UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

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*

AA 2024-2025

Indice

1 Introduzione

Questo elaborato presenta la progettazione e lo sviluppo di un **sistema intelligente ibrido** per l'**analisi** e la **raccomandazione di contenuti** nel dominio dei **manga**.
L'obiettivo è combinare approcci **simbolici** e **subsimbolici** dell'**intelligenza artificiale**, utilizzando **dati reali** da **API pubbliche**, tecniche di **machine learning** (**supervisionato** e **non supervisionato**), e un **motore logico** basato su **Prolog**.
L'integrazione tra **AI simbolica** (basata su **regole** e **conoscenza esplicita**, come **Prolog** e **OWL**) e **AI subsimbolica** (che apprende dai dati con **modelli statistici**) consente di creare un sistema **interpretabile**, **flessibile** e fondato su **conoscenza reale**.

1.1 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**.

Il progetto si colloca all'interno del programma formativo del corso e integra:

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

L'obiettivo didattico è dimostrare la **complementarità** tra approcci basati su **conoscenza esplicita** e **modelli appresi**, realizzando un **sistema raccomandatore ibrido**, **interpretabile** e **aggiornato**, costruito su **dati reali** ottenuti tramite **API pubbliche** (es. **MyAnimeList**).

Il progetto mira a:

- Automatizzare il recupero di **dataset personalizzati** via **OAuth2**;
- Costruire una **knowledge base Prolog** con **fatti estratti** da dati reali;
- Implementare un **motore logico di raccomandazione**;
- Eseguire **analisi supervisionate e non supervisionate** (effettuato come test) con **machine learning**;
- Integrare un'**ontologia OWL** per il **ragionamento semantico** (solo un esempio);
- Generare **output visuali** e **report esplicativi** sulle prestazioni dei modelli;
- Offrire una **valutazione critica** dei diversi approcci, con **analisi di limiti e potenzialità**.

1.2 Approccio ibrido: logico + machine learning

Il progetto si basa su un'**architettura ibrida**, che unisce due paradigmi distinti ma complementari dell'**intelligenza artificiale**:

Approccio simbolico

Questo approccio utilizza **rappresentazioni esplicite della conoscenza** tramite **simboli**, **relazioni** e **regole logiche**. Nel progetto, l'AI simbolica è realizzata attraverso:

- Una **knowledge base Prolog** con **fatti logici** (es. manga/8, lettura_utente/5);
- Un **motore logico di raccomandazione**, basato su **regole dichiarative**;
- Una **ontologia OWL minimale**, per mostrare come la **conoscenza semantica** arricchisca il **ragionamento automatico**.

Questo approccio assicura **trasparenza** e **spiegabilità**, permettendo all'utente o sviluppatore di comprendere le motivazioni delle raccomandazioni.

Approccio subsimbolico

L'**AI subsimbolica** non si basa su regole esplicite, ma apprende **pattern nei dati** tramite **algoritmi statistici**. Nel progetto si traduce in:

- Un modulo di **apprendimento supervisionato** per **classificare manga** in base alla **probabilità di gradimento**, usando algoritmi come **AdaBoost**, **Random Forest**, **KNN**;
- Un modulo di **clustering non supervisionato**, che identifica **gruppi di manga simili** con **KMeans** e **PCA**.

Questo approccio offre **adattabilità** e **potere predittivo**, migliorando con l'aumento dei **dati**.

Integrazione dei due approcci

La combinazione consente di:

- Bilanciare **precisione predittiva** e **trasparenza logica**;
- Creare un **sistema raccomandatore** più **robusto** e **versatile**;
- Valorizzare al meglio le **competenze tecniche** e i **contenuti didattici** affrontati nel corso.

2 Architettura del sistema

L'**architettura del progetto** è **modulare**, **estendibile** e composta da **componenti indipendenti**, ognuno responsabile di una fase specifica del **flusso di lavoro**. Questo approccio permette di separare chiaramente la **raccolta dati**, il **pre-processing**, il **ragionamento simbolico** e l'**analisi statistica**, favorendo sia la **chiarezza** che il **riuso del codice**.

2.1 Struttura del progetto e strumenti utilizzati

Il progetto è stato sviluppato seguendo un **approccio modulare** e suddiviso in **livelli funzionali**, per garantire **chiarezza**, **manutenibilità** e **facilità di testing**. Ogni fase del sistema è organizzata in una **cartella dedicata**, contenente **script specifici** o **file generati**.

Struttura delle directory

📁 `CLUSTERING/`

Script Apprendimento Supervisionato e Non:

- `clustering_runner.py` : clustering base con PCA (k=3)
- `kmeans_improvement.py` : ottimizzazione KMeans con silhouette/elbow
- `main.py` : esecuzione unificata delle analisi
- `model_builder.py` : factory dei modelli ML
- `param_config.py` : griglia parametri per tuning
- `plot_tools.py` : funzioni di visualizzazione dei modelli
- `preprocessing.py` : pulizia e codifica del dataset per il clustering (KMeans)
- `report_utils.py` : valutazione finale modelli (AdaBoost)
- `supervised_runner.py` : pipeline apprendimento supervisionato (classificazione)

📁 `DATASET/`

Contiene i CSV generati:

- `dataset_ml.csv` : dataset unificato con campi numerici e binarizzati
- `mangalist.csv` : lista letta dall'utente
- `top_manga.csv` : 1000 manga più popolari da MyAnimeList

📁 `DOCUMENTAZIONE/`

Contiene la documentazione nel formato docx e pdf:

- `Sistema Intelligente di Raccomandazione e Analisi del Dominio Manga.docx` : documentazione nel formato docx
- `Sistema Intelligente di Raccomandazione e Analisi del Dominio Manga.pdf` : documentazione nel formato pdf

📁 KB/

Knowledge base Prolog:

- `kb_creator.py` : genera la KB Prolog
- `knowledge_base.pl` : contiene fatti `manga/8` e `lettura_utente/5`
- `system.pl` : menu interattivo in Prolog

📁 OWL/

Esempio Ontologia OWL:

- `manga.owl` : struttura semantica con classi `Manga` , `Seinen` , `AwardWinning`
- `ontology_example.py` : script Python per ragionamento OWL (Hermit)

📁 PNG/

Grafici generati:

- Accuratezze, metriche per modello, radar chart, clustering PCA, ecc.

📁 PYTHON_DATASET/

Script estrazione dati da MyAnimeList:

- `mangalist_extended.py` : versione estesa che arricchisce il dataset con mean, rank, popolarità per ogni manga
- `top_manga.py` : scarica i manga top da MAL
- `user_manga.py` : scarica la lista dell'utente da MAL

Tool, librerie e ambienti utilizzati

Categoria	Tecnologie
Linguaggi	Python 3.10+, Prolog (SWI-Prolog), OWL/XML
Machine Learning	scikit-learn, XGBoost
Analisi dati	pandas, numpy
Visualizzazione	matplotlib, seaborn
Clustering	KMeans, PCA (da scikit-learn)
Ontologie OWL	Owlready2, HermiT
HTTP/API	requests
Ambienti IDE	Visual Studio Code, PyCharm, SWI-Prolog
Dataset	MyAnimeList API – accesso OAuth2 + raccolta dati pubblici

3 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**.

3.1 Accesso API MyAnimeList (OAuth2)

Per accedere ai **dati utente** tramite le **API di MyAnimeList (MAL)**, è necessario seguire il flusso di **autenticazione OAuth 2.0**, che consente all'utente di autorizzare un'applicazione ad accedere alle proprie **risorse** (es. lista dei manga letti) senza condividere le **credenziali**. Per utilizzare le API, è necessario richiedere un **Client ID** tramite il link ufficiale fornito da MAL.

```
# --- Credenziali dell'applicazione ---
CLIENT_ID = '823135212a297d25238a81ee65b9e53b'  # ID applicazione registrata su MAL
CLIENT_SECRET = '5ce0b51e70b4df89c3bc9d9e7102755e46cb679150fc99cb4a0a95a6dd1cdbd1'  # Chiave segreta
REDIRECT_URI = 'http://localhost:8080'  # URI di redirect registrato su MAL
```

Il processo si articola in tre **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 aperto nel browser predefinito. L'utente viene reindirizzato su **MyAnimeList** per autorizzare l'app. Dopo il consenso, viene inviato un **parametro code** al server locale;

```
# --- Genera code_verifier per PKCE ---
def generate_code_verifier(length=64):
    chars = string.ascii_letters + string.digits + "-._~"
    return ''.join(secrets.choice(chars) for _ in range(length))

# --- Passo 1: Apertura URL per autorizzazione via browser ---
def open_authorization_url(code_verifier):
    auth_url = (
        f"https://myanimelist.net/v1/oauth2/authorize?"
        f"response_type=code&client_id={CLIENT_ID}&state=1234&"
        f"redirect_uri={urllib.parse.quote(REDIRECT_URI)}&"
        f"code_challenge={code_verifier}&code_challenge_method=plain"
    )
    print("\nAprendo il browser per autorizzare...")
    webbrowser.open(auth_url)
```

2. **Cattura del codice di autorizzazione**:

   Un **server HTTP locale** (in ascolto su localhost:8080) intercetta il **codice di autorizzazione** restituito da MAL;

```
# --- Server HTTP per intercettare la redirect OAuth ---
class OAuthCallbackHandler(BaseHTTPRequestHandler):
    authorization_code = None

    def do_GET(self):
        parsed_path = urllib.parse.urlparse(self.path)
        params = urllib.parse.parse_qs(parsed_path.query)

        if 'code' in params:
            OAuthCallbackHandler.authorization_code = params['code'][0]
            self.send_response(200)
            self.end_headers()
            self.wfile.write(b"<h1>Autorizzazione completata! Puoi chiudere questa finestra.</h1>")
        else:
            self.send_response(400)
            self.end_headers()
            self.wfile.write(b"<h1>Errore: codice mancante!</h1>")
```

3. **Scambio del codice con un access token**:

   Il codice viene inviato tramite **richiesta POST** per ottenere un **access token valido**. Con questo token, è possibile accedere ai **dati utente privati** (es. lista manga, punteggi, ecc.).

```python
# --- Passo 2: Scambio del codice per ottenere l'access token ---
def get_access_token(auth_code, code_verifier):
    token_url = 'https://myanimelist.net/v1/oauth2/token'
    data = {
        'client_id': CLIENT_ID,
        'client_secret': CLIENT_SECRET,
        'grant_type': 'authorization_code',
        'code': auth_code,
        'redirect_uri': REDIRECT_URI,
        'code_verifier': code_verifier
    }
    print("Payload inviato:")
    print(data)
    headers = {
        'Content-Type': 'application/x-www-form-urlencoded',
        'User-Agent': 'Mozilla/5.0'
    }
    response = requests.post(token_url, data=data, headers=headers)

    if response.status_code == 200:
        print("Access Token ottenuto con successo!")
        return response.json()['access_token']
    else:
        print("Errore durante il recupero dell'access token:")
        print(response.status_code, response.text)
        return None
```

3.2 Dataset generati (top_manga.csv, mangalist.csv)

L'interazione con le **API di MyAnimeList** ha portato alla generazione di due **dataset principali** in formato **CSV**, archiviati nella cartella **DATASET/**. Questi file costituiscono la **base informativa** per la costruzione sia della **knowledge base simbolica** che dei **modelli di apprendimento automatico**.

top_manga.csv

Questo file è generato dallo script **top_manga.py** e contiene una selezione dei **mille manga più popolari**, ottenuti tramite la 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 mille:

	ID	Titolo	Generi	Punteggio Medio	Rank	Popolarità	Stato	Autori
2	2	Berserk	Action, Adventure, Award Winning, Drama, Fantasy, Gore, Horror, Military, Psychological, Seinen	9.47	1	1	currently_publishing	Kentarou Miura, Studio Gaga
3	1706	JoJo no Kimyou na Bouken Part 7: Steel Ball Run	Action, Adventure, Historical, Mystery, Seinen, Shounen, Supernatural	9.32	2	23	finished	Hirohiko Araki
4	656	Vagabond	Action, Adventure, Award Winning, Historical, Samurai, Seinen	9.27	3	13	on_hiatus	Takehiko Inoue, Eiji Yoshikawa
5	13	One Piece	Action, Adventure, Fantasy, Shounen	9.22	4	4	currently_publishing	Eiichiro Oda
6	1	Monster	Adult Cast, Award Winning, Drama, Mystery, Psychological, Seinen	9.16	5	28	finished	Naoki Urasawa

Nello script **top_manga.py** sono presenti le funzioni **get_top_manga**, per l'**estrazione dei dati** e **save_manga_to_csv**, per il **salvataggio 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 salvati:

- **ID** e **titolo;**
- **Generi;**
- **Stato di lettura** (es. *reading*, *completed*, *plan_to_read*);
- **Punteggio assegnato dall'utente** (**score**).

Questo file consente di **modellare le preferenze personali dell'utente**, risultando utile sia per il **ragionamento simbolico** (Prolog), sia per l'**etichettatura supervisionata** (*piace / non piace*) nei modelli di **machine learning**.

