

**Sistema Intelligente di Raccomandazione e Analisi del Dominio Manga**

Caso di Studio di “Ingegneria della Conoscenza”

**Studente**: *Antonello Isabella*

**Matricola**: *737827*

**E-Mail**: *a.isabella1@studenti.uniba.it*

**URL Repository**: [*LINK*](https://github.com/AIsabella1/ICON)

AA 2024-2025

***Indice***

**Introduzione ……………………………………………………………………………………….**Obiettivi e contesto del progetto ……………………………………………………………..  
 Approccio ibrido: logico + machine learning ……………………………………………..

**Architettura del sistema ……………………………………………………………………….** Struttura del progetto e strumenti utilizzati ……………………………………………….

**Raccolta e preparazione dei dati ……………………………………………………………** Accesso API MyAnimeList (OAuth2) ……………………………………………….………..  
 Dataset generati (top\_manga.csv, mangalist.csv) ………………………………………  
 Preprocessing e costruzione dataset\_ml.csv …………………………………………….

**Knowledge Base Prolog ………………………………………………………………………..** Fatti manga/8 e lettura\_utente/5 ……………………………………………….…………….  
 Generazione da CSV ……………………………………………….…………………………….

**Motore logico simbolico ……………………………………………………………………….** Regole di raccomandazione ……………………………………………….………………….  
 Menu interattivo in Prolog ……………………………………………….……………………..

**Ontologia OWL (modulo dimostrativo) ……………………………………………………** Struttura dell’ontologia manga.owl ……………………………………………….…………  
 Ragionamento con HermiT ……………………………………………….……………………

**Machine Learning supervisionato ………………………………………………………….** Classificazione con 6 modelli ……………………………………………….………………..  
 Target Piace ……………………………………………….………………………………………..  
 *Grafici, confusion matrix, radar plot* ……………………………………………….………..

**Clustering KMeans ………………………………………………………………………………** Clustering base e ottimizzato (PCA, silhouette) …………………………………………  
 *Visualizzazione dei cluster* ……………………………………………….…………………….

**Risultati e confronto finale ……………………………………………………………………** Analisi simbolico vs statistico ……………………………………………….………………..  
 Riflessioni sui modelli ……………………………………………….…………………………..

**Problemi e soluzioni …………………………………………………………………………….** Blocco CAPTCHA MyAnimeList ……………………………………………….………………  
 Note tecniche sul flusso OAuth ……………………………………………….……………..

**Conclusioni ………………………………………………………………………………………..** Riepilogo del lavoro ……………………………………………….……………………………..  
 Estensioni future ……………………………………………….………………………………….

**Appendice ………………………………………………………………………………………….** Codice selezionato ……………………………………………….………………………………  
 Output aggiuntivo ……………………………………………….………………………………..  
 Riferimenti ……………………………………………….………………………………………….

# ***Introduzione***

Il presente elaborato descrive la progettazione e lo sviluppo di un sistema intelligente ibrido per l’analisi e la raccomandazione di contenuti nel dominio dei *manga*.  
L’obiettivo principale del progetto è combinare approcci **simbolici** e **subsimbolici** dell’intelligenza artificiale, sfruttando dati reali ottenuti tramite API pubbliche, tecniche di machine learning supervisionato e non supervisionato, e un motore logico basato su Prolog.

L’integrazione tra AI simbolica (basata su conoscenza esplicita e regole logiche, come in Prolog e OWL) e AI subsimbolica (che impiega modelli statistici per l’apprendimento dai dati) consente di realizzare un sistema **interpretabile, adattabile e alimentato da conoscenza reale**.

Il progetto si distingue per la sua architettura modulare e flessibile, che unisce componenti di raccolta dati, pre-processing, rappresentazione della conoscenza, ragionamento logico e analisi predittiva. Il dominio scelto, ovvero quello dei manga, si presta particolarmente bene a un’applicazione concreta di tecniche di raccomandazione, grazie alla sua struttura semantica ricca e alla forte componente soggettiva delle preferenze utente.

## ***Obiettivi e contesto del progetto***

Il progetto nasce con l’obiettivo di progettare e realizzare un sistema intelligente capace di analizzare e raccomandare *manga* in base alle preferenze esplicite e implicite di un utente reale. In un contesto in cui la produzione di contenuti multimediali è in costante crescita, i sistemi di raccomandazione intelligenti ricoprono un ruolo chiave per migliorare l’esperienza personalizzata degli utenti, e il dominio dei manga rappresenta un ambito ideale per sperimentare tecniche avanzate di AI grazie alla ricchezza semantica delle opere e alla forte componente soggettiva nei gusti degli utenti.

Il progetto si inserisce all’interno del programma formativo del corso ICON e si propone di applicare in modo integrato:

* Tecniche simboliche, basate su rappresentazione della conoscenza, logica deduttiva e sistemi dichiarativi (es. Prolog, OWL);
* Tecniche subsimboliche, basate su apprendimento dai dati, classificazione e clustering (es. modelli supervisionati e KMeans).

L’obiettivo didattico è quello di dimostrare la complementarità tra approcci basati su conoscenza esplicita e modelli appresi dai dati, creando un sistema raccomandatore ibrido, interpretabile e aggiornato, costruito a partire da dati reali ottenuti tramite API pubbliche (MyAnimeList).

In particolare, il progetto si propone di:

* Automatizzare il recupero di dataset personalizzati tramite autenticazione OAuth2;
* Costruire una knowledge base Prolog contenente fatti estratti dai dati reali;
* Implementare un motore logico di raccomandazione basato su regole;
* Eseguire analisi supervisionate e non supervisionate con metodi di machine learning;
* Integrare un esempio di ontologia OWL a fini dimostrativi per il ragionamento semantico;
* Produrre output visuali e report esplicativi che evidenzino le caratteristiche e le prestazioni dei modelli;
* Fornire una valutazione critica tra i diversi approcci, evidenziandone limiti e potenzialità.

Questo progetto si configura quindi come una dimostrazione pratica e completa dell’integrazione tra AI simbolica e subsimbolica, fondata su strumenti reali e replicabili, in linea con le finalità accademiche e professionali del corso.

## ***Approccio ibrido: logico + machine learning***

Il progetto si fonda su un’architettura **ibrida**, che combina due paradigmi distinti ma complementari dell’intelligenza artificiale:

1. **Approccio simbolico**:  
   Questo approccio si basa su **rappresentazioni esplicite della conoscenza** tramite simboli, relazioni e regole logiche. Nel progetto, l’intelligenza simbolica è realizzata attraverso:
   * Una **knowledge base Prolog** contenente fatti logici (es. manga/8, lettura\_utente/5);
   * Un **motore di raccomandazione logico**, che genera suggerimenti sulla base di regole dichiarative;
   * Un’**ontologia OWL** minimale, utile per dimostrare come la conoscenza semantica possa arricchire il ragionamento automatico.

Questo approccio garantisce **trasparenza e spiegabilità** delle decisioni, e consente all’utente o sviluppatore di comprendere le motivazioni alla base di una raccomandazione.

1. **Approccio subsimbolico**:  
   In contrapposizione, l’AI subsimbolica non utilizza regole esplicite, ma apprende **pattern nascosti nei dati** tramite algoritmi statistici. Nel progetto, ciò si traduce in:
   * Un modulo di **apprendimento supervisionato** per classificare i manga in base alla probabilità che piacciano all’utente, addestrato su dati reali con tecniche come AdaBoost, Random Forest, KNN e altri;
   * Un modulo di **clustering non supervisionato**, che individua gruppi di manga simili sfruttando KMeans e tecniche di riduzione dimensionale (PCA).

Questo approccio fornisce **adattabilità e potere predittivo**, permettendo al sistema di migliorare nel tempo con l’aumento dei dati.

L’integrazione di questi due approcci all’interno di un unico sistema consente di:

* Bilanciare **precisione predittiva e trasparenza logica**;
* Sviluppare un sistema di raccomandazione **più robusto e versatile**;
* Valorizzare al massimo le competenze e i contenuti affrontati nel corso.

