# Gestione e raccomandazione di Playlist di Film

**Componenti del gruppo**

- Gloria Melchiorre, [MAT. 763612], g.melchiorre@studenti.uniba.it
- Nadia Rutigliano, [MAT 758449], n.rutigliano10@studenti.uniba.it

- **Link GitHub:** [Link al tuo progetto]

**A.A. 2024-2025**

## Sommario

# Introduzione

L'obiettivo di questo progetto è sviluppare un sistema per riorganizzare in modo intelligente le playlist di film esistenti e suggerire la playlist più appropriata per film non ancora classificati. Il sistema mira a creare raggruppamenti tematici coerenti (le "playlist") partendo dalle caratteristiche intrinseche dei film (clustering), per poi addestrare modelli in grado di automatizzare l'inserimento di nuovi elementi.

Questo progetto integra quattro paradigmi fondamentali di Intelligenza Artificiale per sviluppare un sistema di gestione intelligente di playlist cinematografiche:

1. **Apprendimento non supervisionato**: Clustering K-Means per identificare playlist tematiche
2. **Apprendimento supervisionato**: Classificazione automatica di nuovi film
3. **Ragionamento probabilistico**: Rete Bayesiana per modellare dipendenze
4. **Ragionamento logico**: Knowledge Base Prolog per inferenze simboliche

Il sistema trasforma un dataset di film in un motore di raccomandazione che combina approcci statistici e simbolici, dimostrando l'integrazione di diversi modelli di ragionamento.

**Dataset:** MovieLens 100k, preprocessato per ottenere 18 feature binarie di genere, rating normalizzato e categorizzato, epoca di rilascio.

**Implementazione:** Python 3.11 con

- matplotlib: visualizzazione dei grafici (curve di apprendimento, grafici a barre, grafici a torta)
- networkx: visualizzazione di grafi (usato per osservare la struttura della rete bayesiana)
- numpy: per il calcolo numerico
- pandas: per la manipolazione dei dati
- scikit_learn: per il clustering e la classificazione
- pgmpy: creazione della rete bayesiana
- pyswip: per l'integrazione con Prolog
- imblearn: per la gestione dello sbilanciamento delle classi (SMOTE)

# Capitolo 1: Creazione e Preprocessing del Dataset

Il punto di partenza è il dataset MovieLens 100k. Per preparare i dati alle fasi successive, è stato eseguito un preprocessing (data_preprocessing.py) per aggregare i dati a livello di film e ingegnerizzare le feature necessarie.

Il dataset finale, a seguito del preprocessing, include le seguenti feature chiave utilizzate nelle analisi successive:

- Feature di Genere: 18 colonne binarie (0 o 1) che indicano l'appartenenza di un film a un genere specifico (es. Action, Comedy, Drama).
  La categorizzazione facilita l'interpretazione nella BN e nelle regole Prolog, mantenendo l'informazione quantitativa per gli algoritmi di clustering
- Feature di Rating:
  - rating_norm: Il rating medio di un film, normalizzato in un intervallo [0, 1], per uniformità negli algoritmi.
  - rating_discreto: Versione categorica del rating, suddivisa in tre livelli (0: basso, 1: medio, 2: alto).
- Feature Temporali:
  - release_year_categoria: Versione categorica dell'anno di rilascio, suddivisa in tre epoche (0: vecchi, 1: intermedi, 2: recenti).
  Esclusione di release_year dal clustering. I test preliminari con feature temporali producevano cluster basati sull'epoca ("film vecchi" vs "recenti") anziché sul contenuto tematico

Queste trasformazioni sono state fondamentali per l'applicazione successiva degli algoritmi.

# Capitolo 2: Apprendimento non Supervisionato: Creazione delle Playlist

**Metodologia e Parametri K-Means**

**Configurazione algoritmo:**

- Inizializzazione: k-means++ per convergenza stabile
- random_state=42, max_iter=300 per riproducibilità
- Esclusione esplicita delle feature temporali per concentrarsi sul contenuto

**Metodologia**

In una prima fase esplorativa, il clustering era stato fortemente influenzato dalle feature temporali, portando a cluster basati sull'epoca di rilascio ("film vecchi" vs "film recenti") piuttosto che sul contenuto. Per superare questa limitazione, l'approccio è stato raffinato escludendo esplicitamente le feature release_year e release_year_categoria. L'algoritmo si è quindi concentrato esclusivamente sui generi e sul rating normalizzato per identificare gruppi tematici basati sul contenuto.

**Determinazione del Numero Ottimale di Cluster (k)**

Per identificare il numero ottimale di cluster (k) per l'algoritmo K-Means, è stato applicato il Metodo del Gomito (Elbow Method). Questa tecnica analizza la variazione dell'inerzia totale (WCSS - Within-Cluster Sum of Squares) al variare del numero di cluster. L'inerzia rappresenta la somma delle distanze quadrate tra ogni punto dati e il centro del proprio cluster; un valore più basso indica cluster più compatti.