Primi dieci manga estratti dall'utente (MAL Utente):

	ID	Titolo	Generi	Punteggio	Stato
2	3	20th Century Boys	Award Winning, Drama, Historical, Mystery, Psychological, Sci-Fi, Seinen	9	completed
3	743	21st Century Boys	Award Winning, Drama, Mystery, Psychological, Sci-Fi, Seinen	0	completed
4	1224	3-gatsu no Lion	Award Winning, Childcare, Drama, Iyashikei, Seinen, Slice of Life, Strategy Game	6	on_hold
5	147337	66,666 Years: Advent of the Dark Mage	Fantasy, Reincarnation	4	dropped
6	134678	A Business Proposal	Adult Cast, Comedy, Drama, Romance, Workplace	5	completed
7	148457	A Dance of Swords in the Night	Action, Fantasy, Martial Arts	5	dropped
8	41723	A Fairytale for the Demon Lord	Action, Adventure, Fantasy, Romance	0	plan_to_read
9	158496	A Modern Man Who Got Transmigrated Into the Murim World	Action, Fantasy, Isekai, Martial Arts, Reincarnation	0	plan_to_read
10	132247	A Returner's Magic Should Be Special	Action, Fantasy, School, Time Travel	6	completed
11	50027	About Death	Psychological, Slice of Life, Supernatural	8	completed

Nello script **user_manga.py** sono presenti le funzioni **get_user_mangalist**, per l'**estrazione dei manga** dalla lista utente e **save_to_csv**, per il **salvataggio dei dati** in formato **CSV**

3.3 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 gestita dallo script **mangalist_extended.py**, che:

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

Dopo l'estrazione, i dati vengono **puliti** e **trasformati** per risultare compatibili con gli algoritmi di **apprendimento automatico**:

- **Filtraggio**: vengono considerati solo i manga valutati dall'utente (**score > 0**);
- **Tokenizzazione dei generi**: i generi sono convertiti in **liste uniformate** (es. "Action, Drama" → ["action", "drama"]);

- **Binarizzazione dei generi**: tramite **MultiLabelBinarizer**, ogni genere diventa una **colonna booleana** (1 se presente, 0 altrimenti);
- **Normalizzazione numerica**: **punteggio medio**, **rank** e **popolarità** sono trattati come **variabili numeriche continue.**

Questo avviene nel file **supervised_runner.py**, dedicato all'**apprendimento supervisionato:**

```python
# --- Pre-processing del dataset ---
df = pd.read_csv('DATASET/dataset_ml.csv')
df = df[df['Punteggio_Utente'] > 0] # Rimuove voti nulli o 0
df['Piace'] = df['Punteggio_Utente'].apply(lambda x: 1 if x >= 7 else 0)    # Target binario
df['Generi'] = df['Generi'].fillna('').apply(lambda x: [g.strip().lower().replace(' ', '_') for g in x.split(',') if g])


mlb = MultiLabelBinarizer()
generi_encoded = pd.DataFrame(mlb.fit_transform(df['Generi']), columns=mlb.classes_, index=df.index)
X = pd.concat([generi_encoded, df[['Punteggio_Medio', 'Rank', 'Popolarita']]], axis=1).fillna(0)
y = df['Piace']
```

E nel file **preprocessing.py**, dedicato alla fase di **apprendimento non supervisionato**:

```python
# --- Carica e pre-processa i dati per clustering con KMeans ---
def load_and_preprocess_kmeans(filepath='DATASET/dataset_ml.csv'):
    # Carica il dataset da CSV
    df = pd.read_csv(filepath)

    # Filtra righe con punteggio utente valido (>0 e non NaN)
    df = df[df['Punteggio_Utente'].notna() & (df['Punteggio_Utente'] > 0)]

    # Trasforma la colonna 'Generi' in liste pulite di stringhe (lowercase + underscore)
    df['Generi'] = df['Generi'].fillna('').apply(
        lambda x: [g.strip().lower().replace(' ', '_') for g in x.split(',') if g]
    )

    # Codifica multilabel in binario (una colonna per ogni genere)
    mlb = MultiLabelBinarizer()
    generi_encoded = pd.DataFrame(mlb.fit_transform(df['Generi']),
                                  columns=mlb.classes_,
                                  index=df.index)

    # Combina codifica dei generi con colonne numeriche per clustering
    X = pd.concat([generi_encoded, df[['Punteggio_Medio', 'Rank', 'Popolarita']]], axis=1).fillna(0)

    return X, df    # Ritorna i dati pronti per il clustering e il DataFrame originale
```

Il file risultante è un **dataset matriciale**, in cui le **colonne** combinano:

- **Feature binarie**, una per ogni **genere;**
- **Feature numeriche**, come **Punteggio Medio**, **Rank** e **Popolarità;**
- Un **target opzionale**, costituito dal **Punteggio Utente** e/o dalla **classe binaria** (*Piace / Non piace*) per la **classificazione.**

Primi dieci manga estratti dell'utente:

	ID	Titolo	Generi	Punteggio_Utente	Stato_Utente	Punteggio_Medio	Rank	Popolarita
2	3	20th Century Boys	Award Winning, Drama, Historical, Mystery, Psychological, Sci-Fi, Seinen	9	completed	8.94	17	24
3	743	21st Century Boys	Award Winning, Drama, Mystery, Psychological, Sci-Fi, Seinen	0	completed	8.43	200	143
4	1224	3-gatsu no Lion	Award Winning, Childcare, Drama, Iyashikei, Seinen, Slice of Life, Strategy Game	6	on_hold	8.85	28	225
5	147337	66,666 Years: Advent of the Dark Mage	Fantasy, Reincarnation	4	dropped	7.31	4781	1431
6	134678	A Business Proposal	Adult Cast, Comedy, Drama, Romance, Workplace	5	completed	7.94	892	1888
7	148457	A Dance of Swords in the Night	Action, Fantasy, Martial Arts	5	dropped	7.07	8053	4986
8	41723	A Fairytale for the Demon Lord	Action, Adventure, Fantasy, Romance	0	plan_to_read	7.09	7784	2086
9	158496	A Modern Man Who Got Transmigrated Into the Murim World	Action, Fantasy, Isekai, Martial Arts, Reincarnation	0	plan_to_read	6.96	9693	7704
10	132247	A Returner's Magic Should Be Special	Action, Fantasy, School, Time Travel	6	completed	7.59	2427	344
11	50027	About Death	Psychological, Slice of Life, Supernatural	8	completed	8.05	673	1395

Questo **dataset** viene infine utilizzato per:

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

4 Knowledge Base Prolog

La **Knowledge Base (KB)** rappresenta la componente **simbolica** del sistema, costruita con il linguaggio logico **Prolog**, noto per la sua efficacia nell'esprimere **conoscenze** e **regole dichiarative**.

Questa base raccoglie e organizza in modo **strutturato** le **informazioni estratte dai dataset**, come i **manga letti dall'utente**, i **punteggi assegnati**, i **generi associati** e altri attributi rilevanti.

L'obiettivo della KB è:

- Rappresentare i dati sotto forma di **fatti logici interrogabili;**
- Consentire al **motore inferenziale** di ragionare **simbolicamente** sulle **preferenze utente;**
- Preparare il terreno per l'attivazione di **regole di raccomandazione**, simili a quelle di un **sistema esperto.**

La KB viene generata automaticamente dallo script **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**, includendo titolo, generi, punteggio medio, popolarità e autori;
- **lettura_utente/5**: rappresenta un manga **letto dall'utente**, con **punteggio assegnato**, **stato di lettura** e **generi.**

Questa rappresentazione simbolica consente di eseguire **query flessibili e interpretabili**, come:

- "**Esistono manga premiati che l'utente non ha ancora letto?**";
- "**Quali generi compaiono più frequentemente nella sua cronologia?**".

La KB diventa così la **base logica** per il **motore di raccomandazione simbolico**, che utilizza tali informazioni per:

- Proporre **titoli affini ai gusti dell'utente**
- Esplorare **generi non ancora letti**, ampliando l'esperienza di lettura

4.1 Fatti manga/8 e lettura_utente/5

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

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 ciascun manga presente nella **classifica globale**. La struttura è la seguente:

- **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;**
- **Popolarità**: 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 **proprio stato di lettura**. 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.

4.2 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**, utilizzabili per **ragionamenti simbolici**, **query** e **raccomandazioni**.

Input

- **top_manga.csv**: contiene i manga della **top mille 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 con **safe_string()**, rimozione spazi, gestione di **valori mancanti;**
3. **Parsing dei campi multipli**: trasformazione di **generi** e **autori** in **liste;**
4. **Scrittura dei fatti Prolog** 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 la compatibilità sintattica con **Prolog**, evitando problemi dovuti a **spazi**, **caratteri speciali** o **virgolette**.

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

Struttura dei fatti

Ogni riga di **top_manga.csv** viene trasformata in un fatto **manga/8**, che contiene:

- **ID**, **titolo** (in formato compatibile con Prolog), **lista di generi;**
- **Punteggio medio**, **rank**, **indice di popolarità;**

- **Stato di pubblicazione, autori.**

Esempio:

```
manga(5394, 'kiss/hug', ['romance', '_shoujo'], 7.79, 1400, 1264, finished, ['kako_mitsuki']).
```

Ogni riga di **mangalist.csv** produce invece un fatto **lettura_utente/5**, utile per il **ragionamento sulle preferenze personali**:

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

Esempio:

```
lettura_utente(143991, 'after_god', plan_to_read, 0.0, ['action', '_fantasy']).
```

5 Motore logico simbolico

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

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

- **Identificare i generi preferiti** dall'utente;
- **Consigliare manga non letti**, ma affini ai **gusti personali;**
- **Valutare la compatibilità** tra manga e lettore, in base alla **frequenza dei generi** letti;
- E tanto altro.

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

5.1 Regole di raccomandazione

Il motore simbolico, implementato in **Prolog** nel file **system.pl**, definisce un insieme di **regole logiche** che operano sui fatti presenti nella **knowledge base** (**manga/8** e **lettura_utente/5**) per generare **raccomandazioni personalizzate** e analizzare le **abitudini dell'utente**. L'interazione avviene tramite il predicato **menu/0**, che offre un'interfaccia testuale con otto opzioni numerate, ognuna corrispondente a una specifica **funzionalità del sistema**.

Di seguito si riportano le principali regole implementate:

1 - Visualizza i generi preferiti (ordinati per frequenza)

Mostra i **generi** dei manga effettivamente letti dall'utente, **ordinandoli per frequenza decrescente**.
A tal fine, raccoglie in una lista tutti i generi associati ai manga letti, **escludendo** quelli presenti nella categoria **"plan_to_read"** mediante il predicato genere_letto(GenerePulito).
Successivamente, la lista viene **deduplicata** e **ordinata alfabeticamente**.
Per ciascun genere, il sistema **calcola il numero di occorrenze** nei manga letti, restituendo una lista di **coppie** nella forma *genere-numero*.
Infine, questa lista viene **ordinata in base alla frequenza**, in modo decrescente, così da rappresentare i generi in ordine di preferenza.

```
% Calcola la frequenza di ciascun genere
frequenza_generi(Frequenze) :-
    findall(Genere, genere_letto(Genere), ListaGeneri),
    sort(ListaGeneri, GeneriUnici),
    findall(Genere-Conta,
        (member(Genere, GeneriUnici),
         aggregate_all(count, genere_letto(Genere), Conta)),
        Frequenze).
```

```
% Estrai i generi dei manga letti (escludendo quelli ancora da leggere)
genere_letto(GenerePulito) :-
    lettura_utente(_, _, Stato, _, Generi),
    Stato \= plan_to_read,
    member(Genere, Generi),
    normalizza_genere(Genere, GenerePulito).
```

```
% Ordina i generi per frequenza (decrescente)
generi_ordinati(GeneriOrdinati) :-
    frequenza_generi(Frequenze),
    sort(2, @>=, Frequenze, GeneriOrdinati).
```

Esempio di output:

```
--- Generi preferiti (ordinati) ---
action-221
fantasy-189
drama-91
seinen-91
comedy-87
adventure-86
school-76
shounen-73
romance-70
martial_arts-46
reincarnation-43
award_winning-41
isekai-41
psychological-41
supernatural-41
slice_of_life-31
time_travel-30
horror-28
mystery-21
sci-fi-21
historical-19
adult_cast-18
gore-16
urban_fantasy-16
suspense-15
childcare-12
survival-12
iyashikei-10
super_power-10
workplace-9
gourmet-8
organized_crime-8
sports-8
love_status_quo-7
gag_humor-6
delinquents-5
military-5
shoujo-5
video_game-5
anthropomorphic-4
josei-4
mythology-4
cgdct-3
combat_sports-3
medical-3
parody-3
pets-3
space-3
team_sports-3
vampire-3
villainess-3
crossdressing-2
ecchi-2
harem-2
magical_sex_shift-2
music-2
otaku_culture-2
visual_arts-2
avant_garde-1
detective-1
girls_love-1
idols_(female)-1
kids-1
love_polygon-1
mecha-1
memoir-1
samurai-1
strategy_game-1
```

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

```
raccomanda_random(TitoloLeggibile) :-
    generi_ordinati(Generi),
    member(Genere-Count, Generi),
    Count >= 10,
    findall(ID-Titolo, (
        manga(ID, Titolo, GeneriManga, _, _, _, _, _),
        member(Genere, GeneriManga),
        \+ lettura_utente(ID, _, _, _, _)
    ), Candidati),
    list_to_set(Candidati, Unici),
    random_permutation(Unici, Mischiati),
    member(_-TitoloGrezzo, Mischiati),
    formatta_titolo(TitoloGrezzo, TitoloLeggibile).
```

Suggerisce **manga non ancora letti** appartenenti ai **generi preferiti** dell'utente, limitandosi a quelli che sono stati letti **almeno dieci volte**, selezionati in modo **casuale** tra quelli compatibili.

