dei geni**, che agisce direttamente sulla codifica binaria degli individui.

Definizione La mutazione *flip* opera invertendo il valore dei geni passando da `0` a `1` o da `1` a `0`. La mutazione viene applicata sull'intera soluzione, ma solo su alcuni geni scelti in modo casuale, in base alla probabilità di mutazione definita.

Motivazioni

- **Adattamento Naturale all'Encoding Binario**: Poiché ogni gene rappresenta una decisione binaria (includere o escludere un prodotto musicale), il *flip* è una scelta diretta e intuitiva per introdurre variazioni.
- **Bilanciamento tra Diversità e Stabilità**:
  - *Diversità*: La mutazione introduce nuove configurazioni, ampliando lo spazio delle soluzioni esplorate.
  - *Stabilità*: La probabilità controllata di mutazione evita cambiamenti drastici che potrebbero compromettere soluzioni promettenti.
- **Efficienza Computazionale**: L'operazione è semplice e richiede solo un confronto tra valori casuali e la soglia di probabilità, garantendo rapidità e accelerando i tempi di esecuzione.

Implementazione La funzione `_mutation_func` applica la mutazione ai figli generati dal crossover:

```
Procedura: Mutazione dei geni

Input:
- Array dei figli generati dal crossover (offspring)
- Percentuale di geni da mutare (mutation_percent_genes)

Output:
- Array dei figli con i geni mutati

Passaggi:
1. Genera un array casuale delle stesse dimensioni di offspring.
2. Confronta ogni elemento dell'array casuale
   con la soglia di mutazione (mutation_percent_genes / 100).
3. Identifica i geni che soddisfano il criterio di mutazione.
4. Per ogni gene selezionato, applica il flip:
   a. Se il valore attuale è 0, diventa 1.
   b. Se il valore attuale è 1, diventa 0.
5. Restituisci l'array mutato.
```

5.4 Criteri di Arresto

L'algoritmo genetico utilizzato in *Brandify* adotta due criteri di arresto principali: **numero di generazioni** e **stagnazione**. Questi criteri sono progettati per bilanciare efficienza computazionale e qualità delle raccomandazioni, adattandosi al contesto specifico del progetto. Di seguito vengono descritti nel dettaglio:

1. **Arresto per Numero di Generazioni**
   - **Definizione**: L'algoritmo termina automaticamente al raggiungimento di un numero massimo di generazioni predefinito (`num_generations`).
   - **Motivazione**:
     - **Controllo del Tempo di Esecuzione**: Poiché *Brandify* è pensato per fornire raccomandazioni personalizzate in tempi rapidi, limitare il numero di generazioni assicura una risposta puntuale.
2. **Arresto per Stagnazione**
   - **Definizione**: L'algoritmo si arresta quando la miglior fitness della popolazione non migliora per un numero consecutivo di generazioni predefinito (`stagnation_limit`).
   - **Motivazioni**:
     - **Evitare Iterazioni Non Produttive**: Se la fitness non migliora per un numero significativo di generazioni, è probabile che l'algoritmo abbia già esplorato la maggior parte delle soluzioni rilevanti. Continuare ad iterare aggiungerebbe costi computazionali senza beneficio tangibile.
     - **Adeguatezza alla Complessità**: Un eventuale filtraggio del dataset per range di prezzo e/o determinate modalità di ricerca come vincoli impostati dall'utente nella fase di configurazione, limitano lo spazio delle soluzioni, rendendo sufficiente un numero contenuto di generazioni per individuare configurazioni ottimali, evitando stagnazioni.

3. Combinazione dei Due Criteri L'uso simultaneo di entrambi i criteri crea un sistema robusto per bilanciare **tempo di esecuzione** ed **efficacia**:

- **Ottimizzazione Flessibile**: Il numero di generazioni fornisce un "freno massimo" mentre la stagnazione si comporta come un sistema di controllo dinamico, adattando il processo di arresto alle necessità della popolazione.
- **Esecuzione Rapida**: Gli utenti si aspettano raccomandazioni in tempi brevi, e i criteri garantiscono che l'algoritmo termini entro un tempo ragionevole.
- **Bilanciamento tra Precisione, Copertura ed Efficienza**: La combinazione dei due criteri permette di esplorare lo spazio delle soluzioni senza calcoli superflui.

Conclusione La combinazione di **numero di generazioni** e **stagnazione** garantisce che l'algoritmo genetico sia rapido, adattabile ed efficace, in linea con gli obiettivi, offrendo un equilibrio tra esplorazione e sfruttamento delle soluzioni migliori.

5.5 Scelta dei Parametri

I parametri configurati per l'algoritmo genetico di *Brandify* sono stati selezionati empiricamente dopo svariati test, per bilanciare efficienza computazionale, diversità genetica e qualità delle raccomandazioni.