Come mostrato in Figura 1, il grafico dell'inerzia in funzione di k rivela un chiaro "gomito" in corrispondenza di k=4. Per valori di k inferiori a 4, l'aggiunta di un nuovo cluster provoca una riduzione significativa dell'inerzia, indicando che si sta catturando una struttura importante nei dati. Per valori di k superiori a 4, la curva si appiattisce notevolmente, e i benefici derivanti dall'aggiunta di ulteriori cluster diventano marginali.

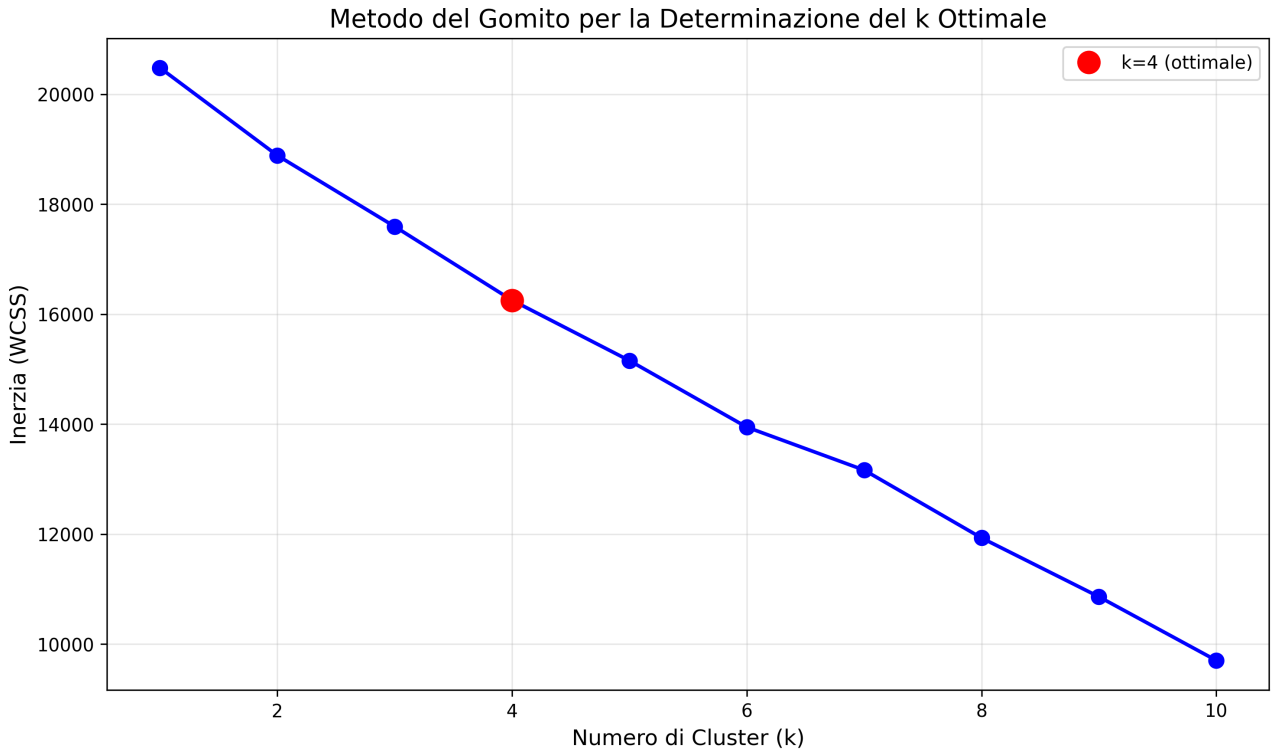


*Figura 1 Metodo del Gomito per la determinazione di k.** Il grafico mostra un "gomito" evidente a k=4, indicando il punto ottimale di compromesso tra compattezza e complessità.*

Il punto k=4 rappresenta quindi il miglior compromesso tra la massimizzazione della compattezza dei cluster e la minimizzazione della complessità del modello. Pertanto, è stato scelto k=4 come numero di cluster per l'analisi successiva.

**Validazione della scelta k=4:**

- Elbow Method mostra chiara inflessione a k=4
- Silhouette Score: 0.31 (accettabile per dataset reale)
- Interpretabilità: 4 cluster semanticamente distinti e bilanciati

**Risultati del Clustering**

L'analisi ha prodotto quattro cluster tematici ben distinti e significativi, come mostrato nel file cluster_analysis.txt.