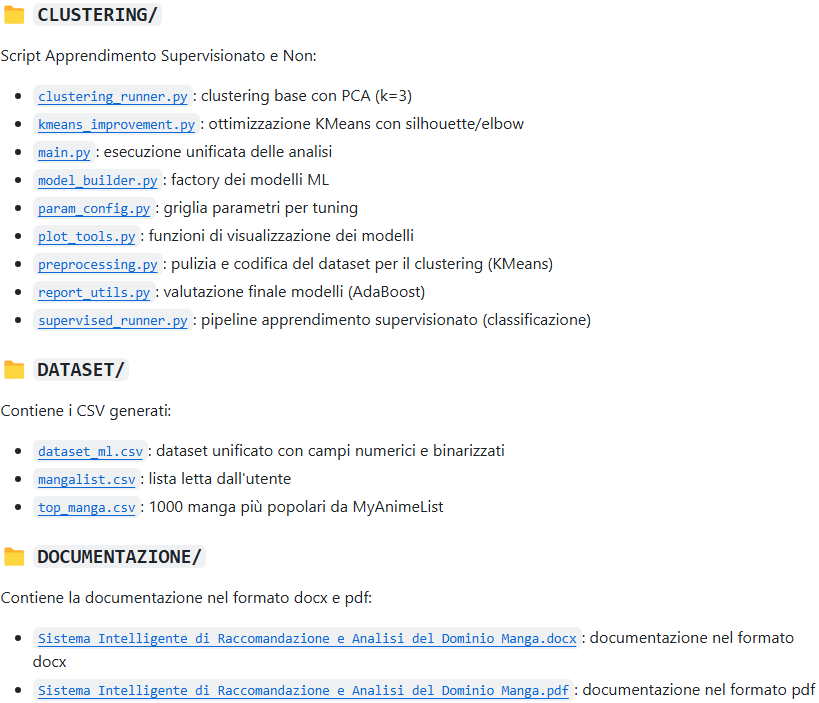
# ***Architettura del sistema***

L’architettura del progetto è **modulare**, facilmente estendibile e suddivisa in componenti indipendenti, ciascuno responsabile di una specifica fase del flusso di lavoro. Questo approccio consente di separare nettamente la logica di raccolta dati, il pre-processing, il ragionamento simbolico e l’analisi statistica, favorendo sia la chiarezza che il riuso del codice.

## ***Struttura del progetto e strumenti utilizzati***

Il progetto è stato sviluppato seguendo un approccio modulare e separato per livelli funzionali, al fine di garantire chiarezza, manutenibilità e facilità di testing. Ogni fase del sistema è supportata da una specifica cartella, con script dedicati o file generati.

**Struttura delle directory**

****

**Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.**

**Tecnologie e strumenti utilizzati**

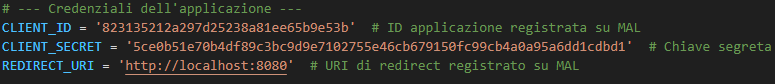
| **Categoria** | **Tecnologie** |
| --- | --- |
| Linguaggi | Python 3, Prolog, OWL/XML |
| Librerie Python | requests, pandas, scikit-learn, xgboost, matplotlib, seaborn, owlready2 |
| Reasoner OWL | [HermiT](http://www.hermit-reasoner.com/) via owlready2 |
| Ambiente logico | [SWI-Prolog](https://www.swi-prolog.org/) |
| Autenticazione API | OAuth 2.0 (MyAnimeList) |
| IDE consigliati | VS Code, PyCharm, SWI-Prolog |

# ***Raccolta e preparazione dei dati***

La fase di raccolta e preparazione dei dati è fondamentale per garantire la qualità e l'efficacia delle analisi successive. In questo progetto, l'obiettivo è ottenere un dataset affidabile e strutturato, partendo da fonti reali, per alimentare sia i moduli di apprendimento automatico che quelli di ragionamento simbolico.

## ***Accesso API MyAnimeList (OAuth2)***

Per accedere ai dati utente tramite le API di **MyAnimeList (MAL)**, è necessario seguire il flusso di autorizzazione **OAuth 2.0**. Questo protocollo consente all’utente di autorizzare un’applicazione ad accedere alle proprie risorse (es. lista dei manga letti) senza esporre le credenziali.  
Per poter utilizzare le API è necessario accedere al seguente [link](https://myanimelist.net/apiconfig) e richiedere l’ID client.



Il processo si articola in 3 fasi principali:

1. Generazione del code verifier e apertura del browser:

Lo script genera un *code\_verifier* e costruisce l’URL di autorizzazione, che viene poi aperto nel browser predefinito dell’utente:

**Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.  
Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.**

L’utente viene così reindirizzato a MyAnimeList per autorizzare l'applicazione. Dopo il consenso, MAL reindirizza al server locale con un parametro code;

1. Cattura del codice di autorizzazione tramite un server HTTP locale:

Un piccolo server Python in ascolto su localhost:8080 cattura il codice:

Immagine che contiene testo, schermata, software

Il contenuto generato dall'IA potrebbe non essere corretto.

1. Scambio del codice con un access token valido:

Dopo aver ricevuto il code, lo si invia tramite POST per ottenere l’access token:

Immagine che contiene testo, schermata

Il contenuto generato dall'IA potrebbe non essere corretto.

Con un token valido, è ora possibile accedere ai dati privati dell’utente (es. lista dei manga, punteggi, ecc.).

## ***Dataset generati (top\_manga.csv, mangalist.csv)***

L'interazione con le API di **MyAnimeList** ha portato alla generazione di due dataset principali, entrambi in formato CSV e archiviati nella cartella DATASET/. Questi file rappresentano la base informativa da cui si sviluppano sia la knowledge base simbolica che i modelli di apprendimento automatico.

**top\_manga.csv**

Questo file viene prodotto dallo script *top\_manga.py* e contiene una selezione dei **1000 manga più popolari**, ottenuti tramite richiesta pubblica alle API (/v2/manga/ranking). Per ciascun manga sono raccolti:

* ID numerico;
* Titolo;
* Generi (es. action, drama);
* Punteggio medio (media delle valutazioni della community);
* Rank generale;
* Indice di popolarità;
* Stato di pubblicazione (es. finished, publishing);
* Autori (nome e cognome).

Questo dataset ha una funzione **oggettiva**: fornisce una visione globale del panorama manga, utile per il confronto, l’analisi e la costruzione di raccomandazioni.

Primi cinque manga estratti dalla top 1000:**Immagine che contiene testo, Carattere, schermata, linea

Il contenuto generato dall'IA potrebbe non essere corretto.**

Nello script *top\_manga.py* sono presenti le funzioni *get\_top\_manga* e *save\_manga\_to\_csv*, che servono rispettivamente a estrarre i manga e a salvarli in un file in formato CSV.

**mangalist.csv**

Generato tramite user\_manga.py (o mangalist\_extended.py), questo file rappresenta la **lista personale dell’utente**. È ottenuto accedendo all’endpoint /v2/users/{username}/mangalist, previa autenticazione OAuth2. Per ciascun manga letto, vengono salvate:

* ID e titolo;
* Generi;
* Stato di lettura (es. reading, completed, plan\_to\_read);
* Punteggio assegnato dall’utente (score).

Il file consente di **modellare le preferenze personali** dell’utente, utile sia per il ragionamento simbolico (Prolog) sia per l’etichettatura supervisionata (piace / non piace) nel machine learning.

Primi dieci manga estratti dall’utente:



Nello script *user\_manga.py* sono presenti le funzioni *get\_user\_mangalist* e *save\_to\_csv*, che servono rispettivamente a estrarre i manga e a salvarli in un file in formato CSV.