Parametri Principali

- **Numero di Generazioni**: `GA_NUM_GENERATIONS = 200`
- **Dimensione della Popolazione**: `GA_SOL_PER_POP = 70`
- **Numero di Genitori**: `GA_NUM_PARENTS_MATING = 25`
- **Probabilità di Crossover**: `GA_CROSSOVER_PROBABILITY = 70`
- **Probabilità di Mutazione**: `GA_MUTATION_PERCENT_GENES = 3`
- **Elitismo**: `GA_KEEP_ELITISM = 2`
- **Criterio di Stagnazione**: `GA_STAGNATION_LIMIT = 35`
- **Pesi della Fitness**:
  - `GA_AFFINITY_WEIGHTS = {"artists": 10, "genres": 10}`
  - `GA_PENALTY_WEIGHT_NON_MATCH = 10`
  - `GA_PENALTY_MISSING_RELEVANT = 15`

La penalità è maggiore per i falsi negativi (15) rispetto ai falsi positivi (10), questo riflette la scelta del **Trade-Off Precisione vs Copertura** menzionato in precedenza, essendo che ciò incentiva soluzioni che massimizzano l'inclusione dei prodotti rilevanti (minimizzando i falsi negativi), a costo di accettare una maggiore tolleranza verso i falsi positivi.

I pesi di affinità invece bilanciano artisti e generi, riflettendo la diversità musicale degli utenti.

6 Interfaccia Utente

In questa sezione vengono illustrate le principali schermate dell'interfaccia utente di *Brandify*, con una breve descrizione delle funzionalità offerte in ciascuna pagina.

6.1 Home e Log-In con Spotify

La schermata iniziale consente all'utente di accedere al sistema tramite Spotify, cliccando sull'apposito pulsante di log-in.

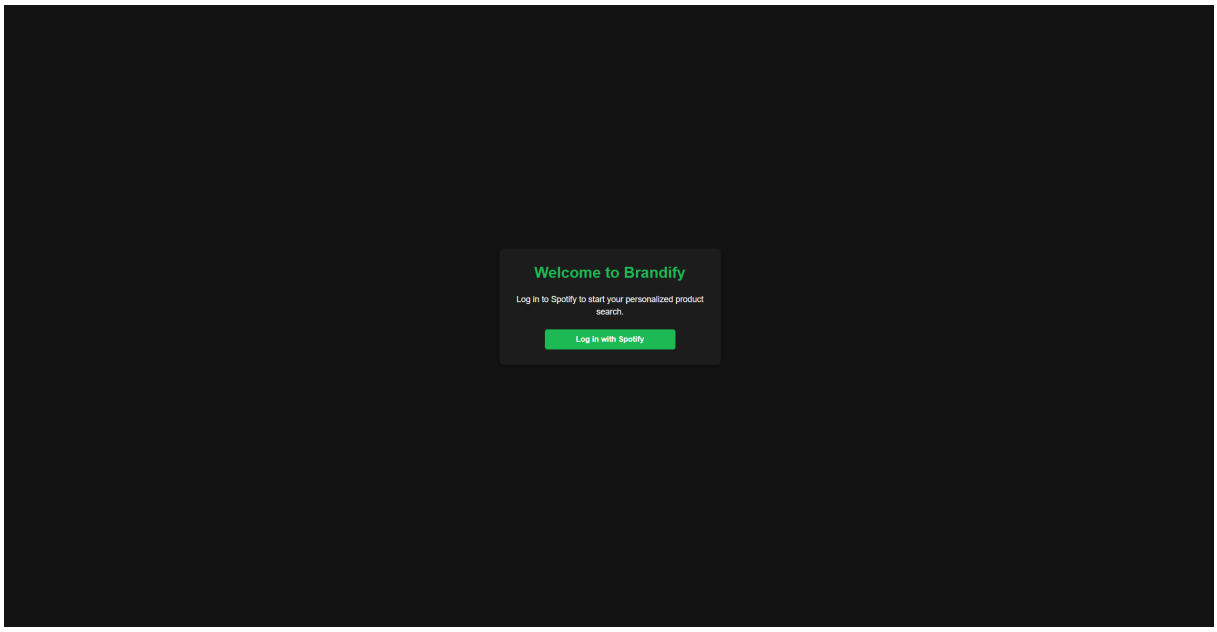

Figure 1: Schermata *Home*: interfaccia principale con il bottone per accedere tramite Spotify.