Il sistema identifica i **generi più frequenti** tra quelli letti, li scorre uno per uno e, per ciascuno, costruisce una lista di manga con quel genere che **non risultano ancora letti**. Di questi, memorizza **ID e titolo**, applicando una normalizzazione del titolo per migliorarne la **leggibilità** (ad esempio, rimuovendo gli **underscore**).
Infine, seleziona casualmente cinque manga dalla lista così ottenuta.

Esempi di output con la stessa KB:

```
--- Manga consigliati in base ai tuoi gusti (randomizzati) ---
iryuu: team medical dragon
last game
boku no chikyuu wo mamotte
adekan
steins;gate: eigou kaiki no pandora
```

```
--- Manga consigliati in base ai tuoi gusti (randomizzati) ---
josee to tora to sakana-tachi
sword art online: progressive
mob psycho 100
adolf ni tsugu
mairimashita! iruma-kun: if episode of mafia
```

3 - Consiglia 5 manga di qualità ma poco popolari

```
manga_qualita_nascosto(TitoloLeggibile) :-
    manga(ID, Titolo, _, Mean, _, Pop, _, _),
    number(Mean), Mean >= 8,
    number(Pop), Pop > 1500,
    \+ lettura_utente(ID, _, _, _, _),
    formatta_titolo(Titolo, TitoloLeggibile).
```

Individua **manga non ancora letti** che presentano un **punteggio medio elevato** (≥ 8) e una **popolarità limitata**, definita da un valore numerico superiore a **1500** (indicativo di opere meno conosciute).

Il sistema esamina ciascun manga, estraendone il **voto medio** e l'indice di **popolarità**, e seleziona solo quelli che soddisfano entrambi i criteri, escludendo quelli già presenti tra le **letture dell'utente**.

Esempi di output con la stessa KB:

```
--- Manga di qualitÃ  poco popolari (randomizzati) ---
zense coupling
chichi to hige-gorilla to watashi
sakamichi no apollon: bonus track
saint seiya: the lost canvas - meiou shinwa gaiden
komatta toki ni wa hoshi ni kike
```

```
--- Manga di qualitÃ  poco popolari (randomizzati) ---
ghost hunt: akumu no sumu ie
itou junji jisen kessakushuu
koi dano ai dano
josee to tora to sakana-tachi
bungou stray dogs wan!
```

4 - Consiglia 5 manga dalla tua lista "plan_to_read" con generi familiari

```
consiglia_plan_to_read(TitoloLeggibile) :-
    lettura_utente(_, Titolo, plan_to_read, _, GeneriPlan),
    findall(G,
        (lettura_utente(_, _, Stato, _, GeneriLetti),
         Stato \= plan_to_read,
         member(G, GeneriLetti)),
        ListaGeneriLetti),
    intersection(GeneriPlan, ListaGeneriLetti, Comune),
    Comune \= [],
    formatta_titolo(Titolo, TitoloLeggibile).
```

Suggerisce **manga presenti nella lista "plan_to_read"** dell'utente che condividono **almeno un genere** con quelli già letti.

Il sistema analizza tutti i titoli marcati come **"plan_to_read"**, confrontando i loro generi con quelli estratti dai **manga effettivamente letti** (escludendo, quindi, quelli pianificati). Se viene rilevata **una sovrapposizione di generi**, il manga viene considerato **compatibile** e inserito tra i suggerimenti.

Esempi di output con la stessa KB:

```
--- Consigliati tra i PLAN_TO_READ (randomizzati) ---
choujin x
chichi to ko
banana fish
mushishi
jinmen
```

```
--- Consigliati tra i PLAN_TO_READ (randomizzati) ---
umi ga hashiru end roll
katabami to ougon
akatsuki no yona
saihate no paladin
mushishi
```

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

```
manga_premiato(TitoloLeggibile) :-
    generi_ordinati(Generi),
    member(Genere-_, Generi),
    manga(ID, Titolo, GeneriManga, _, _, _, _, _),
    member(Genere, GeneriManga),
    member(award_winning, GeneriManga),
    \+ lettura_utente(ID, _, _, _, _),
    formatta_titolo(Titolo, TitoloLeggibile).
```

Raccomanda **manga non ancora letti** che abbiano ottenuto un **riconoscimento ("award_winning")** e contengano **almeno un genere** tra quelli **preferiti dall'utente**.

Il sistema esamina ciascuno dei generi preferiti, identificando i manga che presentano sia il **genere corrispondente** sia il **tag "award_winning"**, escludendo quelli già letti, e propone solo i titoli che soddisfano entrambi i criteri.

Esempi di output con la stessa KB:

```
--- Manga premiati nei tuoi generi preferiti (randomizzati) ---
maiko-san chi no makanai-san
kocchi muite! miiko
boku no hatsukoi wo kimi ni sasagu
life
kimi wa pet
```

```
--- Manga premiati nei tuoi generi preferiti (randomizzati) ---
5-toubun no hanayome
ookiku furikabutte
mystery to iu nakare
capeta
hana yori dango
```

6 - Consiglia 5 manga con almeno 2 generi completamente nuovi per te

```
manga_genere_nuovo(TitoloLeggibile) :-
    findall(Genere,
        (manga(_, _, Generi, _, _, _, _, _), member(Genere, Generi)),
        TuttiGeneri),
    sort(TuttiGeneri, GeneriTotali),

    findall(GenereLetto,
        (lettura_utente(_, _, Stato, _, GeneriLetti),
         Stato \= plan_to_read,
         member(GenereLetto, GeneriLetti)),
        GeneriLetti),
    sort(GeneriLetti, GeneriUtente),
    subtract(GeneriTotali, GeneriUtente, GeneriMaiLetti),

    manga(ID, Titolo, GeneriManga, _, _, _, _, _),
    intersection(GeneriManga, GeneriMaiLetti, Nuovi),
    intersection(GeneriManga, GeneriUtente, Noti),
    length(Nuovi, LN), LN >= 2,
    length(Noti, LO), LO =< 1,
    \+ lettura_utente(ID, _, _, _, _),
    formatta_titolo(Titolo, TitoloLeggibile).
```

Suggerisce **manga non ancora letti** che presentano **almeno due generi mai esplorati** dall'utente, con l'obiettivo di favorire la **scoperta di contenuti nuovi**.

Il sistema raccoglie l'elenco completo dei **generi presenti nel database**, elimina i duplicati e lo confronta con i **generi già letti** dall'utente. Calcola quindi la **differenza tra i due insiemi** e seleziona i manga che contengono **solo generi mai letti**, filtrando quelli che ne includono **almeno due**.

Esempi di output con la stessa KB:

```
--- Manga di un genere mai letto (randomizzati) ---
hanakoi tsurane
ashita no ousama
```

7 - Consiglia 5 manga che combinano generi noti e generi mai letti

Individua **manga non letti** che combinano **almeno un genere già esplorato** con **almeno un genere mai letto**, consentendo all'utente di **ampliare i propri gusti** senza allontanarsi completamente dalla **comfort zone**.

Rispetto alla funzionalità precedente, il filtro include solo i manga che presentano **una sovrapposizione parziale**: almeno un **genere noto** e almeno uno **nuovo** rispetto allo storico delle letture dell'utente.

```
manga_misto_generi_nuovi(TitoloLeggibile) :-
    findall(Genere,
        (manga(_, _, Generi, _, _, _, _, _), member(Genere, Generi)),
        TuttiGeneri),
    sort(TuttiGeneri, GeneriTotali),

    findall(GenereLetto,
        (lettura_utente(_, _, Stato, _, GeneriLetti),
         Stato \= plan_to_read,
         member(GenereLetto, GeneriLetti)),
        GeneriLettiRaw),
    sort(GeneriLettiRaw, GeneriUtente),

    subtract(GeneriTotali, GeneriUtente, GeneriMaiLetti),

    manga(ID, Titolo, GeneriManga, _, _, _, _, _),
    intersection(GeneriManga, GeneriUtente, ComuneLetti),
    intersection(GeneriManga, GeneriMaiLetti, ComuneNuovi),
    ComuneLetti \= [],
    ComuneNuovi \= [],
    \+ lettura_utente(ID, _, _, _, _),
    formatta_titolo(Titolo, TitoloLeggibile).
```

Esempi di output con lo stesso KB:

```
--- Manga che mischiano generi giÃ  letti e generi mai letti (randomizzati) ---
karasugaoka don't be shy!!
gakuen alice
aqua
toumei na ai no utsuwa
hydra
--- Manga che mischiano generi giÃ  letti e generi mai letti (randomizzati) ---
mahou shoujo madoka★magica
kodomo no omocha
toradora!
karasugaoka don't be shy!!
blue lock: episode nagi
```

8 - Valuta la compatibilità di una lista di generi rispetto alle tue preferenze

Dato un **elenco di generi**, il sistema ne valuta la **compatibilità** con i **gusti dell'utente**, confrontandoli con i generi letti ordinati per **frequenza**.

I generi vengono suddivisi in **tre fasce di preferenza**: alta, media e bassa, alle quali sono associati rispettivamente i punteggi **2** (molto compatibile), **1** (abbastanza compatibile) e **0** (non compatibile). Viene poi calcolata la **media dei punteggi** assegnati ai generi forniti, in base alla quale il sistema restituisce un **giudizio sintetico** sulla compatibilità complessiva.

```
valuta_compatibilita(GeneriForniti) :-
    generi_ordinati(GeneriOrdinati),  % Prende i generi ordinati per frequenza
    length(GeneriOrdinati, TotGeneri),
    Half is TotGeneri // 2,
    Quarter is TotGeneri // 4,

    findall(Punteggio,
        (member(Genere, GeneriForniti),
         nth1(Posizione, GeneriOrdinati, Genere-_),
         ( Posizione =< Half -> Punteggio = 2    % Prima metà
         ; Posizione =< Quarter * 3 -> Punteggio = 1 % Tra 25% e 50%
         ; Punteggio = 0                          % Ultimo quarto
         )
        ),
        ListaPunteggi),

    sum_list(ListaPunteggi, Somma),
    length(GeneriForniti, NGen),
    (NGen =:= 0 -> Media = 0 ; Media is Somma / NGen),

    % Media finale valutata
    ( Media >= 1.5 ->
        writeln('Questo manga è MOLTO compatibile con i tuoi gusti!')
    ; Media >= 0.75 ->
        writeln('Questo manga è ABBASTANZA compatibile con i tuoi gusti.')
    ;
        writeln('Questo manga è POCO compatibile con i tuoi gusti.')
    ).
```

Esempi di output con la stessa KB:

```
Inserisci i generi separati da virgola (es: action, fantasy, drama):
|: samurai, avant garde, vampire
Questo manga Ã¨ POCO compatibile con i tuoi gusti.
Inserisci i generi separati da virgola (es: action, fantasy, drama):
|: samurai, avant garde, seinen, drama
Questo manga Ã¨ ABBASTANZA compatibile con i tuoi gusti.
Inserisci i generi separati da virgola (es: action, fantasy, drama):
|: seinen, drama, workplace, sports
Questo manga Ã¨ MOLTO compatibile con i tuoi gusti!
```

5.2 *Menu interattivo in Prolog*

Il **sistema di raccomandazione logico-simbolico** offre un **menu testuale interattivo**, sviluppato in **Prolog** all'interno del file system.pl, che consente all'utente di **esplorare i dati** e ricevere **suggerimenti personalizzati** in base alla propria cronologia di lettura.

Funzionamento

Per avviare il menu, è necessario consultare i file knowledge_base.pl e system.pl, quindi digitare il comando:

```
1 ?- consult('knowledge_base.pl').
true.

2 ?- consult('system.pl').
true.

3 ?- menu.
```

Una volta eseguito, il sistema presenta una **lista numerata di opzioni**, ognuna delle quali attiva una specifica **regola di raccomandazione o analisi**. Selezionando un'opzione (tramite il numero corrispondente), il sistema esegue la regola associata e **mostra il risultato a video**.

```
=== SISTEMA DI RACCOMANDAZIONE MANGA ===
1. Visualizza i generi preferiti (ordinati per frequenza)
2. Consiglia 5 manga basati sui tuoi gusti piÃ¹ frequenti (random)
3. Consiglia 5 manga di qualitÃ  ma poco popolari
4. Consiglia 5 manga dalla tua lista "plan_to_read" con generi familiari
5. Consiglia 5 manga premiati compatibili con i tuoi generi preferiti
6. Consiglia 5 manga con almeno 2 generi completamente nuovi per te
7. Consiglia 5 manga che combinano generi noti e generi mai letti
8. Valuta la compatibilitÃ  di una lista di generi rispetto alle tue preferenze
9. Esci dal programma
Scelta (1-9):
```

Gestione delle scelte

Le opzioni del menu sono gestite tramite il predicato **esegui_scelta(N)**, dove N rappresenta l'opzione selezionata. Ad esempio, l'invocazione di:

```
esegui_scelta(2) :-
    writeln('--- Manga consigliati in base ai tuoi gusti (randomizzati) ---'),
    findall(Titolo, raccomanda_random(Titolo), Tutti),
    list_to_set(Tutti, Unici),
    random_permutation(Unici, Mischiati),
    primi_n(5, Mischiati, Top5),
    stampa_lista(Top5), nl,
    menu.
```

esegue il seguente flusso:

- attiva la regola raccomanda_random/1;
- seleziona **5 risultati casuali** tra quelli compatibili;
- li visualizza a schermo;
- infine, **richiama il menu principale** per consentire nuove interazioni.

6 Ontologia OWL (modulo dimostrativo)

In questa sezione viene presentato un **esempio dimostrativo** di **rappresentazione della conoscenza** mediante un'ontologia OWL, con l'obiettivo di **arricchire il progetto** attraverso un approccio **semantico-simbolico**.