- **Cluster 0 (Commedie Mainstream):** Un cluster di 307 film (28.5% dei dati), quasi interamente dominato dal genere Comedy (99.7%). Presenta un rating medio di 3.06.
- **Cluster 1 (Drammatici e Thriller):** Il cluster più grande con 588 film (54.5% dei dati). Rappresenta i generi più diffusi e acclamati come Drama (65.0%) e Thriller (21.8%), con un rating medio elevato di 3.31.
- **Cluster 2 (Azione, Avventura e Fantascienza):** Un gruppo consistente di 150 film (13.9% dei dati), caratterizzato da generi come Adventure (70.0%), Action (57.3%) e Sci-Fi (46.7%). Il rating medio è di 3.07.
- **Cluster 3 (Film d'Animazione e per Bambini):** Un piccolo cluster di nicchia (3.1% dei dati, 33 film), fortemente specializzato nei generi Animation (100%) e Children's (81.8%). Ha il rating medio più alto del gruppo (3.31), suggerendo film di alta qualità.

La distribuzione dei film nei quattro cluster è mostrata nel grafico a torta seguente.

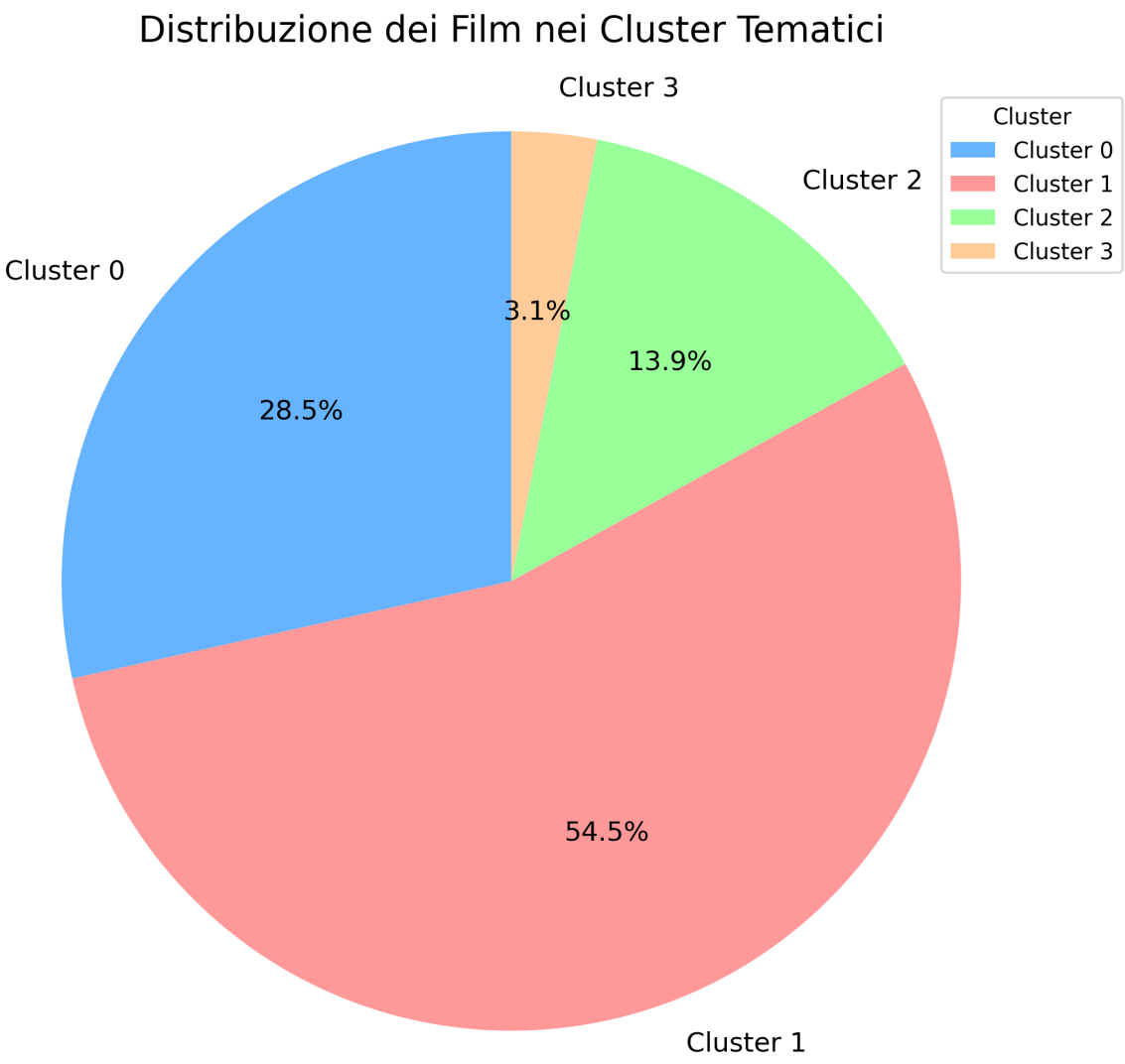


*Figura 2: Distribuzione dei film nei cluster. Si nota la dominanza del cluster 1 (Drammatici e Thriller), mentre gli altri cluster rappresentano gruppi tematici più specifici o di nicchia.*

# Capitolo 3: Apprendimento Supervisionato: Classificazione dei Film

L'obiettivo di questa fase era duplice:

1. Validare la qualità dei cluster creati: Se i cluster fossero ben separati, un modello di classificazione dovrebbe essere in grado di predire l'appartenenza di un film con alta accuratezza.
2. Creare modelli predittivi: Addestrare classificatori in grado di assegnare automaticamente nuovi film alle playlist tematiche corrette.