## ***Preprocessing e costruzione dataset\_ml.csv***

Il file dataset\_ml.csv rappresenta la versione **pulita, integrata e numericamente codificata** dei dati provenienti da mangalist.csv e top\_manga.csv, ed è il **punto di partenza per tutte le analisi di machine learning** supervisionato e non supervisionato.

La costruzione del dataset è realizzata tramite lo script *mangalist\_extended.py*, il quale:

1. **Autentica l’utente tramite OAuth2;**
2. **Scarica la lista dei manga personali;**
3. **Arricchisce ogni voce con metadati aggiuntivi**, prelevati tramite chiamate dedicate per ogni ID manga:
   * mean: punteggio medio globale;
   * rank: posizione nella classifica;
   * popolarity: indice di popolarità nel network MyAnimeList.

Una volta ottenute le informazioni, viene avviato un processo di **pulizia e trasformazione** per produrre una struttura compatibile con gli algoritmi di apprendimento automatico:

* **Filtraggio**: vengono considerati solo i manga valutati dall’utente (score > 0);
* **Tokenizzazione dei generi**: i generi vengono convertiti in una lista di stringhe uniformate (es. "Action, Drama" → ["action", "drama"]);
* **Binarizzazione dei generi**: grazie a MultiLabelBinarizer, ogni genere diventa una colonna booleana (1 se presente, 0 altrimenti);
* **Normalizzazione numerica**: punteggio medio, rank e popolarità vengono inseriti come **variabili numeriche continue.**

Questo avviene nel file preprocessing.py per l’apprendimento non supervisionato:

Immagine che contiene testo, schermata, Carattere, software

Il contenuto generato dall'IA potrebbe non essere corretto.

E nel file supervised\_runner.py per l’apprendimento supervisionato:

Immagine che contiene schermata, testo

Il contenuto generato dall'IA potrebbe non essere corretto.

Il file risultante è un dataset **matriciale** con colonne che uniscono:

* Feature binarie (una per ogni genere);
* Feature numeriche (Punteggio Medio, Rank, Popolarità);
* Target opzionale (Punteggio Utente e/o classe binaria *Piace* per la classificazione).

Primi dieci manga estratti dell’utente:

Immagine che contiene testo, ricevuta, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

Questo dataset viene infine utilizzato per:

* **Classificare** i manga in base al gradimento personale (Piace / Non piace);
* **Clusterizzare** opere simili basandosi sui generi e sugli indicatori numerici;
* **Generare grafici** e valutazioni metriche dei modelli predittivi.

# ***Knowledge Base Prolog***

La **Knowledge Base (KB)** rappresenta la componente simbolica del sistema, costruita con il linguaggio logico **Prolog**, noto per la sua potenza nell’esprimere conoscenze e regole in forma dichiarativa.

Questa base di conoscenza raccoglie e organizza in forma strutturata le informazioni ottenute dai dataset generati, come ad esempio i manga letti dall’utente, i punteggi assegnati, i generi associati e altri attributi significativi.

L’obiettivo di questa sezione è:

* Rappresentare i dati in forma di **fatti logici** e interrogabili;
* Consentire al motore inferenziale di **ragionare simbolicamente** sulle preferenze dell’utente;
* Preparare le basi per l’attivazione di **regole di raccomandazione**, affini a un sistema esperto.

La KB viene generata automaticamente dallo script Python kb\_creator.py, che estrae i dati da top\_manga.csv e mangalist.csv, e li converte in due tipi di fatto Prolog:

* manga/8: rappresenta un manga presente nella top globale, con informazioni come titolo, generi, punteggio medio, popolarità e autori;
* lettura\_utente/5: rappresenta un manga letto dall’utente, con il relativo punteggio assegnato, stato di lettura e generi.

Questa rappresentazione simbolica permette di effettuare query flessibili e interpretabili, come ad esempio:

* *“Esistono manga premiati che l’utente non ha ancora letto?”;*
* *“Quali generi compaiono più frequentemente nella sua cronologia?”.*

La KB diventa quindi la base per il successivo **motore di raccomandazione logico**, che utilizza queste informazioni per proporre titoli affini ai gusti dell’utente o per esplorare generi ancora non letti.

## ***Fatti manga/8 e lettura\_utente/5***

La **base di conoscenza logica** è composta da due tipi principali di fatti, che rappresentano rispettivamente:

1. **Le informazioni oggettive sui manga** (estratte da top\_manga.csv);
2. **Le preferenze personali dell’utente** (estratte da mangalist.csv).

**manga/8**

Questo predicato descrive ogni manga pubblico presente nella classifica globale. La struttura è:

* ID: identificativo numerico del manga;
* Titolo: nome del manga (come atomo testuale);
* Generi: lista di atomi che rappresentano i generi (es. [action, fantasy]);
* Mean: punteggio medio globale (numero);
* Rank: posizione nella classifica generale;
* Popolarita: indice di popolarità su MAL;
* Stato: stato editoriale (es. finished, publishing);
* Autori: lista di nomi (come atomi).

**lettura\_utente/5**

Questo predicato rappresenta i manga letti (o pianificati) dall’utente, con le sue valutazioni e il suo stato personale. La struttura è:

* ID: identificativo del manga;
* Titolo: titolo normalizzato;
* Stato: stato di lettura (es. reading, completed, plan\_to\_read);
* PunteggioUtente: valore da 1 a 10 assegnato dall’utente;
* Generi: lista dei generi associati.

Questi due insiemi di fatti costituiscono l’intera base informativa interrogabile dal motore simbolico, consentendo inferenze personalizzate, raccomandazioni, raggruppamenti per generi e altro ancora.

## ***Generazione da CSV***

La costruzione automatica della base di conoscenza logica è affidata allo script Python kb\_creator.py. Questo modulo legge i file CSV generati nella fase di raccolta dati (top\_manga.csv e mangalist.csv) e li converte in una serie di **fatti Prolog** scritti nel file knowledge\_base.pl.

L’obiettivo è trasformare informazioni tabellari in **predicati logici**, che possano essere successivamente utilizzati per ragionamenti simbolici, query e raccomandazioni.

**Input**

* top\_manga.csv: contiene manga della top 1000 globale;
* mangalist.csv: contiene i manga letti e valutati dall’utente.

**Operazioni principali**

1. **Lettura dei file CSV** tramite csv.DictReader;
2. **Pulizia dei dati**: normalizzazione dei nomi (safe\_string), rimozione spazi, gestione di valori mancanti;
3. **Parsing dei campi multipli** (generi e autori come liste);
4. **Scrittura su file** in formato .pl, generando:
   * Fatti manga/8 per ogni riga di top\_manga.csv;
   * Fatti lettura\_utente/5 per ogni riga di mangalist.csv.

Lo script utilizza la funzione safe\_string() per garantire che i nomi siano compatibili con la sintassi Prolog (niente spazi o virgolette non gestite).

Questo processo rende la knowledge base **totalmente automatizzabile e sincronizzata con i dati reali**, evitando la scrittura manuale di fatti e riducendo il rischio di errori sintattici.

Ogni riga del file top\_manga.csv viene trasformata in un fatto manga/8, che contiene informazioni strutturate e interrogabili:

* ID, titolo (in formato compatibile con Prolog), lista di generi;
* Punteggio medio, rank, indice di popolarità;
* Stato di pubblicazione e autori.

Esempio:



La lista utente, invece, produce fatti del tipo lettura\_utente/5, utili per il ragionamento sulle preferenze personali:

* ID e titolo;
* Stato di lettura (es. completed, reading);
* Punteggio assegnato dall’utente;
* Generi associati all’opera.

Esempio:



# ***Motore logico simbolico***

Il **motore logico simbolico** rappresenta il cuore del componente deduttivo del sistema. Sviluppato in **Prolog**, esso consente di sfruttare la conoscenza esplicita codificata nella base di fatti per eseguire **ragionamenti interpretabili** e **raccomandazioni personalizzate**.