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

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

Il **ragionamento ontologico** è stato realizzato tramite il modulo **Owlready2** in **Python**, con il supporto del reasoner **HermiT**, che consente di **inferire automaticamente** nuove relazioni semantiche tra i concetti definiti.

6.1 Struttura dell'ontologia manga.owl

L'ontologia **manga.owl** è un **modulo dimostrativo** progettato per rappresentare in modo **simbolico e semantico** alcune proprietà fondamentali del **dominio dei manga**. Modellata secondo i principi di **OWL (Web Ontology Language)**, costituisce una base concettuale essenziale, ideata per mostrare l'integrazione tra **conoscenza esplicita** e **ragionamento logico**.

Classi principali

- **Manga**: rappresenta la classe generica delle opere manga.
- **Seinen**: identifica un **genere narrativo** destinato a un pubblico adulto; può essere modellato come **istanza** o **sottoclasse** di un concetto più astratto.
- **AwardWinning**: concetto associato a **manga premiati**, impiegato come valore della proprietà hasAward.

Proprietà (ObjectProperty)

- **hasGenre**: collega un'istanza della classe Manga a uno o più **generi**, come ad esempio Seinen.
- **hasAward**: associa un'opera manga a un **riconoscimento**, rappresentato dal concetto AwardWinning.

Esempio nel file *manga.owl* e *del codice*:

```
=== Classi disponibili ===
manga.Manga
manga.Seinen
manga.AwardWinning
```

```
<!-- Istanza (individuo) della classe Manga -->
<owl:NamedIndividual rdf:about="#Berserk">
  <rdf:type rdf:resource="#Manga"/> <!-- Berserk è un manga -->
  <hasGenre rdf:resource="#Seinen"/> <!-- Genere: Seinen -->
  <hasAward rdf:resource="#AwardWinning"/> <!-- Ha vinto un premio -->
</owl:NamedIndividual>
```

6.2 Ragionamento con HermiT

Il **ragionamento logico** sull'ontologia è stato realizzato tramite **HermiT**, un **reasoner compatibile con OWL 2**, integrato in **Python** attraverso la libreria **Owlready2**.

L'obiettivo è fornire un esempio di **deduzione automatica** di conoscenza a partire da **relazioni dichiarative esplicite**, sfruttando le potenzialità del **Semantic Web**.

Codice per estrarre i manga premiati:

```python
# --- Stampa dei manga che hanno ricevuto un premio ---
print("\n=== Manga premiati ===")
for manga in onto.Manga.instances():    # Cicla tutte le istanze della classe Manga
    if onto.hasAward in manga.get_properties(): # Controlla se hanno la proprietà hasAward
        for prop in manga.get_properties():
            for value in prop[manga]:   # Per ogni valore della proprietà
                if "AwardWinning" in str(value):    # Se la stringa contiene AwardWinning
                    print(manga.name)   # Stampa il nome del manga premiato
```

L'utilizzo combinato di **Owlready2** e **HermiT** permette di eseguire **inferenze logiche automatiche** a partire da un'ontologia espressa in OWL, rendendo il **modello simbolico più espressivo e flessibile**.

Nel contesto del progetto, dopo aver **caricato l'ontologia manga.owl** e attivato il **reasoner**, è possibile ottenere **conoscenza implicita** tramite interrogazioni sugli **individui della classe Manga**.

Esempio output del codice:

```
=== Manga premiati ===
Berserk
```

Questo risultato è **derivato automaticamente** dal reasoner, a partire dalle proprietà dichiarate nell'ontologia, senza che tale classificazione fosse definita esplicitamente tra le istanze.

7 Machine Learning supervisionato

In questa sezione si descrive 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 numeriche e semantiche**.

L'obiettivo è costruire un **classificatore binario** capace di stimare 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** tratte dal dataset, quali **punteggio medio**, **rank** e **popolarità**.

Questa fase rappresenta la **componente subsimbolica** del sistema: l'apprendimento si basa sull'**analisi dei dati storici dell'utente** attraverso modelli statistici in grado di **generalizzare le sue preferenze**. I risultati costituiscono una **base predittiva** utile sia per la **raccomandazione diretta**, sia per l'**integrazione con il motore logico-simbolico**.

Fasi operative:

- **Costruzione del dataset supervisionato** a partire da dataset_ml.csv;
- **Binarizzazione dei generi** tramite MultiLabelBinarizer;
- **Definizione della variabile target** Piace, con valore 1 per manga valutati ≥ 7, 0 altrimenti;
- **Addestramento e confronto** di diversi modelli di classificazione;
- **Valutazione delle performance** tramite **cross-validation 5-fold**, usando le metriche.

Nel flusso supervised vengono eliminate tutte le righe con Punteggio_Utente non valido (NaN o ≤ 0) per non introdurre rumore nel target binario. I generi mancanti sono riempiti con lista vuota e, dopo One-Hot Encoding, tutti i NaN residui diventano 0, evitando errori nei modelli che non supportano valori nulli. Non è stata applicata alcuna normalizzazione sui dati di training/test perché tutti i modelli tree-based (Decision Tree, Random Forest, AdaBoost, XGBoost) e il Naive Bayes sono invarianti alla scala delle feature; inoltre, per il KNN, la scelta di *k* medio e l'omogeneità delle feature (One-Hot + punteggi) riducono l'impatto di eventuali differenze di scala.

Output dell'analisi:

L'analisi produce **grafici comparativi**, **matrici di confusione** e **radar plot**, utili per evidenziare i **punti di forza e di debolezza** di ciascun classificatore.
Questa fase consente di identificare i **modelli più adatti al profilo dell'utente**, offrendo un **supporto predittivo solido** per le successive attività di **raccomandazione** e **integrazione semantica**.

7.1 Classificazione con 6 modelli

7.1.1 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.

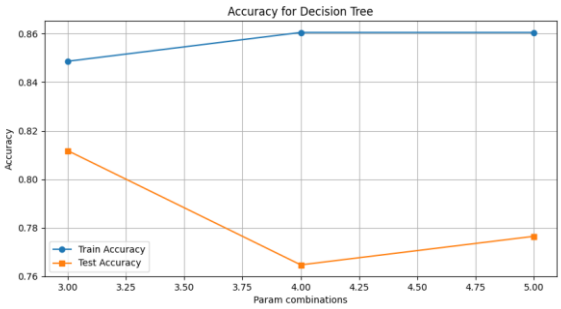
Motivazioni della scelta del modello

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.

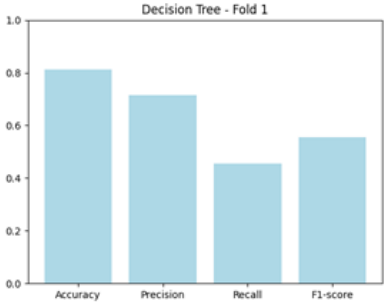
Conclusione: il modello non mostra più la tipica divergenza tra training e test dovuta all'overfitting, ma evidenzia un comportamento regolare e sotto controllo grazie alla regolarizzazione imposta.

Valutazione tramite Cross Validation (5 Fold)

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

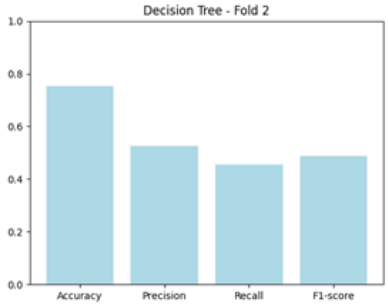
Fold 1

- 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".

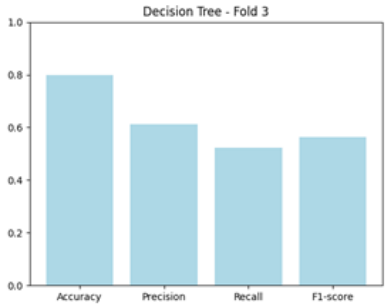
Fold 2



- 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.

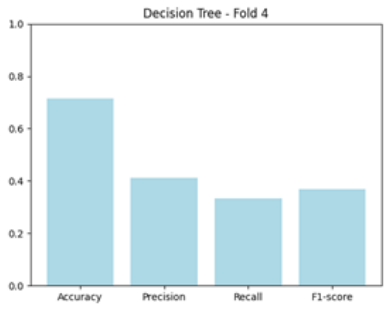
Fold 3



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

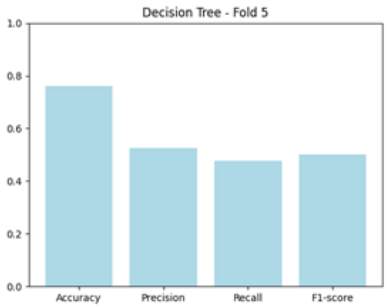
Uno dei migliori fold. Il recall supera il 50%, indicando che il modello è riuscito a **identificare correttamente una buona quantità di manga apprezzati**. Le metriche sono ben bilanciate, con una precisione stabile e un F1-score robusto.

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.

7.1.2 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

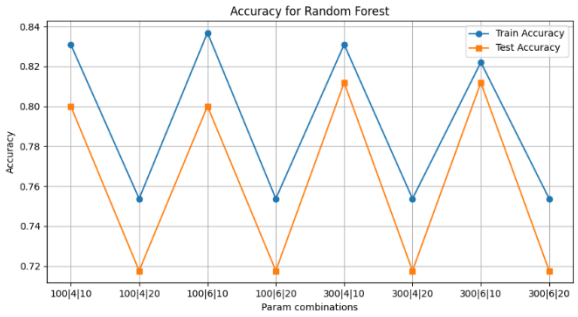
- n_estimators: numero di alberi nella foresta. Un numero maggiore migliora la stabilità, ma aumenta il tempo di calcolo.
- max_depth: profondità massima di ogni albero. Usata per contenere la complessità dei singoli alberi.
- min_samples_leaf: numero minimo di campioni richiesti in un nodo foglia.
- 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:



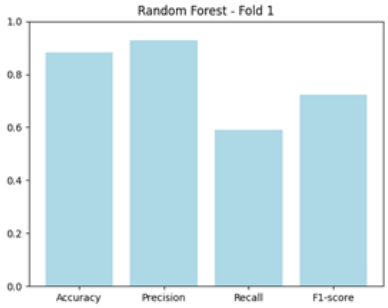
- La **train accuracy** (linea blu) si mantiene sempre alta (tra 0.75 e 0.84), con picchi attorno all'83–84% per configurazioni meno regolarizzate.
- La **test accuracy** (linea arancione) oscilla invece tra ~0.71 e ~0.81, con valori che mostrano minimi e massimi alternati, ma senza cadute drastiche.
- 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 cinque suddivisioni stratificate. Di seguito l'analisi per ciascun fold:

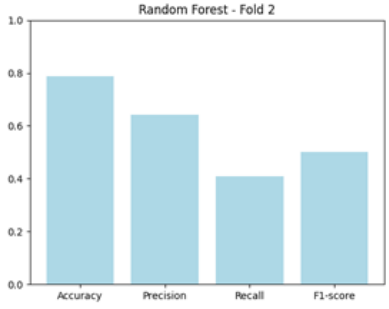
Fold 1



- 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.

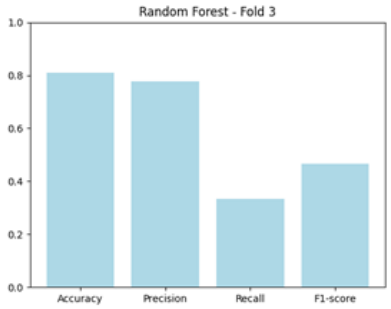
Fold 2



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

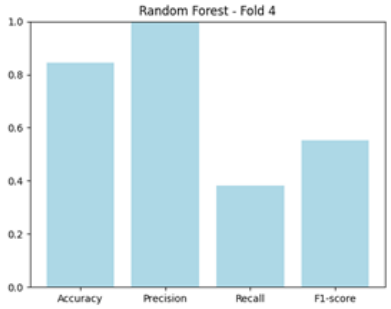
In questo fold, la precisione rimane elevata, ma il recall diminuisce, indicando che il modello è più conservativo nel classificare un manga come "piace", potenzialmente escludendo alcuni manga che l'utente potrebbe gradire.

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.

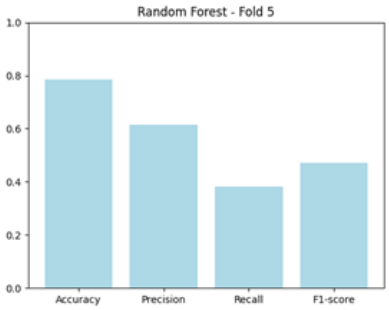


Fold 4

- 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.