**Metodologia e validazione**

Per evitare il "data leakage" e ottenere una valutazione realistica, è stata adottata una metodologia rigorosa:

1. Separazione netta (Train/Test Split): il dataset è stato diviso in training set (80%) e test set (20%) prima di qualsiasi operazione di clustering o addestramento. Questo garantisce che il test set rimanga un "mondo sconosciuto" per la valutazione finale.
2. Addestramento esclusivo su dati di training: il modello K-Means è stato addestrato esclusivamente sul training set. Le etichette dei cluster per il test set sono state predette usando questo modello, simulando l'arrivo di nuovi dati.
3. Gestione dello sbilanciamento: poiché i cluster hanno dimensioni molto diverse, sul training set è stata applicata la tecnica di oversampling SMOTE per creare un set di dati bilanciato e prevenire che i modelli ignorassero i cluster più piccoli.
4. Confronto con una Baseline: Per contestualizzare le performance, i modelli sono stati confrontati con un classificatore naive (DummyClassifier) che predice sempre la classe più frequente. Questo stabilisce il livello minimo di performance da superare.

**Gestione Sbilanciamento Classi**

Problema: Forte sbilanciamento tra i cluster, es. Cluster 1 (588 film) vs Cluster 3 (33 film). Soluzione: SMOTE applicato solo sul training set.

- Parametri: k_neighbors=5, random_state=42
- **Motivazione:** Prevenire bias verso classi maggioritarie senza contaminare test set

Giustificazione della scelta dei parametri:

- k_neighbors=5: Questo è il valore di default e uno standard de-facto nella letteratura per l'algoritmo SMOTE. Indica che per creare un nuovo campione sintetico di una classe minoritaria, l'algoritmo considererà i suoi 5 vicini più prossimi. Un valore troppo basso (es. 1 o 2) potrebbe creare campioni sintetici troppo simili a quelli esistenti, mentre un valore troppo alto potrebbe iniziare a considerare vicini di altre classi, creando rumore. Il valore 5 rappresenta un buon compromesso tra la generazione di diversità e la conservazione della coerenza locale del cluster.
- random_state=42: È stato impostato un seme fisso per garantire la riproducibilità dei risultati. Senza questo, ogni esecuzione di SMOTE produrrebbe un training set leggermente diverso, rendendo i confronti tra i modelli meno affidabili.

**Risultati e Analisi Critica**

I modelli addestrati (Decision Tree, Random Forest, Logistic Regression) hanno mostrato performance eccellenti e realistiche, come riassunto nella tabella seguente (dati estratti da classification_results.txt):

| Modello | F1-Score (Macro) | Accuratezza |
|---|---|---|
| Baseline (Dummy) | 0.1240 | 33.0% |
| Decision Tree | 0.9411 | 98.5% |
| **Random Forest** | **0.9681** | **98.9%** |
| Logistic Regression | 0.9654 | 98.9% |

L'F1-Score, che bilancia precision e recall, si attesta intorno al 97%per i modelli migliori. Questo risultato eccezionale, confrontato con una baseline molto bassa, valida l'ipotesi iniziale: i cluster identificati nella fase precedente sono ben separati, distinti e semanticamente coerenti.

La matrice di confusione del modello RandomForest, il migliore per F1-score, illustra chiaramente il numero quasi nullo di errori.