Attraverso un insieme di **regole logiche** definite manualmente, il motore è in grado di:

* Identificare i **generi preferiti** dell’utente;
* Consigliare manga non letti ma **affini ai gusti personali**;
* Evidenziare **opere di qualità sottovalutate**;
* Valutare la compatibilità tra manga e lettore sulla base delle frequenze di genere.

Il motore è accessibile tramite un **menu interattivo** che guida l’utente tra le varie funzionalità simboliche del sistema, offrendo una modalità di esplorazione **esplicativa e controllabile** dei contenuti raccomandati.

## ***Regole di raccomandazione***

Il motore simbolico definisce una serie di regole logiche implementate in **Prolog** all’interno del file system.pl. Ogni regola sfrutta i fatti della knowledge base (manga/8 e lettura\_utente/5) per generare **raccomandazioni personalizzate** o analizzare i comportamenti dell’utente.

Di seguito si descrivono le principali regole del sistema.

Il menu interattivo è implementato nel predicato menu/0, che presenta una serie di opzioni numerate all'utente. Ogni opzione corrisponde a una specifica funzionalità del sistema di raccomandazione.

Ecco una descrizione dettagliata di ciascuna opzione, inclusi esempi di utilizzo e output atteso:

**Visualizza i generi preferiti (ordinati per frequenza)**

Mostra i generi dei manga letti dall'utente, ordinati per frequenza decrescente.

Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

Raccoglie in ListaGeneri tutti i generi dei manga che l’utente ha effettivamente letto (escludendo quelli “plan\_to\_read” sfruttando genere\_letto(GenerePulito)).

Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

Inoltre, rimuove i duplicati ordinando alfabeticamente.

Ed infine conta quante volte ogni genere compare tra quelli letti, e li restituisce come coppie genere-numero.

Immagine che contiene testo, Carattere, schermata, linea

Il contenuto generato dall'IA potrebbe non essere corretto.

Ordina la lista Frequenze in ordine decrescente rispetto alla seconda componente (il numero), fornendo i generi ordinati per preferenza.

Esempio di output:

Immagine che contiene testo, schermata, Carattere, design

Il contenuto generato dall'IA potrebbe non essere corretto.

**Consiglia 5 manga basati sui tuoi gusti più frequenti (random)**

Suggerisce **manga non ancora letti**, appartenente ai **generi preferiti** dell’utente letti almeno dieci volte, ma **scegliendolo in modo casuale** tra quelli compatibili.

Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

Calcola i generi letti più frequentemente dall’utente, per poi scorre uno alla volta i generi preferiti. Crea una lista di tutti i manga con quel genere **non ancora letti**, salvandone ID e titolo. Seleziona **casualmente** i manga tra quelli trovati. Rende il titolo leggibile (rimuove underscore ecc.).

Esempi di output con la stessa KB:

Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

**Consiglia 5 manga di qualità ma poco popolari**

Trova manga non letti con voto medio >= 8 e popolarità (valore numerico alto = poco noti) superiore a 1500.

**Immagine che contiene testo, Carattere, schermata, numero

Il contenuto generato dall'IA potrebbe non essere corretto.**

Ottiene punteggio medio e popolarità per ogni manga. Seleziona solo manga **ben valutati (mean ≥ 8)** e **non troppo popolari** (pop > 1500) e che non siano stati già letti dall’utente.

Esempi di output con la stessa KB:

**Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.**

**Consiglia 5 manga dalla tua lista "plan\_to\_read" con generi familiari**

Suggerisce manga presenti nella lista "plan\_to\_read" dell'utente che condividono almeno un genere con i manga già letti.

Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

Scorre tutti i manga che l’utente ha nel suo “plan to read”, creando una lista di generi già letti dall’utente, escludendo i “plan\_to\_read”. Se ci sono generi in comune tra quelli già letti e quelli del manga nel plan-to-read, allora lo consiglia.

Esempi di output con la stessa KB:

Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

**Consiglia 5 manga premiati compatibili con i tuoi generi preferiti**

Raccomanda manga non letti che sono "award\_winning" e che contengono almeno un genere tra quelli preferiti.

Immagine che contiene testo, Carattere, schermata

Il contenuto generato dall'IA potrebbe non essere corretto.

Scorre i generi preferiti dell’utente, cerca manga che abbiano sia quel genere, sia il tag award\_winning. Consiglia solo se l’utente non li ha già letti.

Esempi di output con la stessa KB:

Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

**Consiglia 5 manga con almeno 2 generi completamente nuovi per te**

Suggerisce manga non letti con almeno due generi mai letti. Serve per esplorare novità.

Immagine che contiene testo, schermata, menu, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

Raccoglie tutti i generi esistenti nei manga, rimuove duplicati e confronta l’elenco con quelli già letti. Calcola la differenza: generi mai letti e restituisce manga **composti solo da generi nuovi**.

Esempi di output con la stessa KB:

Immagine che contiene testo, Carattere, schermata, linea

Il contenuto generato dall'IA potrebbe non essere corretto.

**Consiglia 5 manga che combinano generi noti e generi mai letti**

Trova manga che combinano almeno un genere mai letto con almeno un genere già letto (anche se visto una sola volta).

Serve per espandere i gusti restando in parte nella propria comfort zone.

**Immagine che contiene testo, schermata, menu, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.**

Molto simile alla precedente, ma il manga deve contenere **almeno un genere già letto** e **almeno uno mai letto**.

Esempi di output con lo stesso KB:

Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

**Valuta la compatibilità di una lista di generi rispetto alle tue preferenze**

Dato un elenco di generi, li confronta con quelli ordinati per frequenza e valuta quanto sono compatibili con i gusti dell'utente.

Immagine che contiene testo, schermata, software, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

Divide i generi ordinati in 3 fasce di preferenza, assegnando punteggi 2 (molto compatibile), 1 (abbastanza) o 0 (non compatibile) in base alla posizione del genere. Calcola la media del punteggio tra i generi forniti e stampa il giudizio finale in base al valore della media.

Esempi di output con la stessa KB:

Immagine che contiene testo, schermata, Carattere, linea

Il contenuto generato dall'IA potrebbe non essere corretto.

Immagine che contiene testo, schermata, Carattere, linea

Il contenuto generato dall'IA potrebbe non essere corretto.

Immagine che contiene testo, schermata, Carattere, linea

Il contenuto generato dall'IA potrebbe non essere corretto.

## ***Menu interattivo in Prolog***

Il sistema di raccomandazione logico-simbolico dispone di un **menu testuale interattivo**, implementato in Prolog nel file system.pl, che permette all’utente di esplorare i dati e ottenere suggerimenti intelligenti in base alla propria esperienza di lettura.

**Funzionamento**

Per avviare il menu, è necessario consultare i file knowledge\_base.pl e system.pl, quindi digitare il comando menu. nella console Prolog:  
 Immagine che contiene testo, Carattere, schermata

Il contenuto generato dall'IA potrebbe non essere corretto.

Dopodiché apparirà il seguente menu:

Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

A quel punto, viene mostrata una lista numerata di opzioni, ciascuna corrispondente a una regola di raccomandazione o analisi. Dopo aver selezionato un’opzione (inserendo il numero corrispondente), il sistema esegue la regola associata e mostra il risultato.

**Gestione delle scelte**

Ogni opzione è gestita da un predicato esegui\_scelta(N) dove N è il numero digitato. Esempio:

Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

Questo codice:

* Richiama la regola raccomanda\_random/1;
* Estrae i primi 5 risultati casuali;
* Li stampa a video;
* Richiama nuovamente il menu.

# ***Ontologia OWL (modulo dimostrativo)***

In questa sezione si presenta un **esempio dimostrativo** di rappresentazione della conoscenza tramite **ontologia OWL**, con l’obiettivo di **arricchire il progetto con un approccio semantico simbolico**.  
L’uso dell’ontologia **non è parte integrante del sistema di raccomandazione**, ma è stato introdotto **a scopo espositivo e completamento formativo**, per dimostrare l’utilizzo di tecnologie legate al **Semantic Web** e alla **logica descrittiva (Description Logics)**.