7.1.3 *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

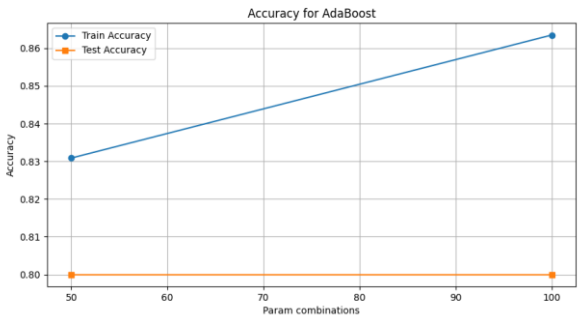
- **n_estimators**: numero di classificatori deboli. Un valore elevato può migliorare le performance, ma rischia l'overfitting.
- **learning_rate**: peso attribuito a ciascun classificatore. Valori bassi rallentano l'apprendimento ma possono aumentare la generalizzazione.
- (Facoltativi, se si usa un 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:



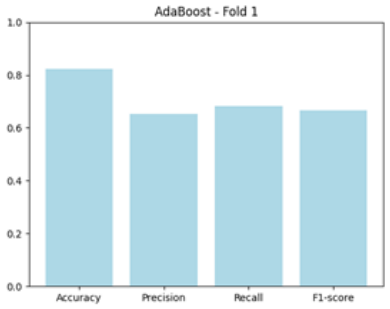
- L'**accuracy su training** aumenta da ~0.83 a ~0.86, come previsto, ma non raggiunge valori estremi: ciò indica una buona capacità di apprendimento senza eccessiva sovradattabilità.
- L'**accuracy su test** rimane costante a ~0.80, indicando che nonostante l'incremento della complessità, il modello 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 cinque suddivisioni stratificate. Di seguito l'analisi per ciascun fold:

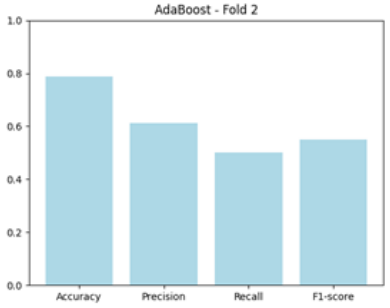
Fold 1



- 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.

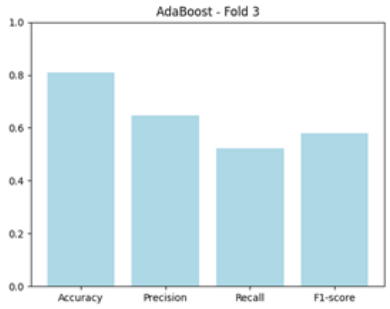
Fold 2



- 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.

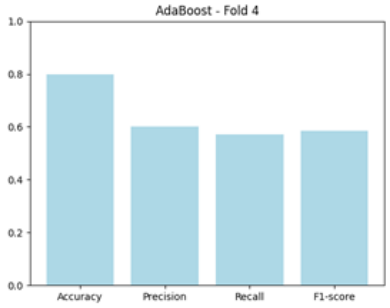
Fold 3



- 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.

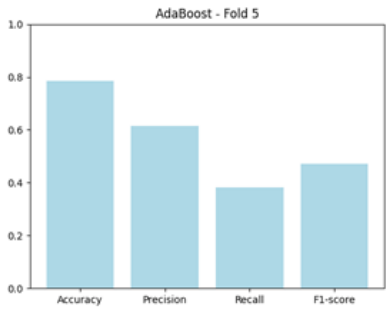
Fold 4



- 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.

7.1.4 KNN

Il K-Nearest Neighbors è un algoritmo di apprendimento supervisionato basato sulla similarità: per classificare un'istanza, il modello esamina i *k* esempi di training più vicini (secondo una distanza, di solito euclidea) e assegna la classe più frequente tra questi. Si tratta di un metodo semplice, non parametrico e istanza-based, che non costruisce un modello esplicito ma memorizza i dati e calcola in fase di predizione.

Iperparametri principali

- n_neighbors: numero di vicini da considerare. Valori più alti tendono a migliorare la generalizzazione ma possono sfocare i confini decisionali.
- weights: ponderazione dei vicini ("uniform" o "distance"), con quest'ultimo che dà maggiore peso ai vicini più prossimi.
- metric: metrica di distanza utilizzata (in questo caso: euclidea).

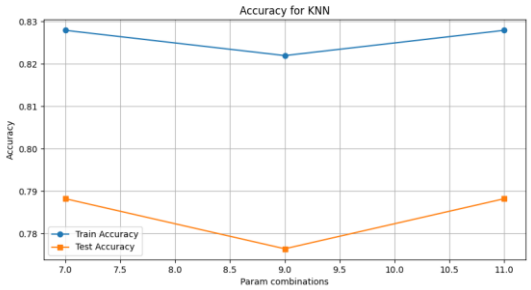
Motivazioni della scelta del modello

KNN è stato scelto come benchmark non parametrico per confrontare l'efficacia della classificazione basata su vicinanza. Il modello è utile per osservare come le caratteristiche numeriche e binarizzate (in particolare i generi) influenzano la similarità tra manga, e per testare la sensibilità del sistema al numero di vicini scelti. KNN fornisce anche un interessante punto di confronto rispetto a metodi più complessi o ensemble.

Grafico delle Accuratezze – Comportamento stabile e simmetrico

Nel grafico viene riportata l'accuracy su training set e test set per tre valori di *k* (7, 9, 11). Si osserva:

- **Stabilità**: le accuracy si mantengono costanti, con lievi oscillazioni (<1%) per entrambi i set.
- **Nessun overfitting**: le curve di train e test sono parallele e ravvicinate (~0.83 vs ~0.78), segno che il modello generalizza bene.
- **Ottima regolarizzazione intrinseca**: all'aumentare di *k*, si assiste a una leggera riduzione della varianza, senza perdita sostanziale di accuratezza.



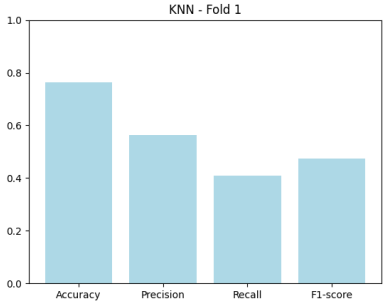
Conclusione: il comportamento simmetrico tra training e test suggerisce che KNN è un modello bilanciato, capace di evitare overfitting senza necessità di tecniche esplicite di regolarizzazione.

Valutazione tramite Cross Validation (5 Fold)

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

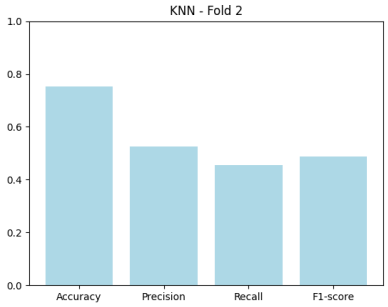
Fold 1

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



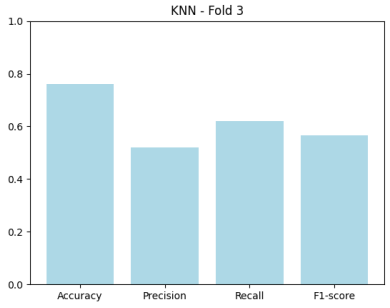
Questo fold presenta performance molto equilibrate. Il recall elevato indica che il modello è stato capace di riconoscere correttamente gran parte dei manga apprezzati. Anche precisione e F1-score sono solidi, segno di un comportamento efficace sia in copertura sia nel controllo dei falsi positivi.

Fold 2



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

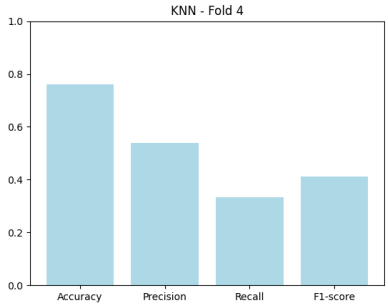
Metrica leggermente inferiore. Il recall si riduce al 50%, suggerendo una leggera difficoltà nel riconoscere tutti i manga apprezzati, ma le performance restano robuste. Il modello mostra un buon compromesso, pur tendendo a favorire la precisione.



Fold 3

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

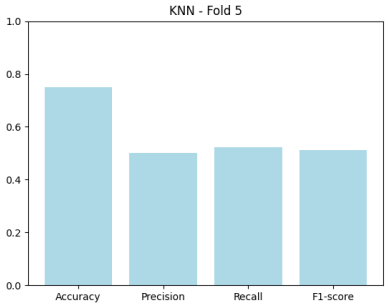
Buona combinazione di precisione e recall. L'F1-score alto indica una corretta bilanciatura. Il grafico mostra performance solide e regolari, con un comportamento affidabile nella classificazione dei manga preferiti.



Fold 4

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

Un fold stabile e bilanciato. Precisione e richiamo sono vicini, il che suggerisce che il modello riesce sia a individuare i manga apprezzati sia a mantenere sotto controllo i falsi positivi. È uno dei fold più regolari.



Fold 5

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

Il fold più conservativo: se da un lato la precisione è buona, il recall scende sotto il 40%, indicando che molti manga positivi non vengono riconosciuti. Il modello si mostra più prudente, classificando come "piace" solo in caso di alta confidenza.

KNN - Analisi dei Risultati

Il modello K-Nearest Neighbors ha prodotto risultati moderati e coerenti, con una **accuracy media del 75.8%**, una **precisione media di 0.529**, un **recall di 0.468** e un **F1-score medio di 0.490**.

Questi valori mostrano un modello semplice, che riesce a mantenere una certa affidabilità pur senza eccellere in alcuna metrica. Il comportamento è leggermente sbilanciato verso la precisione, con una copertura (recall) più contenuta, ma comunque non trascurabile. È adatto per casi in cui si preferisce un compromesso prudente tra sensibilità e conservatorismo.

L'analisi dei 5 fold ha evidenziato:

- **Fold 3** come il più performante (F1-score = 0.565), grazie a un recall relativamente alto (0.619), che segnala una buona capacità di individuare manga apprezzati;
- **Fold 1 e Fold 2** mostrano un comportamento stabile, con F1 intorno a 0.47–0.48 e recall contenuto, segno di una tendenza conservativa ma regolare;
- **Fold 5** presenta risultati intermedi, con un buon equilibrio tra precisione e recall;
- **Fold 4** è quello meno efficace (F1-score = 0.412), penalizzato da un recall basso (0.333), suggerendo una certa difficoltà nell'identificare positivamente gli esempi corretti in quella suddivisione.

Il grafico delle accuratezze rafforza queste considerazioni:

- **La Train Accuracy** si mantiene elevata e stabile (~0.82–0.83), indicando che il modello è in grado di apprendere correttamente la struttura dei dati di addestramento;
- **La Test Accuracy** si colloca attorno a ~0.78–0.79 senza flessioni drastiche, mostrando una **buona generalizzazione** e nessun sintomo evidente di overfitting;
- **La curva delle accuratezze rispetto ai valori di k** (numero di vicini) presenta un andamento regolare a U, suggerendo che esistono valori ottimali che permettono di bilanciare bias e varianza, ma senza impatti marcati sulle prestazioni complessive.

In sintesi, il KNN si è rivelato:

- Un **modello interpretabile e semplice**, adatto a situazioni in cui si richieda facilità di implementazione e controllo;
- **Bilanciato**, pur con prestazioni più modeste rispetto ad altri classificatori avanzati;
- **Utile come baseline** o come componente secondario in un ensemble, dove può contribuire con la sua semplicità a migliorare la diversità del sistema predittivo complessivo.

7.1.5 Naive Bayes

Naive Bayes è una famiglia di classificatori probabilistici basata sul Teorema di Bayes, con l'assunzione (naïve) di indipendenza tra le feature. È un modello semplice ma spesso sorprendentemente efficace, specialmente in contesti con molte feature categoriali o binarie.

Nel nostro caso, il classificatore ha operato su feature binarie (generi binarizzati) e numeriche, utilizzando il modello **GaussianNB** o **BernoulliNB**, a seconda della distribuzione delle variabili.

Iperparametri

Naive Bayes ha pochi iperparametri, il che ne facilita l'uso. Nella versione utilizzata:

- Non è stato necessario effettuare tuning avanzato.
- La semplicità computazionale ha permesso test rapidi e robusti.

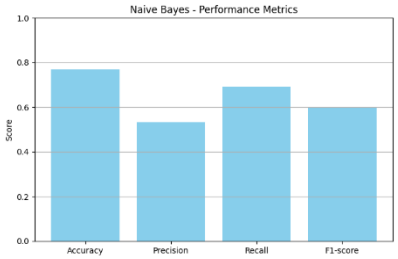
Motivazione della scelta

Naive Bayes è stato scelto per:

- La sua leggerezza computazionale.
- L'efficacia in presenza di feature indipendenti (es. generi binarizzati).
- Il buon comportamento come baseline probabilistica interpretabile.
- La tendenza a generare alti **recall**, utile per non escludere manga potenzialmente graditi.

Grafico delle Metriche – Bar Chart riassuntivo

Nel grafico sono riportate le metriche medie del modello su 5 fold:



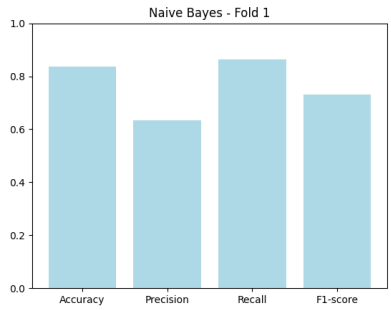
- **Accuracy:** 0.768
- **Precision:** 0.533
- **Recall:** 0.690
- **F1-score:** 0.600

Osservazioni:

- Il recall è significativamente più alto della precision, indicando che il modello tende a includere molti manga potenzialmente apprezzati, anche a costo di introdurre qualche falso positivo.
- L'accuracy è nella media rispetto agli altri classificatori.
- L'F1-score riflette un buon compromesso fra sensibilità e precisione.
- Il comportamento del modello è coerente con la sua natura "inclusiva", utile in contesti dove si desidera non perdere suggerimenti validi.