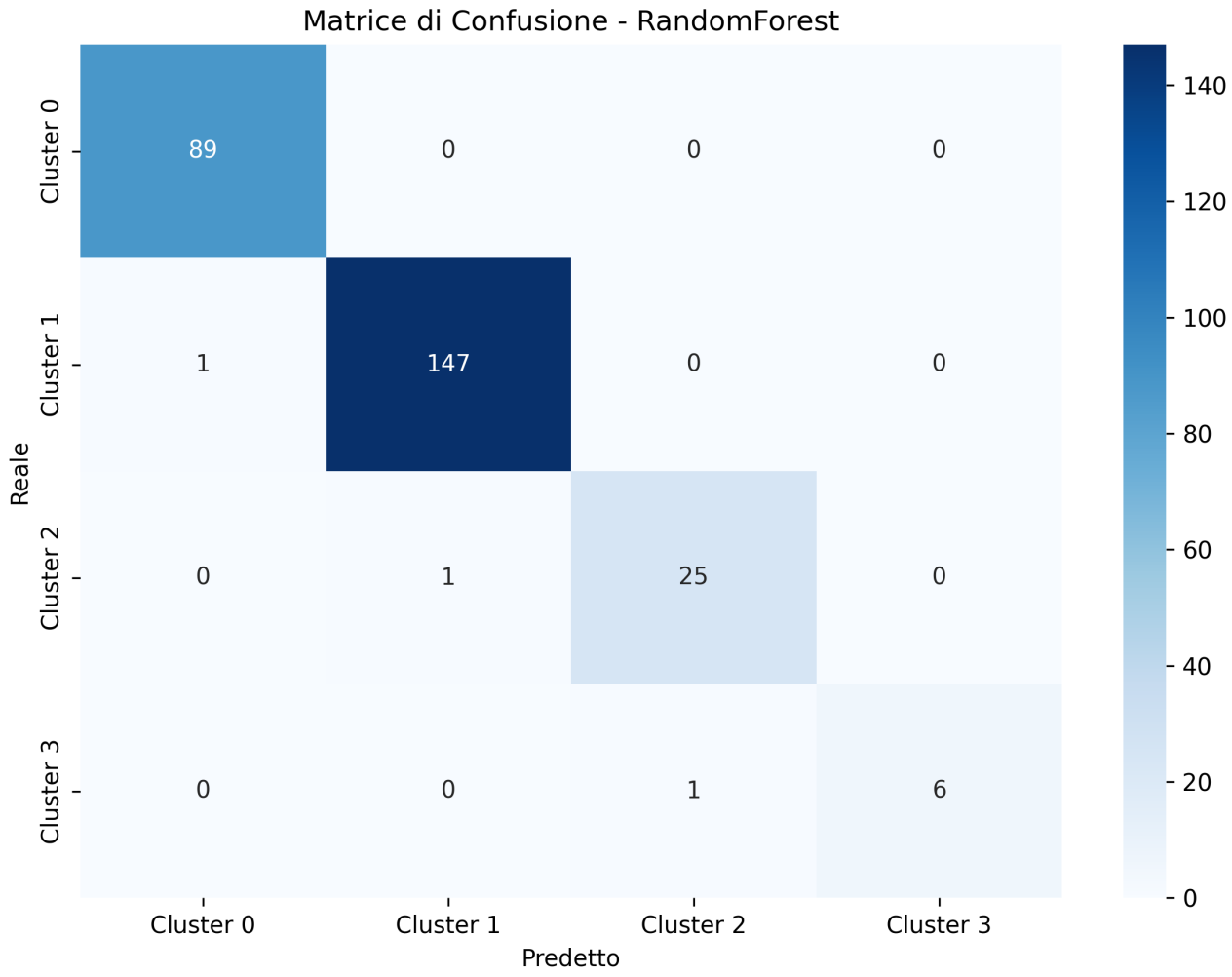


*Figura 3 Matrice di Confusione (Random Forest). Si notano pochissimi errori di classificazione, a dimostrazione dell'alta separabilità dei cluster tematici creati.*

L'analisi dell'importanza delle feature conferma l'intuizione derivata dal clustering: i generi che definiscono le nicchie (Animation, Comedy, Adventure, Sci-Fi) sono i predittori più potenti per identificare i rispettivi cluster.