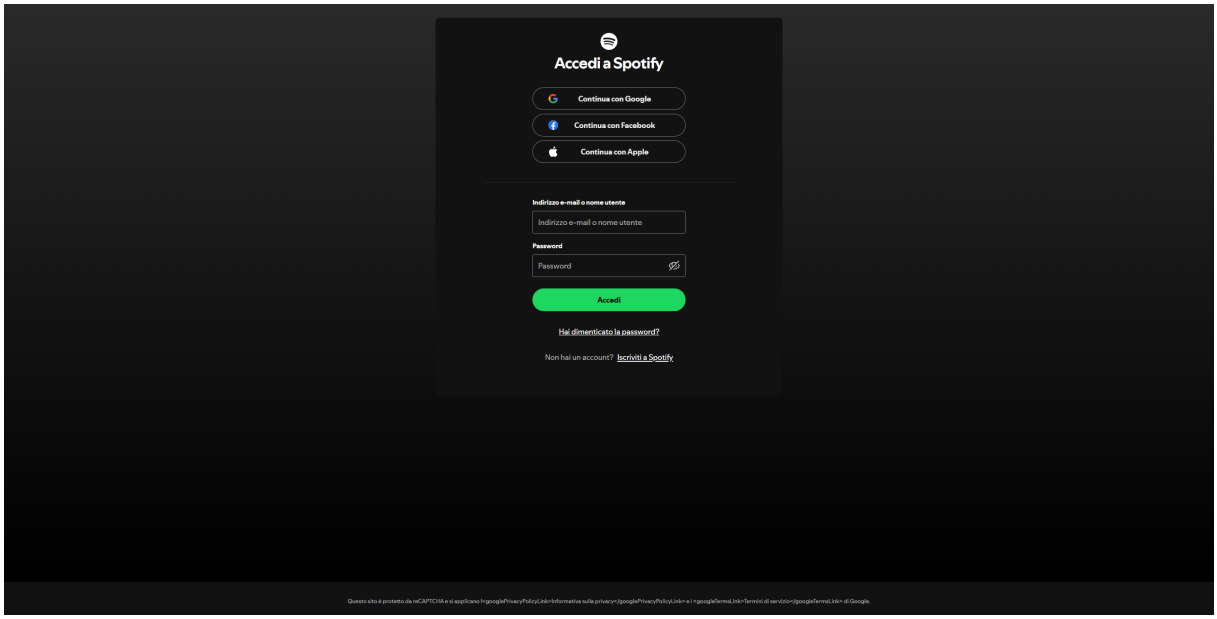

Figure 2: Schermata *Log-In*: procedura di accesso tramite Spotify.

6.2 Configurazione della Ricerca

In questa schermata, l'utente può configurare i parametri di ricerca impostando il range di prezzo desiderato e selezionando la modalità di raccomandazione (per artista, genere o bilanciata).

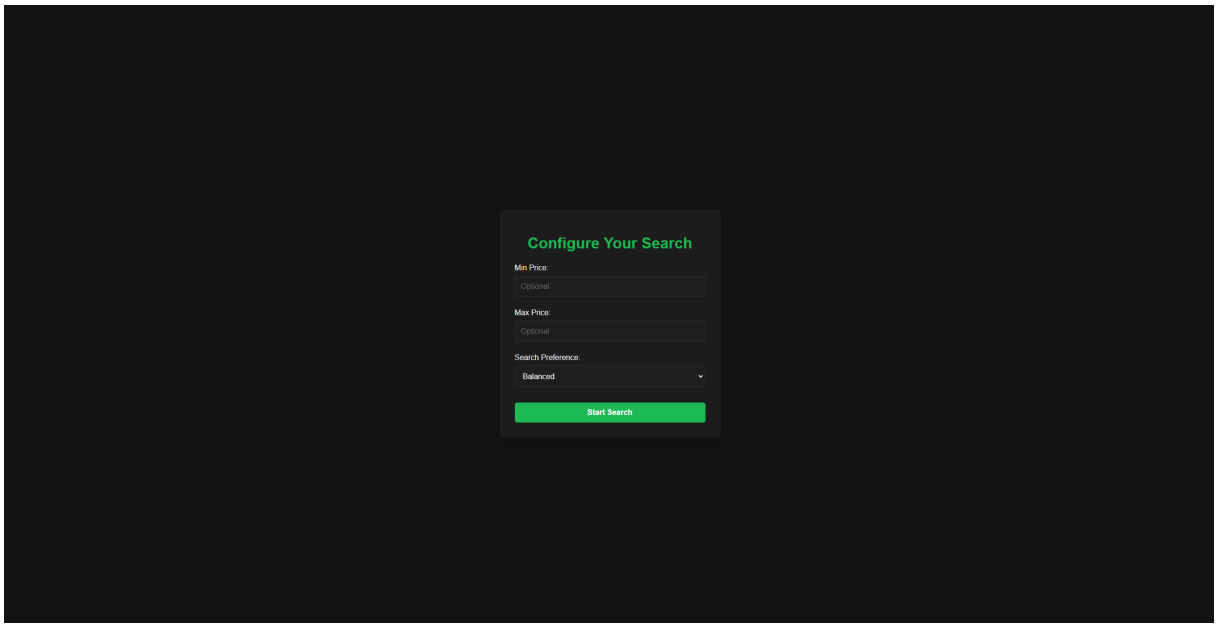

Figure 3: Schermata *Configurazione della Ricerca*: impostazione dei parametri di prezzo e modalità di ricerca.

6.3 Visualizzazione delle Raccomandazioni

La schermata finale mostra all'utente i risultati della ricerca, con i prodotti musicali raccomandati in base ai parametri configurati.