Valutazione tramite Cross Validation (5 Fold)

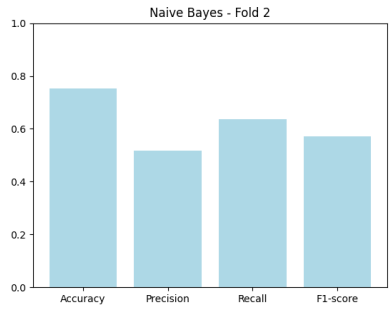
È stata eseguita una validazione incrociata con cinque suddivisioni stratificate. Di seguito l'analisi per ciascun fold:



Fold 1

- **Accuracy:** 0.835
- **Precision:** 0.633
- **Recall:** 0.864
- **F1-score:** 0.731

Ottimo fold, con il miglior recall e il F1 più alto. Il modello riesce a catturare quasi tutti i manga graditi, mantenendo anche una precisione accettabile.



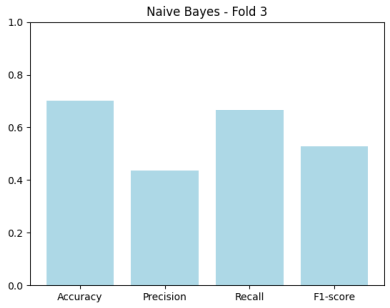
Fold 2

- **Accuracy:** 0.753
- **Precision:** 0.519
- **Recall:** 0.636

- **F1-score:** 0.571

Buona copertura dei positivi, anche se a scapito della precisione. Performance comunque robuste.

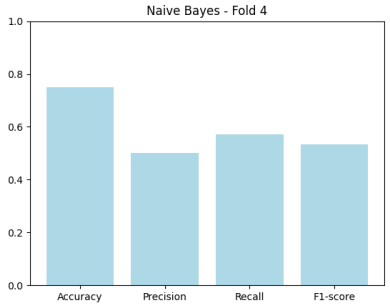
Fold 3



- **Accuracy:** 0.702
- **Precision:** 0.438
- **Recall:** 0.667
- **F1-score:** 0.528

Fold più debole. Il recall resta alto, ma la precisione scende notevolmente, indicando un aumento dei falsi positivi.

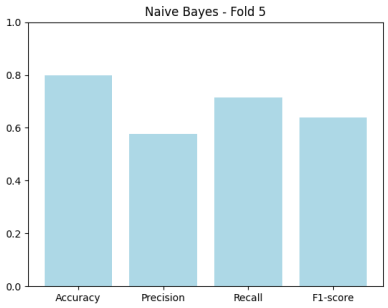
Fold 4



- **Accuracy:** 0.750
- **Precision:** 0.500
- **Recall:** 0.571
- **F1-score:** 0.533

Prestazioni intermedie e regolari. Metriche bilanciate ma non elevate, buona stabilità complessiva.

Fold 5



- **Accuracy:** 0.798
- **Precision:** 0.577
- **Recall:** 0.714
- **F1-score:** 0.638

Ottimo recall e buon equilibrio generale. Performance solide e consistenti.

Naive Bayes – Analisi dei Risultati

Il modello **Naive Bayes** ha evidenziato una spiccata capacità di **copertura**, con un recall medio del **69%**, superiore a quello degli altri classificatori finora analizzati.
L'accuracy media è **76.8%**, mentre la precisione, più bassa (**0.533**), conferma una certa tendenza del modello a classificare positivamente anche casi dubbi.

Analizzando i 5 fold:

- **Fold 1** è il migliore (F1=0.731), con recall 86.4%, segno di ottima sensibilità;
- **Fold 5** mostra un buon bilanciamento, con F1=0.638 e recall oltre il 70%;
- **Fold 3** è il più debole, con precision sotto 0.45, ma mantiene comunque recall > 66%;
- Gli altri fold mostrano metriche stabili, senza grandi oscillazioni.

In sintesi, **Naive Bayes** si dimostra:

- Un modello **inclusivo e sensibile**, utile quando è fondamentale non perdere elementi potenzialmente apprezzati dall'utente;

- Adatto in **sistemi di raccomandazione orientati alla copertura**, soprattutto se integrato in un **ensemble** con modelli più conservativi (es. Decision Tree);
- Veloce, interpretabile, robusto, e quindi efficace anche in contesti a risorse limitate.

7.1.6 XGBoost

XGBoost (Extreme Gradient Boosting) è un algoritmo di boosting basato su alberi decisionali, che costruisce i modelli in modo sequenziale, correggendo iterativamente gli errori commessi dai classificatori precedenti. Si distingue per l'elevata efficienza computazionale, la regolarizzazione integrata e la capacità di gestire overfitting e dati rumorosi. È particolarmente adatto per problemi complessi in cui si vogliono ottenere modelli robusti e generalizzabili.

Iperparametri principali:

- n_estimators: numero di alberi nel modello;
- learning_rate: tasso di apprendimento che controlla il contributo di ciascun albero;
- max_depth: profondità massima degli alberi;
- subsample: frazione dei dati da usare per ogni albero (per migliorare la generalizzazione).

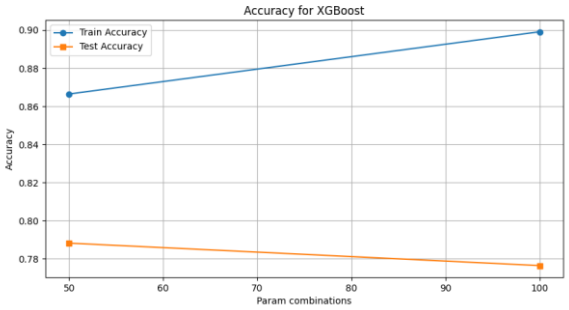
Motivazione della scelta

XGBoost è stato scelto per valutare un approccio di boosting avanzato, che potesse offrire un miglior compromesso tra bias e varianza rispetto ai modelli di base (es. Decision Tree). Grazie alle tecniche di regolarizzazione e al meccanismo di aggiornamento graduale, è spesso in grado di ottenere performance superiori su dati rumorosi o sbilanciati, come nel caso delle preferenze utente.

Grafico delle Accuratezze – Stabilità e Overfitting controllato

Nel grafico sottostante si osserva il comportamento del modello al variare del numero di stimatori. La Train Accuracy (linea blu) aumenta visibilmente (dal 86.7% al 89.9%), indicando che il modello apprende in modo crescente dai dati. Tuttavia, la Test Accuracy (linea arancione) rimane pressoché stabile (tra 0.779 e 0.788), suggerendo un leggero rischio di overfitting. La separazione tra le due curve cresce, ma resta entro margini accettabili.

Osservazioni:

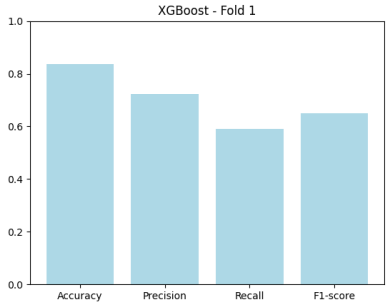


- Il modello è molto preciso sull'addestramento, ma mantiene la Test Accuracy stabile attorno all'80%, senza crolli evidenti.
- Questo comportamento suggerisce una buona capacità di generalizzazione, pur lasciando spazio a un potenziale tuning ulteriore.
- La stabilità della curva arancione è indice di robustezza.

Conclusione: XGBoost è in grado di apprendere con efficienza e mantenere una buona coerenza predittiva anche con l'aumento della complessità.

Valutazione tramite Cross Validation (5 Fold)

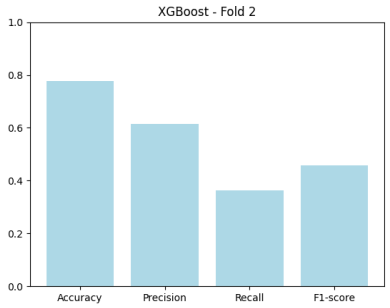
È stata eseguita una validazione incrociata con cinque suddivisioni stratificate. Di seguito l'analisi per ciascun fold:



Fold 1

- **Accuracy**: 0.835
- **Precision**: 0.722
- **Recall**: 0.591
- **F1-score**: 0.650

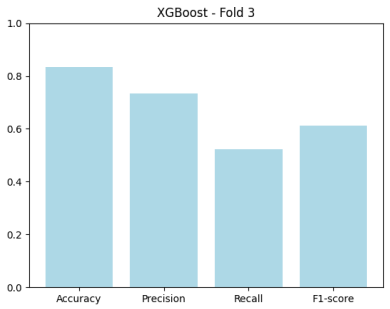
Fold molto solido, con buon equilibrio tra precisione e recall. L'F1-score alto riflette una classificazione efficace sia dei casi positivi che negativi.



Fold 2

- **Accuracy**: 0.776
- **Precision**: 0.615
- **Recall**: 0.364
- **F1-score**: 0.457

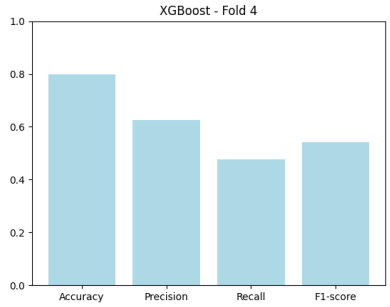
Fold meno performante: il recall è basso (36.4%), segnale di una difficoltà nel riconoscere tutti i manga graditi. La precisione resta però soddisfacente.



Fold 3

- **Accuracy**: 0.833
- **Precision**: 0.733
- **Recall**: 0.524
- **F1-score**: 0.611

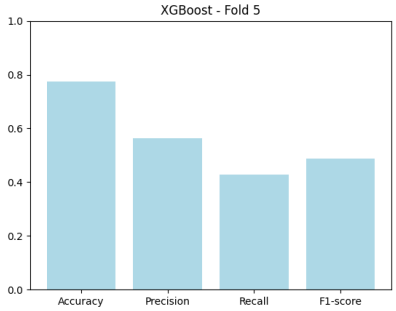
Ottimo equilibrio. Recall oltre il 50% e precisione elevata, con un F1-score robusto che conferma la solidità del modello.



Fold 4

- **Accuracy**: 0.798
- **Precision**: 0.625
- **Recall**: 0.476
- **F1-score**: 0.541

Fold coerente, con metriche tutte al di sopra del 47%. Buona affidabilità predittiva.



Fold 5

- **Accuracy**: 0.774
- **Precision**: 0.562
- **Recall**: 0.429
- **F1-score**: 0.486

Fold più debole: il recall è contenuto, ma precisione e accuratezza rimangono su valori accettabili.

XGBoost – Analisi dei Risultati

Il modello XGBoost ha evidenziato performance globalmente positive, con:

- **Accuracy media**: 80.3%
- **Precisione media**: 0.652
- **Recall medio**: 0.477
- **F1-score medio**: 0.549

Questi risultati indicano una chiara inclinazione verso la precisione, a discapito della copertura. Il modello è conservativo: preferisce segnalare un manga come “non piace” piuttosto che rischiare un falso positivo. L’analisi dei singoli fold mostra:

- **Fold 1 e 3** con le migliori performance generali (F1 ≥ 0.61), a conferma della stabilità del modello;
- **Fold 2 e 5** leggermente più deboli in recall, ma comunque consistenti;
- **Fold 4** intermedio e bilanciato.

Il grafico delle accuratezze mostra un incremento della capacità predittiva sul training set, ma una leggera discesa su quello di test, da monitorare per evitare overfitting eccessivo.

In sintesi, XGBoost è:

- Un modello solido e preciso, con prestazioni elevate sui dati visti;
- Particolarmente adatto a scenari in cui si vuole minimizzare il rischio di falsi positivi;
- Un ottimo candidato da combinare in ensemble, in grado di rafforzare modelli meno stabili.

7.1.7 Analisi Parametri Esplorati e Complessità

Analisi Parametri Esplorati

Per ogni modello si è scelto di limitare la complessità strutturale e di bilanciare la sensibilità alle classi per mitigare overfitting e sbilanciamenti.

Modello	Iper-parametri esplorati	Motivazione dal libro
Decision Tree	max_depth ∈ {3,4,5} min_samples_leaf ∈ {10,20} class_weight="balanced"	Limita profondità e dimensione foglie per ridurre overfitting (§7.4-7.5). Bilanciamento classi (§7.7).
Random Forest	n_estimators ∈ {100,300} max_depth ∈ {4,6} min_samples_leaf ∈ {10,20} max_features ∈ {"sqrt","log2"} class_weight="balanced"	Ensemble stabile con controllo varianza (§8.2-8.5).
AdaBoost	n_estimators ∈ {50,100} learning_rate ∈ {0.05,0.1,0.5}	LR basso per evitare overfitting in boosting (§9.3).
K-Nearest Neighbors	n_neighbors ∈ {7,9,11}	k medio → compromesso rumore/generalizzazione (§10.2).
Naive Bayes	–	Modello a bassa varianza, nessun tuning (§11.1).
XGBoost	n_estimators ∈ {50,100} max_depth ∈ {3,4} learning_rate ∈ {0.05,0.1} subsample=0.7 colsample_bytree=0.7 reg_alpha=1 reg_lambda=1 scale_pos_weight=1.5	Regolazione di profondità e regolarizzazione per evitare overfitting (§12.4-12.7).

Decision Tree: il controllo della profondità (max_depth={3,4,5}) e della dimensione minima delle foglie (min_samples_leaf={10,20}) riduce la varianza del modello, mentre class_weight="balanced" compensa le classi meno rappresentate.

Random Forest: aumentando il numero di alberi (n_estimators={100,300}) miglioriamo la stabilità, ma limitiamo la profondità (max_depth={4,6}) e il numero di feature per split (max_features={"sqrt","log2"}) per non far crescere troppo la complessità computazionale e la varianza residua; anche qui class_weight="balanced" aiuta il recall della classe più rara.