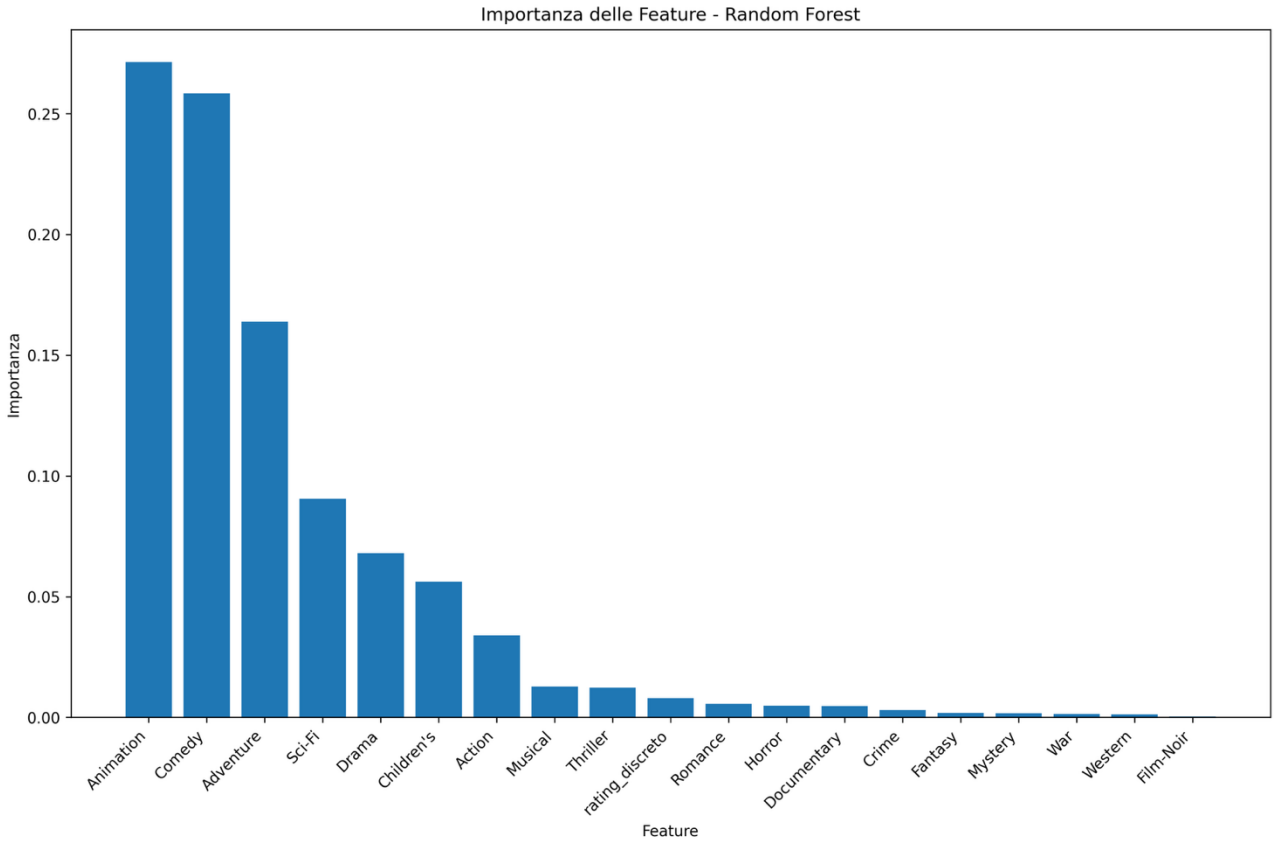


*Figura 4 Importanza delle Feature (Random Forest). I generi più specifici come Animation e Comedy risultano i più discriminanti, seguiti da Adventure e Sci-Fi, confermando la natura tematica dei cluster.*

# Capitolo 4: Ragionamento Probabilistico e Rete Bayesiana

Per esplorare le dipendenze probabilistiche tra le variabili in modo più trasparente, è stata costruita una Rete Bayesiana (BN). A differenza dei classificatori, che sono modelli discriminativi, la BN è un **modello generativo** che apprende la distribuzione di probabilità congiunta di tutte le variabili.

**Algoritmo di apprendimento struttura:**

- Hill Climb Search con BIC score
- Motivazione BIC: Penalizza complessità eccessiva, evita overfitting
- Training esclusivamente su training set

**Variabili nel modello:**

- 18 generi binari + rating_discreto + release_year_categoria + cluster
- Decisione: Mantenimento di tutte le variabili per catturare dipendenze complesse

**Metodologia e Struttura Appresa**

Il ragionamento probabilistico è una forma di ragionamento che sfrutta la teoria della probabilità, in particolar modo dipendenza e indipendenza tra variabili e regola di Bayes.

La struttura della rete è stata appresa esclusivamente dal training set tramite l'algoritmo HillClimbSearch con score BIC, per evitare l'overfitting. Il grafo risultante mostra una struttura di dipendenze ricca e plausibile.