L'ontologia definisce le seguenti entità nel dominio dei manga:

* la classe Manga;
* il genere Seinen come sottoclasse o istanza di un concetto generico;
* la proprietà hasGenre;
* la proprietà hasAward;
* e l'individuo Berserk, classificato come AwardWinning.

Il ragionamento è stato implementato tramite **Owlready2** in Python e l'invocazione del **reasoner Hermit**, che consente di inferire automaticamente relazioni semantiche.

## ***Struttura dell’ontologia manga.owl***

L’ontologia manga.owl è un **modulo dimostrativo** sviluppato per rappresentare in modo simbolico alcune proprietà semantiche del dominio dei manga. È modellata secondo i principi di OWL (Web Ontology Language) e rappresenta una **base concettuale minimale**, pensata per illustrare l’integrazione tra ragionamento logico e conoscenza esplicita.

**Classi principali**

* **Manga**: classe generica che rappresenta un'opera manga.
* **Seinen**: genere narrativo rivolto a un pubblico adulto, modellato come concetto (e.g. può essere trattato come individuo o sottoclasse).
* **AwardWinning**: concetto che indica manga premiati, utilizzato come oggetto della proprietà hasAward.

**Proprietà (ObjectProperty)**

* **hasGenre**: collega un'istanza di Manga a uno o più generi (es. Seinen).
* **hasAward**: collega un'istanza di Manga a un concetto premiale (es. AwardWinning).

Esempio nel file manga.owl:

Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

Esempio output del codice:

Immagine che contiene testo, Carattere, schermata, design

Il contenuto generato dall'IA potrebbe non essere corretto.

## ***Ragionamento con HermiT***

Il ragionamento logico sulle ontologie è stato implementato tramite **HermiT**, un motore compatibile con OWL 2, integrato in Python grazie alla libreria Owlready2.  
L’obiettivo è mostrare un esempio di **deduzione automatica di conoscenza** partendo da relazioni dichiarative esplicite.

Codice per estrarre i manga premiati:

Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

L’utilizzo di Owlready2 in combinazione con il reasoner **HermiT** consente di eseguire **inferenze logiche automatiche** a partire da un’ontologia espressa in OWL. Questo permette di **derivare conoscenza implicita**, rendendo più potente e flessibile il modello simbolico.

Nel contesto del progetto, dopo aver caricato l’ontologia manga.owl e attivato il reasoner, si può ottenere un **output inferenziale** mediante semplici interrogazioni sugli individui della classe Manga.

Esempio output del codice:



# ***Machine Learning supervisionato***

In questa sezione viene descritta l’applicazione di tecniche di apprendimento supervisionato con l’obiettivo di prevedere il gradimento di un manga da parte dell’utente, sulla base di caratteristiche sia numeriche sia semantiche.  
L’obiettivo è costruire un **classificatore binario** in grado di stabilire se un manga possa piacere (1) o meno (0) a un determinato utente, utilizzando come feature i **generi** associati all’opera e alcune metriche globali estratte dal dataset (punteggio medio, rank e popolarità).

Questa fase rappresenta la componente **subsimbolica** del sistema, in cui l’apprendimento dai dati storici dell’utente avviene tramite modelli statistici che cercano di generalizzare le sue preferenze. Il risultato è una base predittiva utile sia per la raccomandazione diretta, sia per l'integrazione con il motore logico-simbolico.

L’intero processo si articola nei seguenti passaggi:

* **Costruzione del dataset supervisionato** a partire da dataset\_ml.csv;
* **Binarizzazione delle etichette di genere** tramite MultiLabelBinarizer;
* **Definizione della variabile target** Piace, 1 per manga con punteggio assegnato ≥ 7, 0 altrimenti;
* **Addestramento e confronto di diversi algoritmi di classificazione**, tra cui:
  + Decision Tree;
  + Random Forest;
  + AdaBoost;
  + K-Nearest Neighbors (KNN);
  + Naive Bayes;
  + XGBoost.
* **Valutazione delle performance tramite cross-validation** (5-fold), utilizzando le metriche:
  + Accuracy;
  + Precision;
  + Recall;
  + F1-score.

L’output dell’analisi comprende **grafici comparativi**, **matrici di confusione** e **radar plot** che evidenziano punti di forza e limiti di ciascun classificatore.  
Questa fase permette di individuare i modelli più adatti al profilo dell’utente, fornendo una base predittiva fondamentale per le successive fasi di raccomandazione e integrazione simbolica.

## ***Classificazione con 6 modelli***

### **Decision Tree**

Il **Decision Tree** è un algoritmo supervisionato che rappresenta le decisioni tramite una struttura ad albero, in cui:

* Ogni **nodo interno** rappresenta un attributo (una feature),
* Ogni **ramo** rappresenta una condizione/valore,
* Ogni **foglia** rappresenta un output (in questo caso: "piace" = 1 o "non piace" = 0).

L'algoritmo costruisce l’albero in modo ricorsivo, cercando a ogni passo di dividere il dataset in sottoinsiemi il più omogenei possibile, minimizzando l’impurità (ad es. tramite Gini o Entropia).

**Iperparametri principali**:

* **max\_depth**: la profondità massima dell’albero. Un valore troppo alto può portare a **overfitting**, uno troppo basso a **underfitting**.
* **min\_samples\_leaf**: numero minimo di campioni richiesti in un nodo foglia. Aumentarlo migliora la generalizzazione.
* **splitter**: strategia di suddivisione ("best" o "random") durante la costruzione dell’albero.

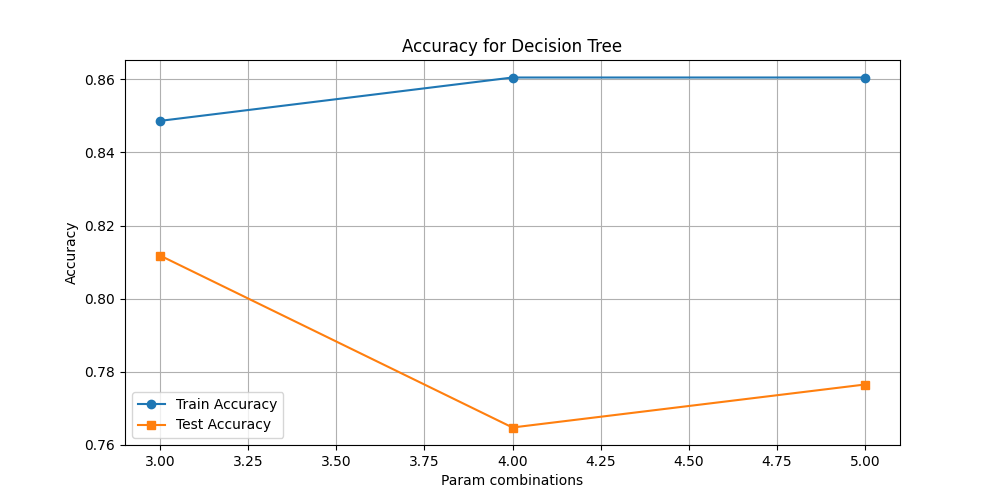
**Motivazioni della scelta del modello**  
Il Decision Tree è stato selezionato come primo approccio supervisionato per via della sua semplicità, interpretabilità e capacità di rappresentare decisioni gerarchiche in modo visivo. Inoltre, permette di studiare in modo diretto il fenomeno dell’overfitting tramite iperparametri come max\_depth e min\_samples\_leaf. Il modello è stato utilizzato anche come baseline da confrontare con metodi più complessi come Random Forest e XGBoost.

**Grafico delle Accuratezze – Overfitting contenuto**

Dopo aver introdotto vincoli come max\_depth=3–5 e min\_samples\_leaf≥10, il comportamento del modello si è stabilizzato:

* L’**accuracy sul training set** non raggiunge più il 100%, segno che il modello non sta più memorizzando i dati.
* L’**accuracy sul test set** si mantiene stabile (~76–78%) senza più decrescere drasticamente.