AdaBoost: poiché il boosting adattivo tende a sovra-adattarsi con troppi stimatori o tassi di apprendimento alti, abbiamo scelto pochi alberi deboli (n_estimators={50,100}) e tassi di apprendimento contenuti (learning_rate={0.05,0.1,0.5}) per trovare un compromesso fra accuratezza e generalizzazione.

KNN: valori di k medi (n_neighbors={7,9,11}) attenuano l'effetto del rumore mantenendo un buon potere discriminante, come suggerito dal capitolo sui vicini più prossimi.

Naive Bayes: essendo un modello a bassa varianza, non richiede iper-parametri, il che ne garantisce efficienza e semplicità.

XGBoost: per questo potente ensemble abbiamo imposto profondità contenute (max_depth={3,4}) e un learning rate piccolo (learning_rate={0.05,0.1}), insieme a subsample e colsample_bytree a 0.7 per regolarizzare campioni e feature; infine, reg_alpha e reg_lambda (=1) aggiungono penalità L1/L2, mentre scale_pos_weight=1.5 rafforza la classe minoritaria.

In tutti i casi, la ricerca a griglia combinatoria, unita a 5-fold stratificato e ripetuto con seed fissati, garantisce risultati ripetibili e confrontabili; i range selezionati si basano sui consigli teorici di Poole & Mackworth e mirano a esplorare un compromesso tra bias e varianza senza andare in overfitting su dataset di dimensione limitata.

Analisi Complessità

Per ogni modello si è scelto di limitare la complessità strutturale e di bilanciare la sensibilità alle classi per mitigare overfitting e sbilanciamenti.

Modello	**Tempo di training**	**Memoria**
Decision Tree	$O(n \cdot d \cdot \log n)$	$O(n)$
Random Forest	$O(t \cdot n \cdot d \cdot \log n)$ ($t = n_{estimators}$)	$O(t \cdot n)$
AdaBoost	$O(t \cdot T_{base})$	$O(t \cdot n)$
KNN	O(1) training; $O(n \cdot d)$ inferenza	$O(n \cdot d)$
Naive Bayes	$O(n \cdot d)$	$O(d \cdot k)$ (k = # classi)
XGBoost	$O(t \cdot n \cdot \log n)$	$O(t \cdot n)$

n = numero esempi, *d* = numero feature, *t* = numero stimatori.

Decision Tree presenta un costo di training di $O(n \cdot d \cdot \log n)$, dove n è il numero di esempi e d il numero di feature, e richiede memoria $O(n)$ è quindi piuttosto scalabile su dataset di medie dimensioni, ma cresce più velocemente se aumentano gli attributi o i campioni.

Random Forest, con t stimatori, ha tempo di training $O(t \cdot n \cdot d \cdot \log n)$ e memoria $O(t \cdot n)$; l'aumento degli alberi migliora la stabilità, ma allunga proporzionalmente i tempi e il footprint in memoria.

AdaBoost richiede $O(t \cdot T_{base})$ tempo, dove T_{base} è il costo di training del classificatore debole (tipicamente un albero poco profondo), e $O(t \cdot n)$ spazio per mantenere pesi e predizioni intermedie. È più leggero di Random Forest finché gli stadi t restano contenuti.

K-Nearest Neighbors non ha fase di training O(1), ma l'inferenza costa $O(n \cdot d)$ per ogni previsione e richiede memoria $O(n \cdot d)$ per conservare l'intero dataset, il che può diventare oneroso in fase di deployment su grandi insiemi di dati.

Naive Bayes scorre linearmente sui dati con $O(n \cdot d)$ sia in tempo che in spazio $O(d \cdot k)$ (con k numero di classi): ciò lo rende estremamente efficiente e adatto a scenari real-time o vincolati in memoria.

XGBoost fonde boosting e struttura ad albero, con tempo $O(t \cdot n \cdot \log n)$ e memoria $O(t \cdot n)$; pur generando modelli molto performanti, conviene mantenere basso t e profondità per contenere l'impatto computazionale.

Le notazioni Big-O riportate per ciascun algoritmo derivano dall'analisi asintotica delle operazioni fondamentali descritte nel libro (Poole & Mackworth, capitoli dedicati agli algoritmi corrispondenti ed internet). In sostanza, le formule asintotiche standard dai testi di riferimento sono state adattate ai parametri (*n*, *d*, *t*), per dare un'indicazione chiara dell'impatto computazionale di ciascuna scelta.

7.2 Target Piace

Per trasformare il problema del gradimento in una classificazione binaria supervisionata, è stata definita una variabile target denominata **Piace**, che assume valore:

- 1 se il manga ha ricevuto un **punteggio medio (score) maggiore o uguale a 7** da parte dell'utente;
- 0 in tutti gli altri casi.

Questa soglia è stata scelta sulla base di considerazioni sia **empiriche** che **semantiche**:

- Nei principali portali di valutazione di anime e manga (es. **MyAnimeList**, **AniList**), il punteggio 7 rappresenta una soglia comunemente interpretata come **"buono"** o **"gradito"**, a differenza di voti inferiori che indicano disinteresse o insoddisfazione.
- L'analisi esplorativa del dataset ha mostrato che la **distribuzione dei punteggi** presenta una **curva centrata attorno a 6.5–7**, rendendo la soglia 7 un punto discriminante utile per separare classi bilanciate con sufficiente densità.
- Soglie più alte (es. ≥8) avrebbero ristretto eccessivamente la classe positiva, causando **sbilanciamento** e perdita di informazioni. Soglie più basse (es. ≥6) avrebbero incluso molti titoli neutri o mediocri, riducendo la qualità predittiva.

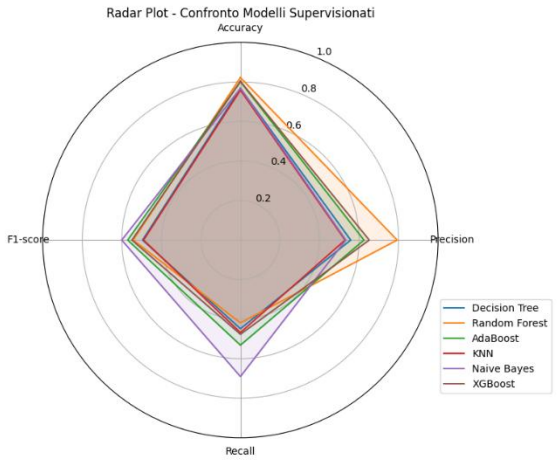
La scelta della soglia 7 consente di:

- **Semplificare il problema** in un task di **classificazione binaria**, interpretabile e gestibile da tutti i principali modelli supervisionati;
- **Mantenere un buon equilibrio** tra le due classi, evitando squilibri che renderebbero l'apprendimento difficile o ingannevole;
- **Valutare le performance in modo coerente**, attraverso metriche classiche (accuracy, precision, recall, F1-score), mantenendo un significato pratico nelle predizioni (ovvero: "questo manga *piacerà* con buona probabilità").

7.3 Confusion Matrix e Radar Plot

I grafici riepilogativi permettono un confronto visivo ed efficace tra i modelli supervisionati analizzati. Il *radar plot* mostra le performance medie dei sei classificatori (Decision Tree, Random Forest, AdaBoost, KNN, Naive Bayes, XGBoost) sulle quattro metriche fondamentali: **Accuracy**, **Precision**, **Recall** e **F1-score**. Si evidenziano alcune considerazioni chiave:

- **Random Forest** eccelle in *precision*, confermando la sua tendenza conservativa (alta confidenza per i positivi);
- **Naive Bayes** mostra il *recall* più elevato, indicando una forte propensione alla copertura, anche a costo di qualche falso positivo;
- **AdaBoost** e **XGBoost** si mantengono equilibrati, con performance solide e stabili su tutte le metriche;
- **KNN** e **Decision Tree** riportano metriche più contenute, ma con profili regolari e coerenti.

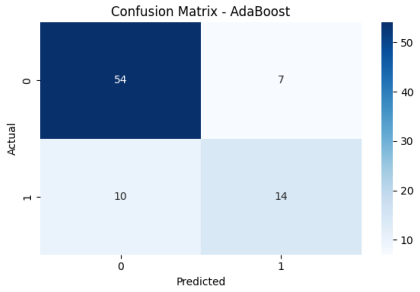


Questa visualizzazione sintetica consente di valutare rapidamente i punti di forza di ciascun modello e di orientare la scelta verso quello più adatto agli obiettivi (alta precisione, elevata sensibilità o bilanciamento tra le due).

Accanto al radar plot, la *confusion matrix* di **AdaBoost** fornisce un'ulteriore chiave di lettura:

- Dei 64 manga effettivamente non apprezzati (classe 0), **54** sono stati correttamente classificati (*True Negative*), mentre **7** sono stati erroneamente indicati come "piace" (*False Positive*);
- Dei 24 manga apprezzati (classe 1), **14** sono stati riconosciuti correttamente (*True Positive*), mentre **10** sono stati ignorati dal modello (*False Negative*).

Questo evidenzia un buon compromesso tra conservatorismo e sensibilità. Il modello mantiene il rischio di falsi positivi relativamente basso, ma riesce comunque a recuperare una parte significativa dei manga positivi, risultando efficace in uno scenario realistico di raccomandazione.



8 Clustering KMeans (TEST)

Il **clustering** è una tecnica di apprendimento non supervisionato che ha l'obiettivo di raggruppare istanze simili in insiemi distinti, detti *cluster*, sulla base di caratteristiche condivise. A differenza dei metodi supervisionati, non richiede etichette di classe predefinite: il modello apprende direttamente dalla struttura intrinseca dei dati.

Nel presente progetto è stato adottato l'algoritmo **KMeans**, uno dei metodi di clustering più diffusi e semplici. L'idea alla base di KMeans è iterativamente assegnare ogni punto al cluster il cui centroide (media geometrica dei punti) è più vicino, e poi aggiornare tali centroidi fino a convergenza. Il numero di cluster k deve essere definito a priori: per questo motivo si sono adottate strategie sia manuali che automatiche per la sua determinazione.

L'analisi è stata condotta su una rappresentazione vettoriale dei manga basata su:

- **generi** (codificati in forma binaria);
- **valori numerici** come Punteggio_Medio, Rank e Popolarità.

Per migliorare la visualizzazione dei risultati e semplificare la struttura dei dati, si è impiegata la **Principal Component Analysis (PCA)** per ridurre le dimensioni del dataset a due componenti principali. Ciò ha consentito di rappresentare visivamente i cluster in un piano bidimensionale e di interpretare più agevolmente la segmentazione effettuata dal modello.

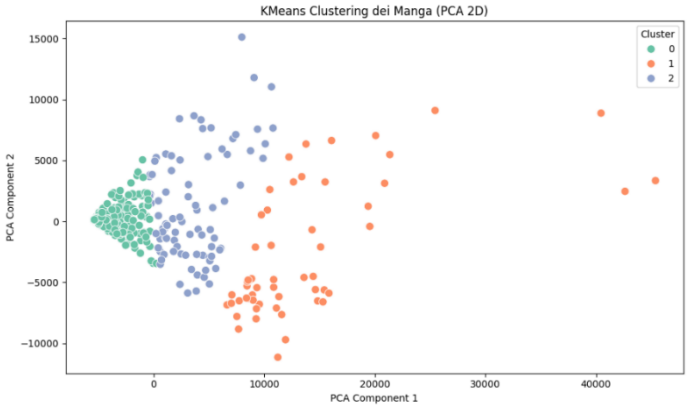
8.1 Clustering base e ottimizzato (PCA, silhouette)

L'analisi KMeans è stata implementata in due versioni: una **versione base** con un valore fisso di cluster, e una **versione ottimizzata** in cui il numero ottimale di cluster è determinato automaticamente. Entrambe si basano su una riduzione dimensionale tramite **Principal Component Analysis (PCA)** per consentire una visualizzazione bidimensionale efficace dei risultati.

Clustering base (k=3)

Nel file clustering_runner.py è stato eseguito il clustering con k=3 fissato manualmente. I passaggi includono:

- Pre-elaborazione dei dati (inclusa codifica binaria dei generi);
- Applicazione della PCA per ridurre a due le dimensioni;
- Esecuzione del clustering con KMeans(n_clusters=3);
- Visualizzazione 2D dei cluster ottenuti.



In questo grafico, i manga sono distribuiti su due assi (PCA1 e PCA2), colorati secondo il cluster assegnato. Sebbene siano visibili alcune tendenze di separazione, i gruppi non risultano ben definiti: i punti mostrano **sovrapposizione significativa** e **bassa coesione intra-cluster**, sintomo di una scelta di k non ottimale.

Distribuzione multipla dei cluster: nei primi 10 risultati si osserva una **distribuzione dei manga su più cluster** (es. 0 e 2), coerente con l'assegnazione casuale a tre gruppi. Tuttavia, i gruppi sembrano **squilibrati**, con la maggior parte dei titoli assegnata al cluster 0.

```
=== ANALISI NON SUPERVISIONATA (KMeans) ===

Esempio cluster assegnati:
                                  Titolo  Cluster
0                        20th Century Boys        0
2                          3-gatsu no Lion        0
3     66,666 Years: Advent of the Dark Mage        0
4                      A Business Proposal        0
5            A Dance of Swords in the Night        2
8     A Returner's Magic Should Be Special        0
9                               About Death        0
10                    Absolute Regression        2
11                    Absolute Sword Sense        0
13  After School Lessons for Unripe Apples        0
```

Questo comportamento riflette **l'instabilità** tipica del KMeans con k arbitrario, specialmente se i dati non presentano confini netti.