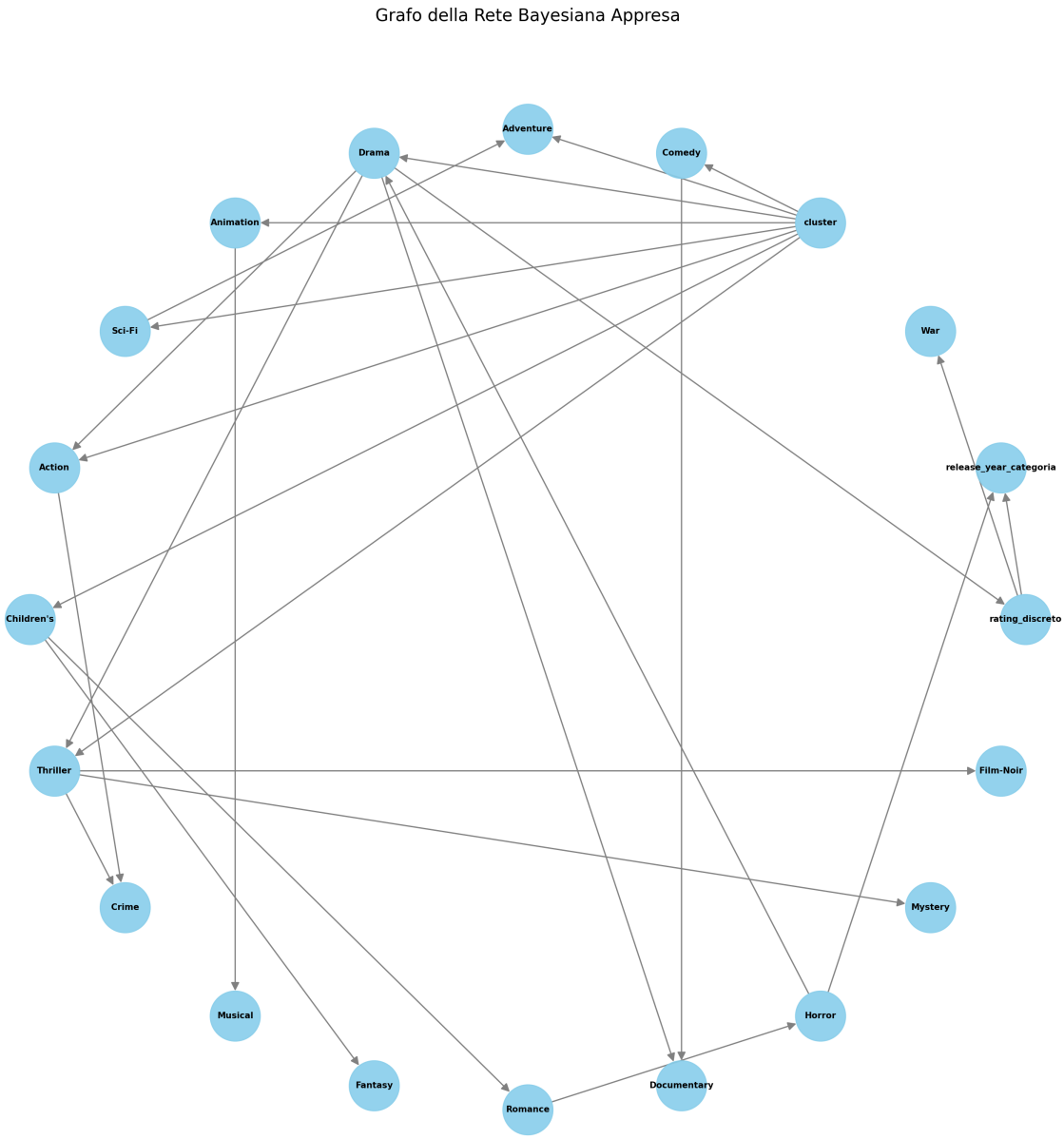


*Figura 5 Grafo della Rete Bayesiana Appresa. Il grafo mostra le relazioni di dipendenza apprese dal modello.*

L'analisi della struttura appresa, visibile nel grafo e nei file di output, rivela diverse dipendenze significative:

- **Hub Centrali e Loro Ruolo:** I nodi con il maggior numero di connessioni (cluster, Drama, Thriller) agiscono come "hub" informativi. Dramae Thriller, essendo generi molto diffusi, influenzano o sono influenzati da molti altri nodi, confermando il loro ruolo centrale nel panorama cinematografico. Il nodo cluster è un hub "ricevente", che aggrega informazioni da molti generi per determinare l'appartenenza tematica.
- **Dipendenze Dirette e Intuitive:** La rete ha appreso relazioni attese. Ad esempio, la CPD (Conditional Probability Distribution) tra Animatione cluster mostra che P(cluster=3 | Animation=1) = 1.0, una dipendenza deterministica che cattura perfettamente la natura del Cluster 3 (Film d'Animazione). Similmente, l'arco War -> rating_discreto indica che i film di guerra tendono ad avere rating più alti, un pattern plausibile nei dati.
- **Indipendenze Condizionali Rivelatrici:**L'assenza di archi è altrettanto informativa. Il criterio BIC, penalizzando la complessità, ha evitato di creare un arco diretto tra release_year_categoria e Drama. Questo suggerisce che, una volta che conosciamo altre variabili (come il cluster o altri generi), l'anno di uscita non fornisce informazioni aggiuntive significative sul fatto che un film sia drammatico. Questo è un esempio di indipendenza condizionale appresa automaticamente dal modello.
- **Relazioni complesse tra generi:** Si notano archi come Thriller -> Crime o Comedy -> Documentary. Queste non sono necessariamente relazioni causali, ma dipendenze statistiche. Un arco A -> B significa che conoscere il valore di A cambia la nostra credenza sul valore di B. Ad esempio, sapere che un film è un Thriller potrebbe aumentare la probabilità che sia anche un Crime.

**Valutazione e Inferenza**

*La BN è stata valutata sul test set per la sua capacità di predire il cluster, ottenendo un'accuratezza del 41.1%. Questo risultato, come previsto, è inferiore non solo a quello dei classificatori specializzati (~98%), ma anche alla baseline della classe maggioritaria (54.8%). Questo non è un difetto del modello, ma una caratteristica intrinseca: la BN non è ottimizzata per la massima accuratezza predittiva, ma per modellare fedelmente la distribuzione di probabilità congiunta di tutte le variabili. Il suo vero valore non risiede nella classificazione, ma nell'interpretabilità e nella flessibilità inferenziale, come dimostrato dagli esempi.*

La rete può essere usata per un ragionamento del tipo:

- **Query:** Qual è il cluster più probabile per un film Azione, Sci-Fi e con rating_discreto alto (2)?
- **Risposta della Rete:** La probabilità che appartenga al Cluster 2 (Azione, Avventura e Fantascienza) è del 97.5%, un'inferenza logica e precisa.