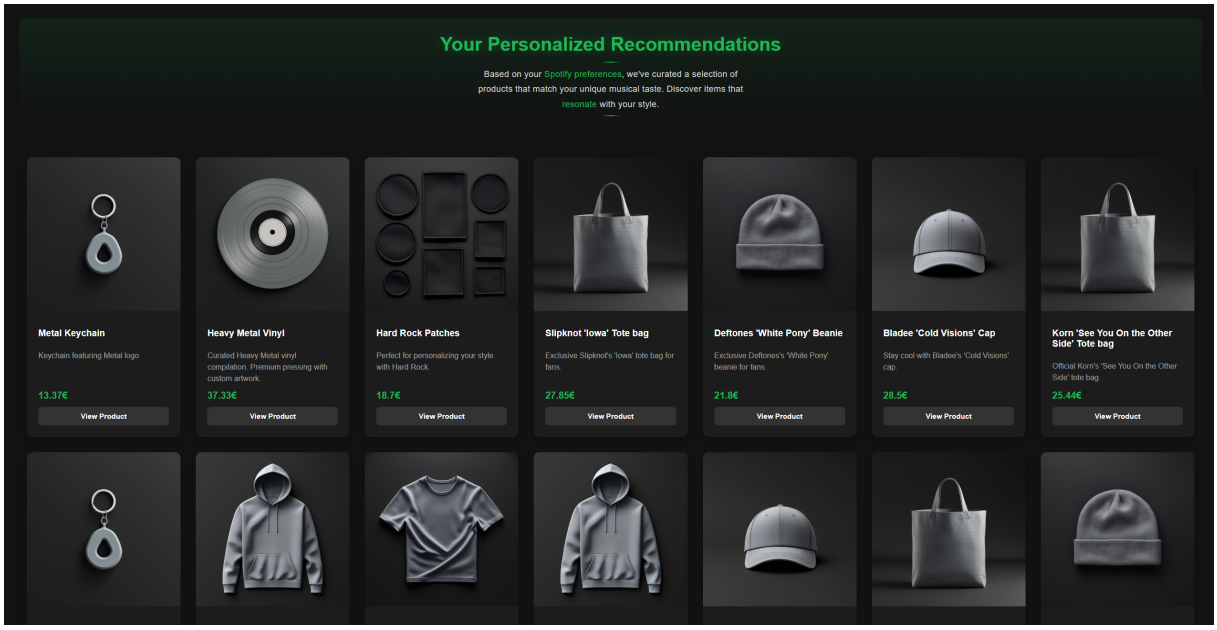

Figure 4: Schermata *Visualizzazione delle Raccomandazioni*: output delle raccomandazioni generate.

7 Valutazione e Benchmark

7.1 Metriche di Valutazione

In precedenza abbiamo parlato di metriche come Precisione e Copertura, come accennato, possono essere definite in termini di *True Positives* (TP), *False Positives* (FP) e *False Negatives* (FN):

Precisione La precisione (`precision`) misura la proporzione dei prodotti raccomandati che sono effettivamente pertinenti:

$$\text{Precisione} = \frac{TP}{TP + FP} \times 100$$

Dove:

- **TP (True Positives)**: Prodotti pertinenti correttamente raccomandati.
- **FP (False Positives)**: Prodotti non pertinenti erroneamente raccomandati.

Copertura La copertura (`coverage`) misura la proporzione dei prodotti pertinenti che sono stati inclusi nelle raccomandazioni:

$$\text{Copertura} = \frac{TP}{TP + FN} \times 100$$

Dove:

- **TP (True Positives)**: Prodotti pertinenti correttamente raccomandati.
- **FN (False Negatives)**: Prodotti pertinenti che non sono stati raccomandati.

Da notare che le metriche `precision` e `coverage` non tengono conto dei **True Negatives (TN)** (prodotti non pertinenti non raccomandati), poiché non influenzano la qualità delle raccomandazioni.

7.2 Analisi dei Risultati

L'algoritmo genetico di *Brandify* è stato testato su diversi profili musicali, ciascuno caratterizzato da artisti e generi preferiti (recenti e globali). I profili utilizzati nei test sono i seguenti:

Profilo 1 - Metal/Rock

```
PROFILE_1 = {
    "artists": ['slipknot', 'tool', 'deftones', 'alice_in_chains', 'korn', 'mudvayne',
                'radiohead', 'city_morgue', 'the_cure', 'led_zeppelin', 'the_beatles',
                'title_fight', 'death', 'opeth', 'mastodon', 'bad_religion', 'nofx',
                'cannibal_corpse'],
    "genres": ['metal', 'nu_metal', 'alternative_metal', 'rock', 'hard_rock',
               'progressive_metal', 'industrial_metal', 'classic_rock', 'psychedelic_rock',
               'rap_metal', 'indie_rock', 'grunge', 'alternative_rock', 'punk_rock'],
    "recent_artists": ['tool', 'slipknot', 'mudvayne', 'korn', 'death', 'city_morgue',
                       'radiohead', 'alice_in_chains', 'deftones', 'led_zeppelin',
                       'the_beatles', 'the_cure', 'title_fight', 'black_sabbath',
                       'nine_inch_nails', 'system_of_a_down', 'rage_against_the_machine'],
    "recent_genres": ['metal', 'nu_metal', 'alternative_metal', 'industrial_metal',
```

```
                    'rap_metal', 'rock', 'hard_rock', 'classic_rock', 'progressive_rock',
                    'grunge', 'industrial_rock', 'heavy_metal', 'thrash_metal',
                    'death_metal', 'punk']
}
```

Profilo 2 - Hip-Hop/Trap

```
PROFILE_2 = {
    "artists": ['drake', 'kendrick_lamar', 'travis_scott', 'future', 'playboi_carti',
                'lil_uzi_vert', 'lil_baby', 'kanye_west', 'jay_z', 'bad_bunny', 'uicideboy',
                'xxxtentacion', 'lil_peep', 'dmx', 'eminem', 'geolier', 'daddy_yankee'],
    "genres": ['hip_hop', 'trap', 'rap', 'underground_hip_hop', 'dark_trap', 'emo_rap',
               'cloud_rap', 'horrorcore', 'rap_metal', 'drill', 'melodic_rap'],
    "recent_artists": ['travis_scott', 'drake', 'future', 'lil_baby', 'playboi_carti',
                       'bad_bunny', 'peso_pluma', 'kendrick_lamar', 'kanye_west',
                       'lil_uzi_vert', 'uicideboy', 'xxxtentacion', 'city_morgue', 'eminem',
                       'jay_z'],
    "recent_genres": ['trap', 'hip_hop', 'rap', 'dark_trap', 'melodic_rap', 'emo_rap',
                      'drill', 'underground_hip_hop', 'horrorcore', 'mumble_rap',
                      'conscious_rap', 'alternative_rap']
}
```