Questo evidenzia che l’**overfitting è stato efficacemente contenuto**, pur preservando una buona capacità predittiva.



Nel grafico si osservano le accuratezze su training set (linea blu) e test set (linea arancione) al variare di diverse combinazioni di iperparametri, in particolare max\_depth e min\_samples\_leaf.

**Osservazioni:**

* La **train accuracy** resta alta (~0.85–0.86), ma costante, suggerendo che anche con parametri restrittivi il modello apprende efficacemente dai dati.
* La **test accuracy** varia leggermente (~0.76–0.81), ma non presenta cali drastici, dimostrando una buona generalizzazione.
* Il punto debole (param 4) mostra un minimo locale, ma il recupero al punto successivo indica che il modello tende a stabilizzarsi.

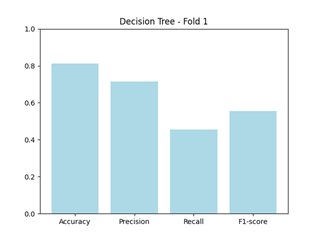
**Conclusione**: il modello non mostra più la tipica divergenza tra training e test dovuta all’overfitting, ma evidenzia un comportamento regolare e sotto controllo grazie alla regolarizzazione imposta.

**Valutazione tramite Cross Validation (5 Fold)**

È stata eseguita una validazione incrociata con 5 suddivisioni stratificate. Di seguito l’analisi per ciascun fold:

**Fold 1**

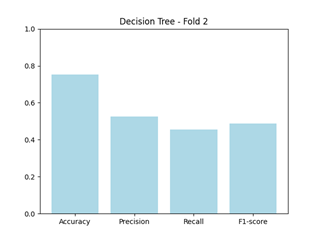
* Accuracy: ~0.81
* Precision: ~0.72
* Recall: ~0.46
* F1-score: ~0.55



Il modello mostra una **precisione elevata**, il che indica una buona capacità di evitare falsi positivi. Tuttavia, con un recall al 46%, tende a **non identificare tutti i manga potenzialmente graditi**, classificandone alcuni erroneamente come "non piace".

**Fold 2**

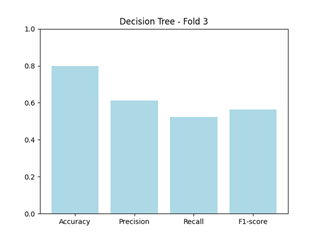
* Accuracy: ~0.75
* Precision: ~0.52
* Recall: ~0.46
* F1-score: ~0.48



Prestazioni leggermente inferiori al fold 1, ma ancora **coerenti**. L’F1-score è stabile, segno che il modello mantiene un comportamento equilibrato. È possibile che in questo fold ci sia una distribuzione meno favorevole tra le classi.

**Fold 3**

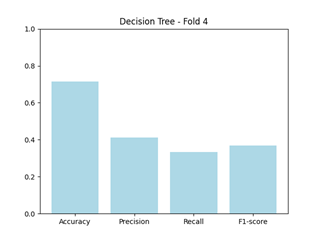
* Accuracy: ~0.80
* Precision: ~0.62
* Recall: ~0.52
* F1-score: ~0.56



Uno dei migliori fold. Il recall supera il 50%, indicando che il modello è riuscito a **identificare correttamente una buona quantità di manga apprezzati**. Le metriche sono ben bilanciate, con una precisione stabile e un F1-score robusto.

**Fold 4**

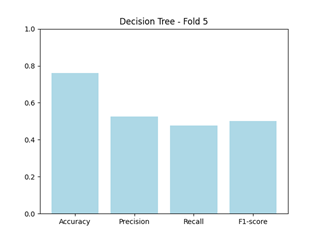
* Accuracy: ~0.71
* Precision: ~0.41
* Recall: ~0.31
* F1-score: ~0.36



Il fold più debole. L’accuratezza scende al minimo, e il recall al 31% indica che molti manga positivi sono stati **ignorati** dal classificatore. Potrebbe derivare da uno sbilanciamento delle classi o da dati atipici in questa partizione.

**Fold 5**

* Accuracy: ~0.74
* Precision: ~0.52
* Recall: ~0.48
* F1-score: ~0.50



Fold intermedio e stabile. Le metriche sono coerenti con il resto del modello. Si nota un buon compromesso tra **conservatorismo (precisione)** e **copertura (recall)**.

**Decision Tree – Analisi dei Risultati**

Il modello Decision Tree ha mostrato una **performance stabile ma contenuta**, con una media accuracy del **76.8%** e un F1-score medio pari a **0.495**.  
Le metriche rivelano una discreta capacità del classificatore di distinguere tra manga apprezzati e non apprezzati, ma con un **equilibrio fragile tra precisione (0.558) e recall (0.448)**.

Nell’analisi dei singoli fold, si è osservata una certa **variabilità**, con:

* Fold 1 e 3 che mostrano buone prestazioni complessive;
* Fold 4 in evidente difficoltà, con recall molto basso (0.333);
* Fold 5 più bilanciato, ma privo di eccellenze.

Il recall inferiore al 50% nella maggior parte dei casi evidenzia una **difficoltà del modello a individuare tutti i manga apprezzati**, pur mantenendo una precisione accettabile.  
L’adozione di tecniche di regolarizzazione (es. max\_depth limitato, min\_samples\_leaf ≥ 10) ha permesso di **contenere l’overfitting**, come confermato dal grafico delle accuratezze: le linee di training e test si mantengono vicine e stabili.

Nel complesso, il Decision Tree si configura come:

* **Interpretabile e semplice da comprendere**;
* **Conservativo**, ma penalizzato da una copertura non ottimale;
* **Adatto come baseline** o modulo interpretativo, più che come classificatore finale.

### ***Random Forest***

Il **Random Forest** è un algoritmo supervisionato basato su un insieme di alberi decisionali (ensemble di Decision Tree). Ogni albero viene addestrato su un sottoinsieme casuale del dataset (bootstrapping) e, durante la crescita dell’albero, viene selezionato un sottoinsieme casuale di feature per decidere le divisioni interne (bagging + feature randomness).  
Il risultato finale è ottenuto aggregando le predizioni dei singoli alberi (voto di maggioranza per classificazione).

Questa strategia:

* Riduce l’overfitting tipico del singolo Decision Tree;
* Aumenta la robustezza del modello;
* Fornisce stime più stabili e generalizzabili.

**Iperparametri principali**

* n\_estimators: numero di alberi nella foresta. Un numero maggiore migliora la stabilità, ma aumenta il tempo di calcolo.
* max\_depth: profondità massima di ogni albero. Usata per contenere la complessità dei singoli alberi.
* min\_samples\_leaf: numero minimo di campioni richiesti in un nodo foglia.
* max\_features: numero di feature considerate per ogni split (sqrt, log2, ecc.).
* class\_weight: utile per bilanciare classi sbilanciate, es. balanced.