Clustering ottimizzato (scelta automatica di k)

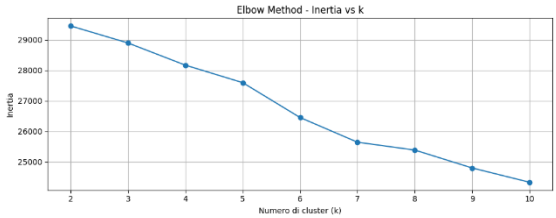
Il file kmeans_improvement.py introduce un processo per determinare il numero ideale di cluster utilizzando due metriche:

- **Elbow Method**, che misura l'**inerzia** (compattezza);
- **Silhouette Score**, che valuta la **coerenza interna dei cluster**.

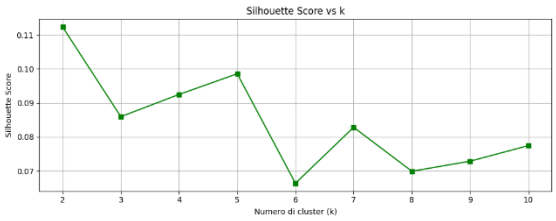
Nella pipeline di clustering migliorato, prima di applicare K-Means, i dati vengono scalati su media zero e varianza unitaria tramite StandardScaler(). Questo passaggio garantisce che tutte le dimensioni contribuiscano in modo equilibrato al calcolo delle distanze, migliorando la qualità e la stabilità dei cluster. I missing value residui, se presenti, sono comunque riempiti con 0 per coerenza con il preprocessing supervised.

Il range esplorato va da 2 a 10 cluster. I valori delle due metriche sono rappresentati in forma grafica.

La curva dell'inerzia mostra un declino costante con l'aumentare di k, ma non presenta un gomito netto. Un possibile punto di flesso si intravede attorno a k=5, dopo il quale il guadagno in compattezza diventa meno marcato. Tuttavia, l'analisi da sola non è conclusiva.



Il silhouette score, che misura la separazione tra i cluster, **raggiunge il massimo per k=2**, suggerendo che in base alla distribuzione interna dei dati, due cluster risultano meglio definiti rispetto ad altri valori. Tuttavia, il valore assoluto (~0.11) è piuttosto basso, indicando che la struttura di clustering è **debole e sovrapposta**.

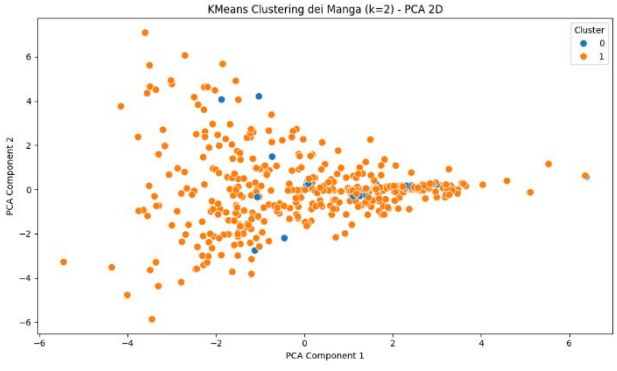


Dopo la valutazione incrociata delle due metriche, viene comunque scelto il valore di k che **ottimizza il silhouette score**.

Visualizzazione finale del clustering ottimizzato

Una volta identificato il k ottimale, il clustering viene rieseguito e proiettato nel piano PCA.

A differenza del clustering base, il risultato mostra gruppi **maggiormente separati e compatti**, suggerendo che la determinazione automatica di k ha migliorato la qualità della segmentazione. Le componenti principali evidenziano la presenza di due aree ben definite, associate a caratteristiche comuni tra i manga raggruppati.



Assegnazione univoca a un solo cluster: tutti i manga mostrati nell'output di esempio appartengono al **cluster 1**, indicando una **forte sbilanciatura** nei gruppi.

Questa polarizzazione è coerente con il **silhouette score basso** osservato nel grafico precedente, che suggerisce **debole struttura di clustering**. In pratica, **i dati non si segmentano bene** in cluster distinti, e KMeans finisce per concentrare quasi tutto in un unico gruppo.

```
--- KMeans Clustering con miglioramento ---
Numero di cluster ottimale (k): 2

Esempio di manga e cluster assegnato:
                                  Titolo  Cluster
0                          20th Century Boys        1
2                            3-gatsu no Lion        1
3    66,666 Years: Advent of the Dark Mage        1
4                        A Business Proposal        1
5           A Dance of Swords in the Night        1
8     A Returner's Magic Should Be Special        1
9                                About Death        1
10                     Absolute Regression        1
11                     Absolute Sword Sense        1
13  After School Lessons for Unripe Apples        1
```

Configurazione

Per gestire l'elevata dimensionalità delle feature originali ho applicato una PCA che riduce da d dimensioni alle prime 10 componenti, mantenendo circa il 70 % della varianza: questo accelera sia il calcolo delle distanze in K-Means sia la convergenza, oltre ad attenuare il rumore (§17.3).

Nella fase di clustering, abbiamo esplorato valori di k in {2, 3, 4, 5} utilizzando sia il "punto di gomito" sull'inerzia intra-cluster sia il coefficiente di silhouette (valori target > 0.5) per scegliere il numero di gruppi più interpretabile e ben separato (§17.5).

Considerazioni conclusive

I risultati ottenuti dall'algoritmo KMeans hanno evidenziato alcune **criticità strutturali** nel dataset:

- I manga analizzati **non mostrano una separazione naturale** in cluster ben definiti. Le feature disponibili, basate su generi (multilabel binari) e caratteristiche numeriche (punteggio, rank, popolarità), non sono sufficientemente discriminanti.
- L'algoritmo KMeans, che si basa sulla distanza euclidea e sull'ipotesi di cluster sferici e di dimensioni simili, tende a **collassare i dati in un unico cluster dominante** se non rileva confini chiari.
- La **riduzione dimensionale tramite PCA**, sebbene utile per la visualizzazione, può distorcere le relazioni nei dati originali, nascondendo sovrapposizioni reali tra i punti.

Questo comportamento è confermato da:

- **Silhouette score basso** per tutti i valori di k, anche quello ottimale;
- **Distribuzione sbilanciata dei manga** nei cluster, specialmente nel caso ottimizzato (k=2), dove la quasi totalità dei punti ricade in un solo gruppo.

Questi risultati indicano che la **struttura dei dati non è adatta a un clustering efficace con KMeans**, almeno con le feature attualmente disponibili.

Possibili alternative future includono l'uso di:

- algoritmi basati sulla **densità** (es. DBSCAN), più robusti in presenza di forme irregolari;
- tecniche **semantiche o embedding** per rappresentare i generi in modo più informativo;
- **clustering gerarchico**, che non richiede la scelta preventiva di k.

9 Risultati e confronto finale

9.1 Analisi simbolico vs statistico

Il progetto ha integrato due approcci distinti per l'analisi dei dati:

Apprendimento simbolico (basato su logica Prolog): costruisce raccomandazioni attraverso regole esplicite e inferenze deduttive;

Apprendimento statistico (basato su machine learning supervisionato e non supervisionato): apprende dai dati tramite pattern ricorrenti.

L'approccio simbolico si è dimostrato **più interpretabile**, poiché le raccomandazioni derivano direttamente da regole esplicite definite dall'utente o generate automaticamente a partire da dati strutturati. Tuttavia, è **meno flessibile** rispetto alle variazioni nei dati: non può generalizzare oltre le regole codificate.

Al contrario, i modelli supervisionati (es. Random Forest, XGBoost, AdaBoost) hanno mostrato una **maggiore capacità predittiva**, raggiungendo accuracies comprese tra 70% e 80%, con valori di precision e recall bilanciati. Questi modelli riescono a **catturare relazioni non evidenti** tra le feature e il target, ma a scapito della trasparenza e della spiegabilità.

Il clustering non supervisionato (KMeans), invece, ha prodotto **risultati più deboli**: i dati non presentano una struttura naturalmente clusterizzata, e i gruppi individuati dall'algoritmo non riflettono separazioni semantiche solide. Questo conferma che il dataset si presta meglio a una **classificazione supervisionata** piuttosto che a una segmentazione non guidata.

9.2 Riflessioni sui modelli

L'analisi supervisionata ha coinvolto sei modelli principali:

Decision Tree: interpretabile ma soggetto ad overfitting.

Random Forest: miglior compromesso tra accuratezza e stabilità.

AdaBoost: ha mostrato buoni risultati nei fold ma sensibile al rumore.

K-Nearest Neighbors: efficace ma rallenta su grandi dataset.

Naive Bayes: semplice e veloce, ma meno performante.

XGBoost: potente ma richiede attenzione nel tuning per evitare overfitting.

Le metriche aggregate su 5 fold (accuracy, precision, recall, F1-score) sono state sintetizzate in un radar plot comparativo. Da questa analisi è emerso che **Random Forest e XGBoost** sono i modelli con le performance più elevate, seguiti da AdaBoost. Naive Bayes, pur mostrando risultati più modesti, ha avuto il pregio di essere molto rapido e stabile.

Il confronto finale ha confermato che **modelli ensemble** (come Random Forest e XGBoost) offrono un equilibrio ideale tra accuratezza, robustezza e capacità di generalizzazione.

Infine, va sottolineato che **le performance sono state ottenute con feature semplici** (generi + metadati), dimostrando che anche un dataset moderatamente strutturato può essere informativo, purché analizzato con metodi adeguati.

10 Problemi e soluzione

10.1 Note tecniche sul flusso OAuth (MyAnimeList)

L'accesso programmato all'API ufficiale di MyAnimeList, previsto per l'integrazione dinamica dei dati utente, è stato compromesso da un blocco server-side introdotto successivamente allo sviluppo del sistema.
Il codice sviluppato implementa **correttamente l'intero flusso OAuth2 con PKCE**, secondo la guida ufficiale MyAnimeList:

- Generazione sicura del code_verifier;
- Uso del metodo plain per il code_challenge (accettato dall'API);
- Server HTTP locale per ricezione del code;
- Scambio code → access_token tramite chiamata POST correttamente parametrizzata.

Dal **1° maggio 2024**, MyAnimeList ha introdotto un blocco automatico tramite **AWS WAF (Web Application Firewall)** che:

- Impedisce le chiamate POST non validate verso l'endpoint /v1/oauth2/token;
- Richiede la **verifica CAPTCHA lato browser**, impedendo l'automazione completa del flusso.

Soluzione adottata:

- I dati necessari (top manga, lista utente) sono stati **raccolti prima dell'attivazione del blocco** e salvati nel progetto in formato .csv;
- Per evitare dipendenza da fonti instabili, si è deciso di lavorare su **dataset statici** e localmente disponibili.

Nota importante
La protezione introdotta da MyAnimeList **non invalida l'implementazione**:

1) Il codice è **funzionante** e conforme allo standard OAuth2;
2) Può essere **riattivato immediatamente** qualora il servizio rimuova o modifichi le restrizioni attuali.

Per test futuri o integrazioni dinamiche, il sistema è **già pronto**: basterà aggiornare il token e rilanciare lo script senza modifiche strutturali.

11 Conclusioni

11.1 Riepilogo del lavoro

Il progetto ha sviluppato un **sistema di raccomandazione ibrido** per manga, integrando approcci **simbolici** (regole logiche in Prolog) e **statistici** (modelli supervisionati e clustering). Il sistema è strutturato in modo modulare e include:

- Raccolta dati da MyAnimeList;
- Generazione automatica di una **knowledge base** Prolog;
- **Motore logico** per raccomandazioni esplicite;
- Sei **modelli supervisionati** per predire il gradimento utente;
- **Clustering KMeans** con selezione automatica di k;
- Analisi finale con grafici, matrici di confusione e radar plot.

L'integrazione dei due paradigmi ha permesso di coniugare **spiegabilità e potere predittivo**, offrendo un sistema intelligente capace di adattarsi e motivare le proprie scelte.

11.2 Estensioni future

Il sistema è stato progettato in modo estensibile, e sono possibili numerose evoluzioni:

- Sostituire la codifica binaria dei generi con **embedding semantici** (es. Word2Vec, BERT);
- Adottare algoritmi di clustering più robusti (es. **DBSCAN**, **clustering gerarchico**);
- Integrare una **web app interattiva** per l'esplorazione visiva della knowledge base;
- Riattivare l'integrazione dinamica con MyAnimeList tramite OAuth2, **qualora venga rimosso il blocco CAPTCHA**;
- Combinare classificatori in un **modello ensemble intelligente**, per sfruttare i punti di forza di ciascun approccio.

Questo lavoro rappresenta un esempio pratico e replicabile di intelligenza artificiale ibrida, utile sia in ambito accademico che professionale.

12 Appendice

12.1 Riferimenti

Fonti teoriche e accademiche

- **Poole, D., & Mackworth, A.** – *Artificial Intelligence: Foundations of Computational Agents*.
  Disponibile online: https://artint.info
  Rilevanti in questo progetto:
  - Rappresentazione della conoscenza e logica dei predicati;
  - Deduzione logica e Prolog;
  - Apprendimento supervisionato;
  - Clustering e apprendimento non supervisionato;
  - Sistemi ibridi e ragionamento simbolico + statistico.

Siti web e risorse esterne

- **MyAnimeList API**: https://myanimelist.net/apiconfig
- **SWI-Prolog**: https://www.swi-prolog.org/
- **Owlready2**: https://owlready2.readthedocs.io/
- **HermiT Reasoner**: https://www.hermit-reasoner.com/
- **scikit-learn**: https://scikit-learn.org/
- **XGBoost**: https://xgboost.ai/