Profilo 3 - Pop/Electronic

```
PROFILE_3 = {
    "artists": ['taylor_swift', 'ariana_grande', 'the_weeknd', 'billie_eilish', 'drake',
                'bad_bunny', 'bjrk', 'the_prodigy', 'gigi_dagostino', 'radiohead',
                'pino_daniele', 'bladee', 'lorde'],
    "genres": ['pop', 'electronic', 'dance', 'techno', 'house', 'trance', 'ambient',
               'experimental', 'art_pop', 'indie_pop'],
    "recent_artists": ['taylor_swift', 'ariana_grande', 'billie_eilish', 'the_weeknd',
                       'bjrk', 'the_prodigy', 'bad_bunny', 'drake', 'gigi_dagostino',
                       'pino_daniele', 'mitski'],
    "recent_genres": ['pop', 'electronic', 'dance', 'techno', 'house', 'trance',
                      'experimental', 'ambient', 'reggaeton', 'dance_pop', 'art_pop', 'rb']
}
```

Sono state testate tutte le possibili combinazioni di range di prezzo e modalità di raccomandazione (`artist`, `genre`, `balanced`). I range di prezzo sono stati definiti sulla base della distribuzione dei prezzi all'interno del dataset dei prodotti:

- **Range Stretto**: Include i valori compresi tra il primo quartile (`Q1`) e il terzo quartile (`Q3`), ovvero il 50% centrale dei dati.
- **Range Largo**: Si estende oltre il range stretto e copre l'intervallo calcolato come:

$$\text{Range Largo} = \left[Q1 - 1.5 \cdot \text{IQR},\ Q3 + 1.5 \cdot \text{IQR}\right]$$

  dove l'`IQR` (*Interquartile Range*) rappresenta la differenza tra `Q3` e `Q1`.

In breve:

- **Q1**: è il valore che separa il 25% più basso dei dati dal resto.
- **Q3**: è il valore che separa il 75% più basso dei dati dal restante 25%.
- **IQR**: rappresenta la distanza tra `Q3` e `Q1`, indicando la dispersione dei dati centrali.

I risultati di ogni combinazione sono stati registrati e analizzati, valutando le performance dell'algoritmo in termini di precisione, copertura, fitness, numero di generazioni e tempo di esecuzione, fornendo un quadro dettagliato delle raccomandazioni generate per ciascun profilo musicale.

Di seguito vengono riportati i grafici relativi a precisione e copertura per ciascun profilo:

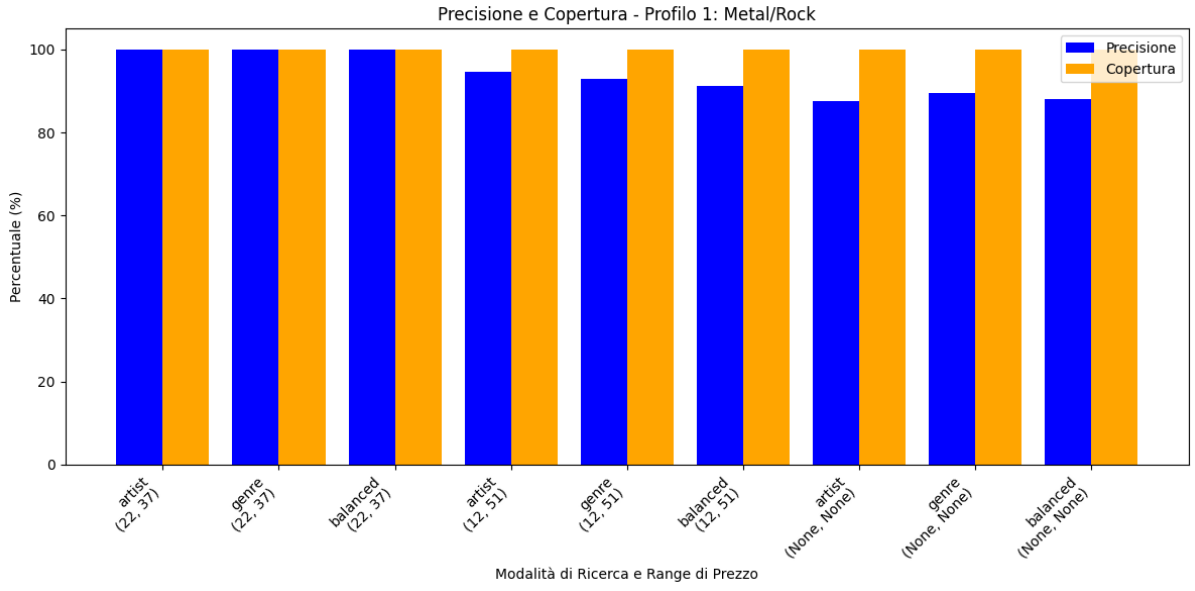

Figure 5: Precisione e Copertura - Profilo 1: Metal/Rock.