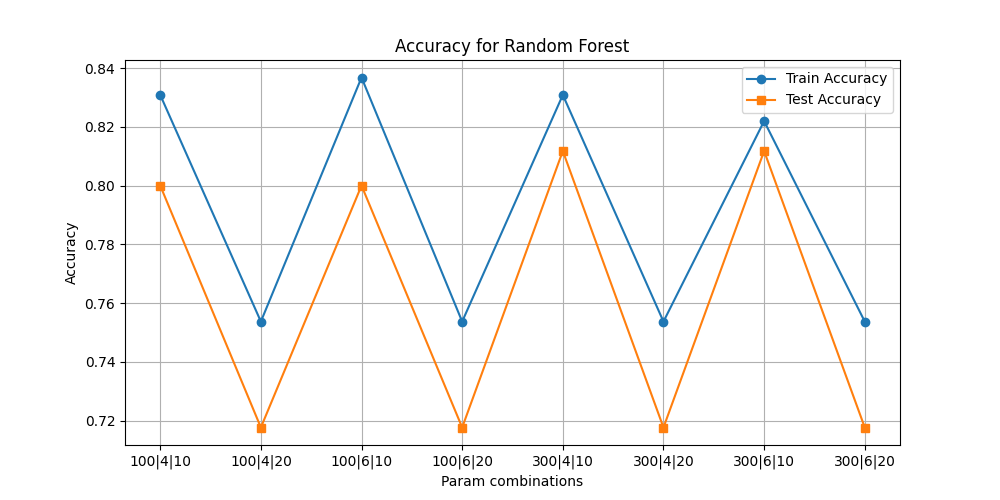
**Motivazioni della scelta del modello**

La Random Forest è stata selezionata per le sue proprietà robuste: riduce l’overfitting tipico del singolo albero, gestisce bene i dataset con molte feature e fornisce buone prestazioni anche senza una pesante ottimizzazione. È ideale come secondo passo dopo il Decision Tree, per valutare quanto un ensemble possa migliorare la stabilità e la capacità predittiva del sistema.

**Grafico delle Accuratezze – Fluttuazioni e stabilità**

Nel grafico delle accuratezze al variare delle combinazioni iperparametriche si osserva:

* La **train accuracy** (linea blu) si mantiene sempre alta (tra 0.75 e 0.84), con picchi attorno all'83–84% per configurazioni meno regolarizzate.
* La **test accuracy** (linea arancione) oscilla invece tra ~0.71 e ~0.81, con valori che mostrano minimi e massimi alternati, ma senza cadute drastiche.
* Le coppie di parametri max\_depth=4 e min\_samples\_leaf=20 tendono a generare i minimi, suggerendo una leggera **sottoconfigurazione** (underfitting) in quelle combinazioni.
* Tuttavia, l’andamento regolare e simmetrico tra i valori bassi e alti dei parametri indica che l’algoritmo è **stabile e ben controllato**, senza segnali gravi di overfitting.



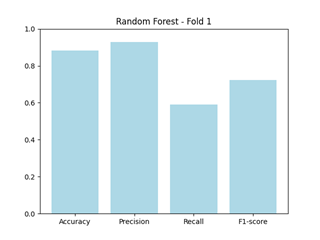
**Conclusione**: la Random Forest mantiene un buon equilibrio tra bias e varianza. La generalizzazione è efficace e l’oscillazione della test accuracy indica sensibilità agli iperparametri, ma entro limiti contenuti.

**Valutazione tramite Cross Validation (5 Fold)**

È stata eseguita una validazione incrociata con 5 suddivisioni stratificate. Di seguito l’analisi per ciascun fold:

**Fold 1:**

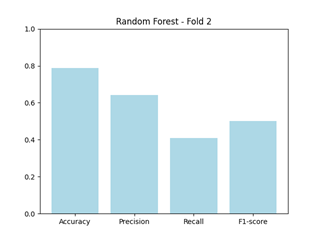
* Accuracy: 0.882
* Precision: 0.929
* Recall: 0.591
* F1-score: 0.722



Questo fold mostra le migliori prestazioni complessive, con un'alta precisione che indica una forte capacità del modello di evitare falsi positivi. Il recall del 59.1% suggerisce che il modello riesce a identificare correttamente una buona parte dei manga apprezzati.

**Fold 2:**

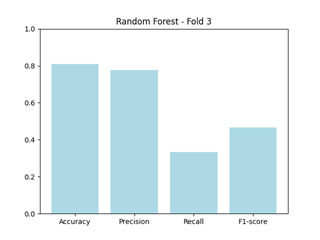
* Accuracy: 0.788
* Precision: 0.643
* Recall: 0.409
* F1-score: 0.500



In questo fold, la precisione rimane elevata, ma il recall diminuisce, indicando che il modello è più conservativo nel classificare un manga come "piace", potenzialmente escludendo alcuni manga che l'utente potrebbe gradire.

**Fold 3:**

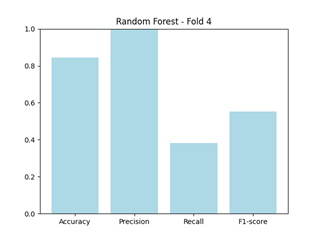
* Accuracy: 0.810
* Precision: 0.778
* Recall: 0.333
* F1-score: 0.467



Qui, la precisione è alta, ma il recall è il più basso tra tutti i fold, suggerendo che il modello è molto selettivo e potrebbe non riconoscere molti manga apprezzati.

**Fold 4:**

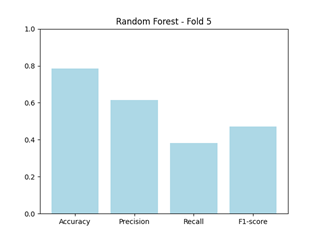
* Accuracy: 0.845
* Precision: 1.000
* Recall: 0.381
* F1-score: 0.552



Questo fold mostra una precisione perfetta, ma un recall moderato. Il modello è estremamente cauto, classificando un manga come "piace" solo quando è molto sicuro, il che può portare a perdere alcuni manga che l'utente potrebbe apprezzare.

**Fold 5:**

* Accuracy: 0.786
* Precision: 0.615
* Recall: 0.381
* F1-score: 0.471



Le prestazioni in questo fold sono coerenti con gli altri, con una buona precisione e un recall moderato, indicando un equilibrio tra evitare falsi positivi e identificare correttamente i manga apprezzati.

**Random Forest – Analisi dei Risultati**

La Random Forest ha ottenuto **ottime prestazioni complessive**, con una accuracy media dell’**82.2%**, una precisione molto alta (**0.793**) e un F1-score pari a **0.542**.  
Questi valori indicano un modello **robusto**, capace di apprendere pattern predittivi in modo efficace e di generalizzare su dati non visti, riducendo gli errori.

L’analisi dei 5 fold ha mostrato:

* Fold 1 come il più performante (F1-score = 0.722), con precisione eccellente (0.929);
* Fold 4 con precisione perfetta (1.000), ma recall contenuto (0.381);
* Fold 2 e 3 con recall più bassi, indicando una **tendenza conservativa** del modello, che classifica un manga come “piace” solo se ha alta confidenza.

Il comportamento del modello nei grafici di accuratezza conferma questa interpretazione:

* Le linee di accuracy per training e test sono **costanti, senza divergenze marcate**, suggerendo un buon equilibrio tra bias e varianza.
* Le leggere fluttuazioni tra le combinazioni iperparametriche evidenziano una certa **sensibilità ai parametri**, ma all’interno di un margine sicuro.

In sintesi, il modello Random Forest si è dimostrato:

* **Efficace, preciso e stabile**;
* **Ideale per contesti dove si desidera minimizzare i falsi positivi**;
* Utile come **modello predittivo principale** o componente centrale in un ensemble.

### **AdaBoost**

AdaBoost (Adaptive Boosting) è un algoritmo di ensemble supervisionato che combina più classificatori deboli (tipicamente alberi decisionali di profondità uno, detti "stump") in un classificatore forte. Funziona assegnando pesi maggiori agli esempi difficili da classificare correttamente e aggiornando iterativamente il modello per focalizzarsi sugli errori commessi in precedenza. Ogni classificatore debole contribuisce con un peso proporzionale alla sua accuratezza, e le predizioni finali si ottengono tramite una votazione pesata.

**Iperparametri principali**

* **n\_estimators**: numero di classificatori deboli. Un valore elevato può migliorare le performance, ma rischia l’overfitting.
* **learning\_rate**: peso attribuito a ciascun classificatore. Valori bassi rallentano l’apprendimento ma possono aumentare la generalizzazione.
* (Facoltativi, se si usa un base estimator personalizzato): **max\_depth**, **min\_samples\_split**, ecc.

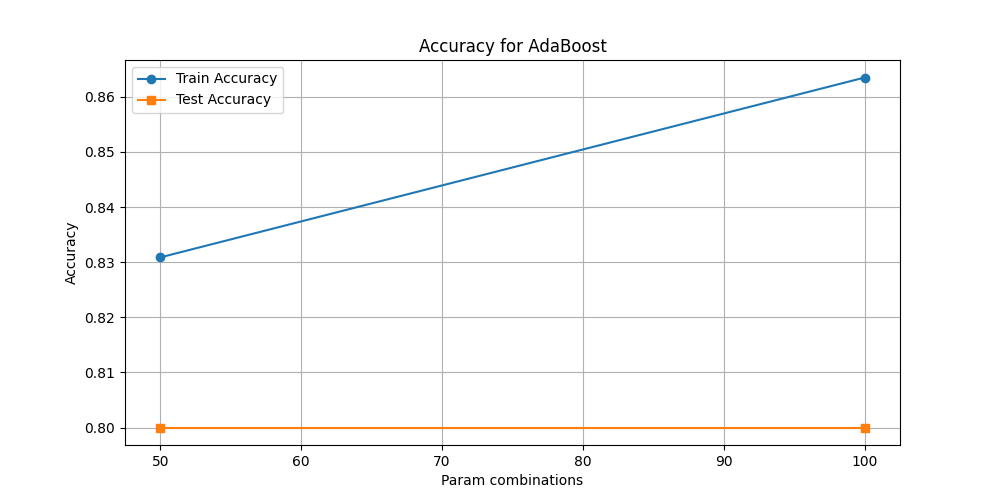
**Motivazione della scelta**

AdaBoost è stato selezionato per la sua capacità di migliorare le prestazioni di classificatori semplici e per la sua robustezza contro l’overfitting, soprattutto se ben regolarizzato. L’adattività dell’algoritmo è utile in contesti in cui i dati presentano esempi "difficili", come nella classificazione di preferenze soggettive. Inoltre, AdaBoost ha dimostrato buone prestazioni in scenari reali con dataset di dimensioni moderate.

**Grafico delle Accuratezze – Overfitting contenuto**

Il grafico mostra le accuratezze su training set (linea blu) e test set (linea arancione) al variare del numero di stimatori (n\_estimators = 50, 100).  
Osservazioni:

* L’**accuracy su training** aumenta da ~0.83 a ~0.86, come previsto, ma non raggiunge valori estremi: ciò indica una buona capacità di apprendimento senza eccessiva sovradattabilità.
* L’**accuracy su test** rimane costante a ~0.80, indicando che nonostante l’incremento della complessità, il modello non peggiora la generalizzazione.



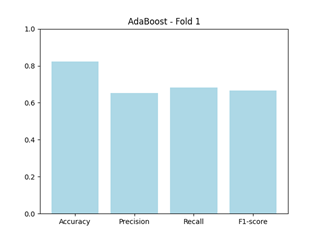
**Conclusione**: il comportamento è stabile e regolarizzato. Non si osservano segni evidenti di overfitting; l’algoritmo riesce a migliorare leggermente la performance mantenendo l’equilibrio tra precisione e recall.

**Valutazione tramite Cross Validation (5 Fold)**

È stata eseguita una validazione incrociata con 5 suddivisioni stratificate. Di seguito l’analisi per ciascun fold:

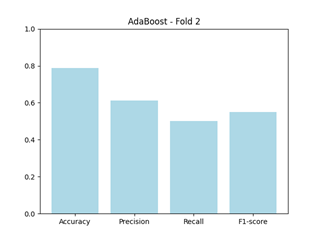
**Fold 1**

* Accuracy: 0.824
* Precision: 0.652
* Recall: 0.682
* F1-score: 0.667

  
Grafico coerente con metrica elevata su tutti i fronti. Il modello individua bene i positivi (recall alto) mantenendo buona precisione. Questo indica un equilibrio ottimale.

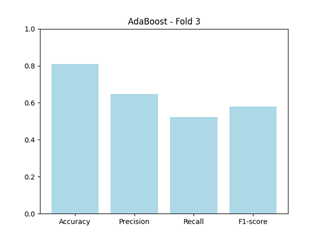
**Fold 2**

* Accuracy: 0.788
* Precision: 0.611
* Recall: 0.500
* F1-score: 0.550

  
Metrica leggermente inferiore. Il recall si riduce al 50%, suggerendo una leggera difficoltà nel riconoscere tutti i manga apprezzati, ma le performance restano robuste.

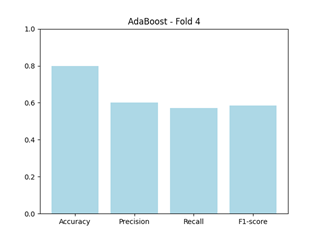
**Fold 3**

* Accuracy: 0.810
* Precision: 0.647
* Recall: 0.524
* F1-score: 0.579

  
Buona combinazione di precisione e recall. L’F1-score alto indica una corretta bilanciatura. Il grafico mostra performance solide e regolari.

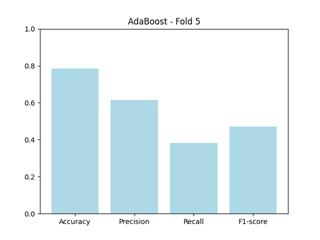
**Fold 4**

* Accuracy: 0.798
* Precision: 0.600
* Recall: 0.571
* F1-score: 0.585

  
Fold molto stabile. L’accuratezza elevata e l’F1-score oltre lo 0.58 confermano che il classificatore si comporta bene su questa partizione, mantenendo recall sopra la media.

**Fold 5**

* Accuracy: 0.786
* Precision: 0.615
* Recall: 0.381
* F1-score: 0.471

  
Fold con recall più basso (38%), ma con buona precisione. Il comportamento suggerisce maggiore cautela nelle predizioni positive.

**AdaBoost – Analisi dei Risultati**

Il modello **AdaBoost** ha mostrato performance complessive solide e coerenti, con una **accuracy media dell’80.1%**, una **precisione di 0.625**, un **recall di 0.532** e un **F1-score medio di 0.570**.

Questi risultati indicano un comportamento bilanciato tra conservatorismo e copertura, con una buona capacità di classificare correttamente sia i manga apprezzati che quelli non apprezzati, pur mantenendo un rischio contenuto di falsi positivi.

L’analisi dei 5 fold ha evidenziato:

* **Fold 1** come il migliore in termini generali, con F1-score = 0.667 e recall = 0.682, suggerendo un ottimo compromesso tra accuratezza e sensibilità;
* **Fold 3 e 4** mostrano anch’essi una buona solidità, con F1 superiori a 0.57 e metriche ben bilanciate;
* **Fold 5** ha riportato la performance più debole (F1-score = 0.471), principalmente a causa di un recall più basso (0.381), pur mantenendo una precisione accettabile;
* **Fold 2**, infine, riflette una configurazione intermedia, con recall e precision in equilibrio, ma un F1-score leggermente inferiore alla media.

Il comportamento del modello nei **grafici di accuratezza** conferma questa tendenza:

* La **Train Accuracy** cresce con l’aumentare del numero di stimatori, segnalando che il modello apprende progressivamente dai dati;
* La **Test Accuracy**, tuttavia, si mantiene stabile (~0.80), senza flessioni marcate, dimostrando che l’aggiunta di classificatori deboli non porta a overfitting;
* Il modello mostra **robustezza e resistenza alla variazione degli iperparametri**, con un margine di miglioramento ancora presente ma già buone prestazioni in partenza.

In sintesi, AdaBoost si è rivelato:

* **Affidabile e bilanciato**, capace di ridurre sia falsi positivi che falsi negativi;
* **Adatto per contesti in cui è importante preservare la copertura**, senza sacrificare troppo la precisione;
* **Un ottimo candidato per l’integrazione in un ensemble**, dove può contribuire a migliorare la sensibilità complessiva del sistema di raccomandazione.

 Target Piace ……………………………………………….………………………………………..  
 *Grafici, confusion matrix, radar plot* ……………………………………………….………..