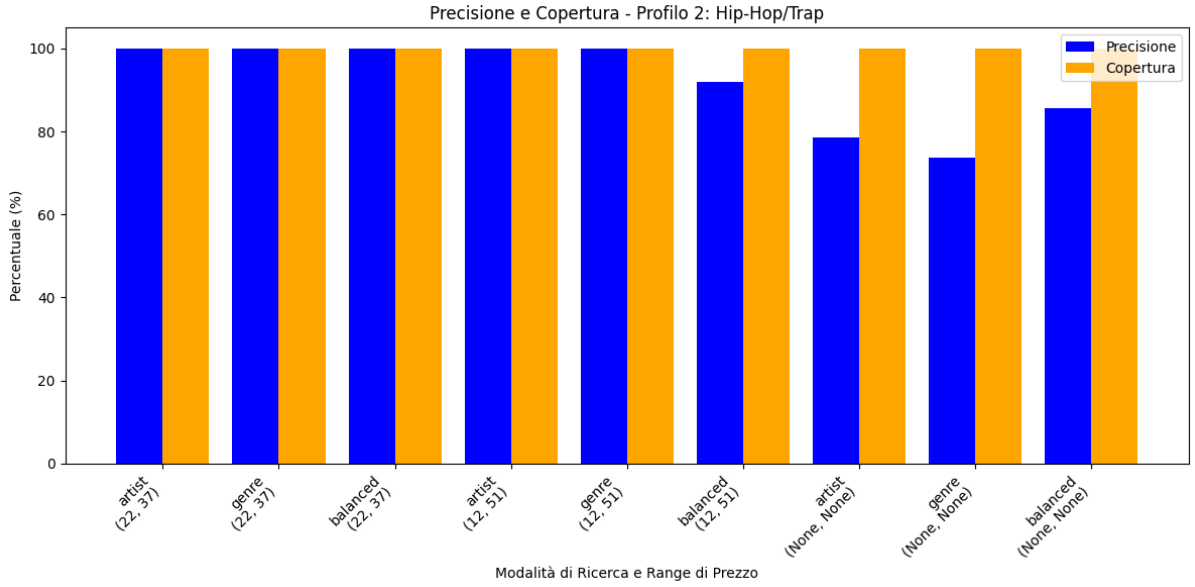

Figure 6: Precisione e Copertura - Profilo 2: Hip-Hop/Trap.

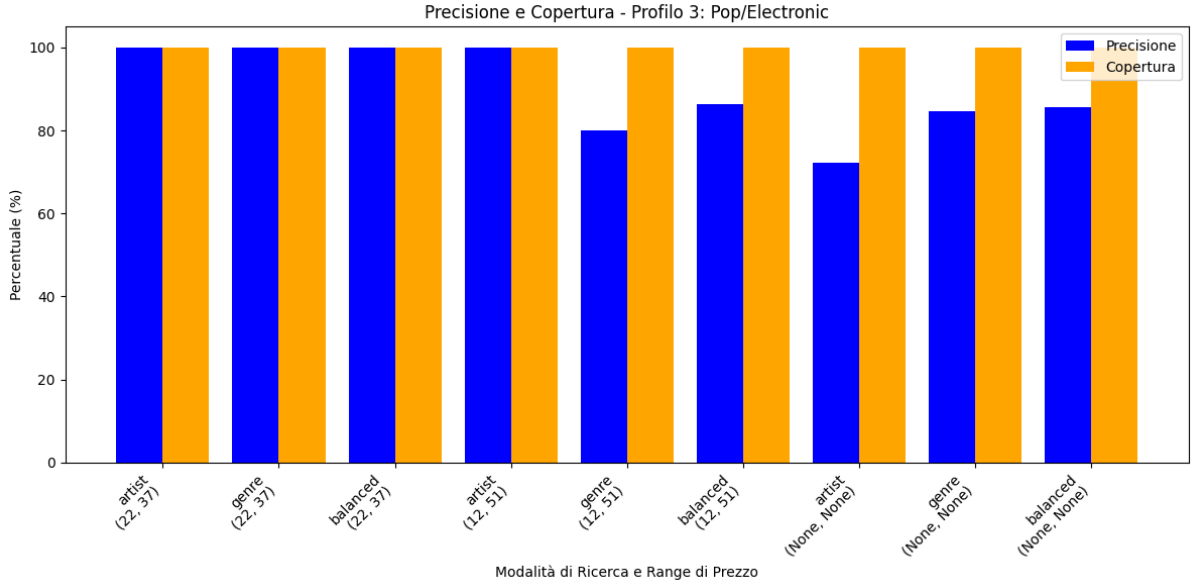

Figure 7: Precisione e Copertura - Profilo 3: Pop/Electronic.

Oltre alle metriche di precisione e copertura, sono stati analizzati i seguenti aspetti del comportamento dell'algoritmo genetico:

- Miglior punteggio di fitness ottenuto (`Best Fitness`).
- Numero di generazioni necessarie (`Generations`).
- Tempo di esecuzione (`Duration (s)`).

I risultati sono visualizzati nei grafici che seguono:

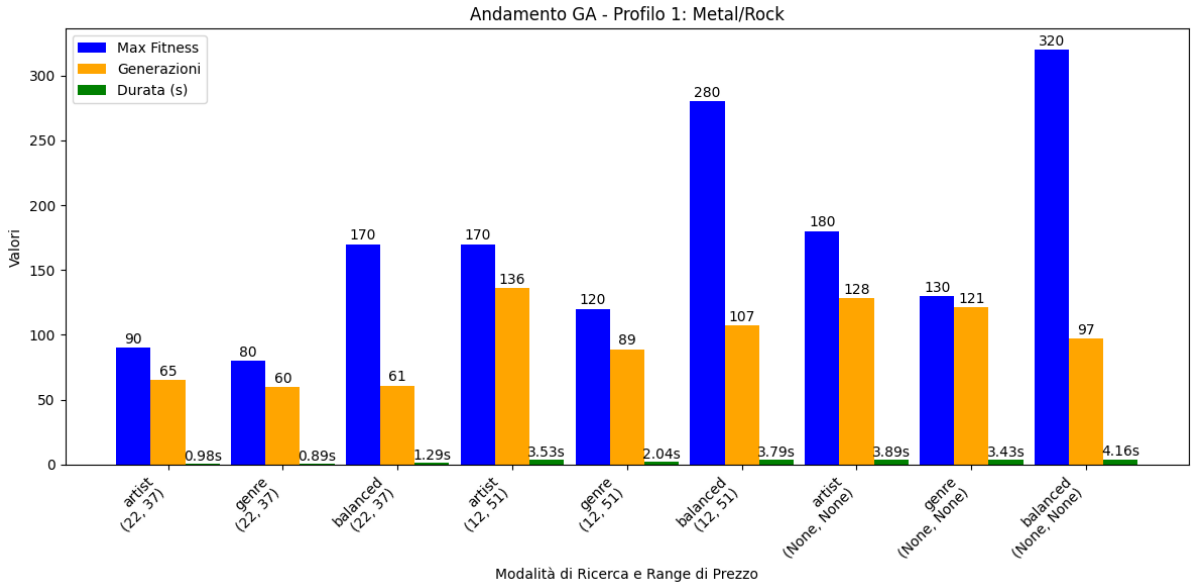

Figure 8: Prestazioni del GA - Profilo 1: Metal/Rock.

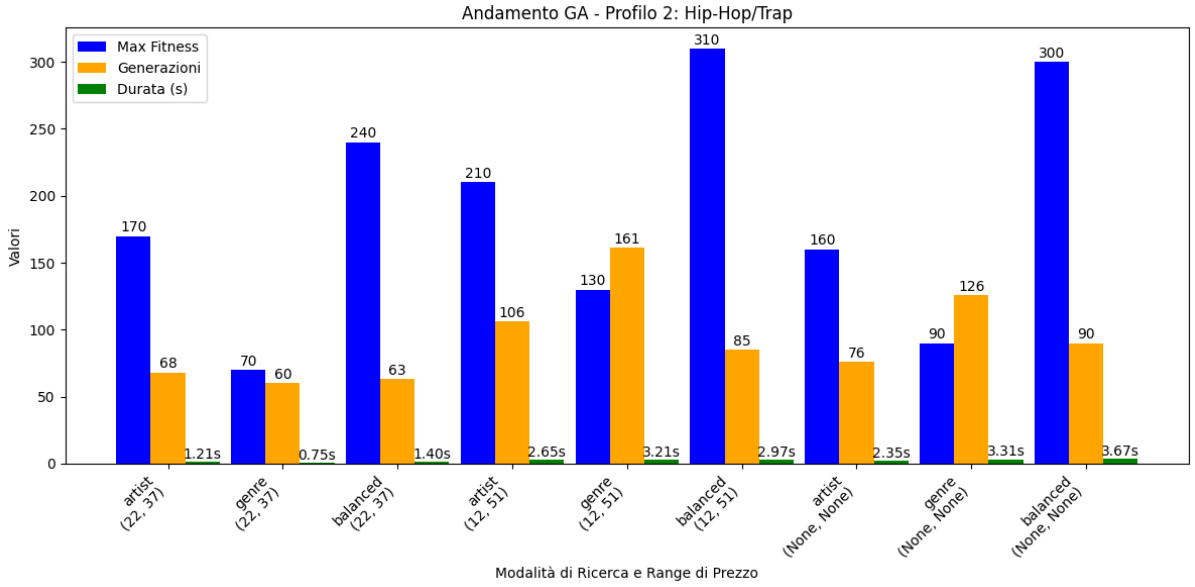

Figure 9: Prestazioni del GA - Profilo 2: Hip-Hop/Trap.

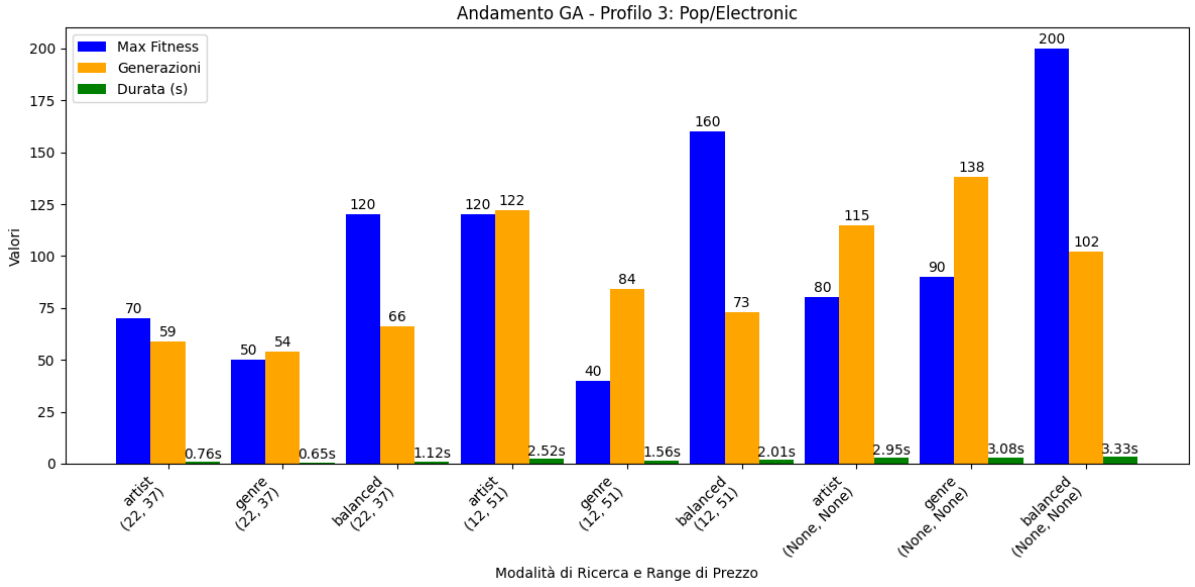

Figure 10: Prestazioni del GA - Profilo 3: Pop/Electronic.

7.3 Considerazioni sui Test

I risultati dei test mostrano che:

- La precisione tende a diminuire leggermente all'aumentare del range di prezzo, poiché vengono incluse più raccomandazioni fuori dalle preferenze specifiche.
- La copertura rimane costantemente al 100% in tutte le modalità, garantendo che tutti i prodotti pertinenti siano raccomandati.
- Il tempo di esecuzione e il numero di generazioni variano in base al range di prezzo e alla modalità di ricerca, con la modalità `balanced` che tende a richiedere più generazioni per convergere.
- Le penalità sui falsi negativi e positivi risultano ben bilanciate, migliorando la qualità complessiva delle raccomandazioni.

8 Conclusioni

8.1 Limiti del Progetto

Durante lo sviluppo di *Brandify*, sono emersi alcuni limiti che è importante sottolineare:

1. **Obiettivo Accademico:** *Brandify* è stato progettato principalmente come un "*playground*" per sperimentare gli Algoritmi Genetici studiati durante il corso, uno degli argomenti che più mi ha affascinato, applicandoli a un contesto che mi appassionasse, come quello musicale. Non si tratta quindi di un sistema progettato per essere scalabile o direttamente utilizzabile in un ambiente reale.

2. **Gestione dei Range di Prezzo:** Inizialmente avevo pianificato di integrare la gestione dei range di prezzo direttamente nella funzione di fitness. Tuttavia, questa scelta si è rivelata inefficace, poiché degradava significativamente sia la qualità delle soluzioni sia l'efficienza computazionale. Per questo motivo, (a mio dispiacere) ho deciso di utilizzare un approccio più diretto, basato sul filtraggio dei prodotti prima dell'esecuzione dell'algoritmo genetico.

3. **Dipendenza dal Dataset:** Le metriche di valutazione sono altamente sensibili alla composizione del dataset. Ad esempio, l'aggiunta anche di sole 50 nuove istanze può richiedere una completa riconfigurazione dei parametri (fitness weights, penalità, ecc.). Questo accade perché ogni modifica significativa nel dataset altera l'equilibrio tra precisione, copertura e fitness, rendendo necessaria un'ottimizzazione dei pesi per mantenere performance coerenti.

8.2 Considerazioni finali

Lo sviluppo di *Brandify* è stato per me un'esperienza estremamente formativa e stimolante. Partendo dalla semplice idea di applicare gli Algoritmi Genetici a un contesto che mi appassionasse, ho avuto l'opportunità di mettere in pratica i concetti teorici studiati durante il corso, affrontando problemi reali e trovando soluzioni creative.
È stato affascinante osservare come le scelte progettuali (ad esempio, il filtraggio dei range di prezzo o l'utilizzo delle diverse modalità di ricerca) abbiano avuto un impatto diretto sui risultati. Ho apprezzato particolarmente la libertà di sperimentare, sbagliare e migliorare iterativamente, il che mi ha aiutato a sviluppare ancora di più una mentalità orientata al problem solving.
Questa esperienza mi ha spronato a tirare fuori il mio potenziale, confermando il mio interesse verso l'applicazione di tecniche di Intelligenza Artificiale in ambiti che uniscono tecnologia e creatività.

8.3 Bibliografia e Riferimenti

Di seguito sono riportati i principali riferimenti utilizzati per lo sviluppo del progetto:

- Documentazione ufficiale di **PyGAD**: per la configurazione e l'uso dell'algoritmo genetico.
- Documentazione ufficiale di **Spotify API** e **Spotipy**: per l'integrazione con i dati musicali degli utenti.
- Documentazione ufficiale di **Last.fm API**: una risorsa fondamentale per comprendere come ottenere e sfruttare dati strutturati su generi musicali, artisti e tendenze, integrandoli efficacemente nel sistema di raccomandazione.

9 Repository del Progetto

Il codice sorgente del progetto è disponibile su GitHub al seguente link: **Repository GitHub**.