Questo dimostra la capacità della BN di modellare le relazioni complesse tra tema, valutazione e raggruppamento tematico.

**Discussione sulla Scelta degli Iperparametri**

In questo progetto, sono state adottate le seguenti strategie:

- **Ricerca a Griglia (Grid Search) per Classificatori:** Per i modelli di classificazione (Decision Tree, Random Forest, Logistic Regression), è stata impiegata una GridSearchCV con 5-fold cross-validation. Questo approccio testa sistematicamente un'ampia gamma di combinazioni di parametri, scegliendo quella che massimizza una metrica di performance (F1-Score macro), garantendo una scelta ottimale e basata sui dati. I parametri ottimali sono riportati nel file classification_results.txt.
- **Valori Standard e Best Practice (K-Means, SMOTE):** Per algoritmi come K-Means e SMOTE, sono stati utilizzati valori standard basati sulla letteratura e sulle best practice. Ad esempio, k_neighbors=5 per SMOTE è un valore di default robusto che bilancia la generazione di diversità e la coerenza. Una potenziale analisi di sensibilità (sviluppo futuro) potrebbe esplorare come la performance del classificatore finale varia al cambiare di k_neighbors (es. [3, 5, 7, 10]), per confermare se k=5 sia effettivamente la scelta ottimale anche per questo specifico dataset.
- **Criteri Informativi per la Selezione del Modello (Rete Bayesiana):** Per l'apprendimento della struttura della Rete Bayesiana, la scelta non è un iperparametro ma un criterio di scoring. È stato scelto il Bayesian Information Criterion (BIC)perché penalizza la complessità del modello in modo più aggressivo rispetto ad altri score (es. K2), riducendo il rischio di overfitting e favorendo una rete più sparsa e interpretabile.

# Capitolo 5: Ragionamento Logico con Prolog

Per superare i limiti dei modelli statistici e abilitare un ragionamento simbolico e trasparente, è stata costruita una Knowledge Base (KB) in Prolog. Questo approccio trasforma i dati grezzi e i risultati del machine learning in una base di conoscenza strutturata e interrogabile in modo dichiarativo.

**Architettura KB Multi-Livello**

1. **Livello 1 - Fatti Base:** La KB viene popolata dinamicamente tramite lo script prolog_interface.py con informazioni dettagliate per ogni film, includendo non solo il **cluster**, ma anche il **rating** e l'**epoca di uscita**.

```
57          # Fatti base più ricchi
58          prolog.assertz(f"film_info({prolog_title}, {genres_list_str}, {cluster_val})")
59          prolog.assertz(f"film_rating({prolog_title}, {rating_discrete}, {rating_norm})")
60          prolog.assertz(f"film_era({prolog_title}, {year_cat})")
```

2. **Integrazione di Conoscenza di Fondo (Background Knowledge):** È stata definita una tassonomia gerarchica di generi (macro_genere), che raggruppa generi specifici in categorie semantiche più ampie (es. 'action' e 'thriller' appartengono alla macro-categoria azione). Questa conoscenza, non presente nei dati originali, permette un ragionamento più astratto e flessibile.

```
62    # --- FASE 2: DEFINIZIONE TASSONOMIE E GERARCHIE ---
63    print("Definizione tassonomie di generi...")
64
65    # Gerarchia di generi - Macro-categorie
66    prolog.assertz("macro_genere('action', azione)")
67    prolog.assertz("macro_genere('adventure', azione)")
68    prolog.assertz("macro_genere('thriller', azione)")
69    prolog.assertz("macro_genere('crime', azione)")
```

3. **Regole Complesse e Multi-criterio:** Sono state implementate diverse regole di ragionamento che operano su più livelli, combinando i risultati del clustering con la conoscenza di fondo:
    - consiglia_film_qualita/3: Suggerisce film dello stesso cluster, ma solo se hanno un rating uguale o superiore.
    - consiglia_cross_cluster/3: Aumenta la serendipità delle raccomandazioni, cercando film di alta qualità in cluster diversi ma con generi "semanticamente affini" secondo la tassonomia.
    - tipo_film/2: Classifica i film in categorie qualitative come 'blockbuster' o 'film_d_autore' basandosi su una combinazione di genere e rating.
    - raccomandazione_forte/3: Esegue un meta-ragionamento, assegnando un punteggio di "forza" a una raccomandazione in base al numero di generi condivisi e al rating del film consigliato.

**Complessità Computazionale della KB**

Regole implementate:

1. Pattern matching semplice: O(n) per film_info/3
2. Ragionamento con vincoli: $O(n^2)$ per consiglia_film_qualita/3
3. Ragionamento gerarchico: $O(n^2 \times m)$ dove m = numero macro-generi
4. Meta-ragionamento: $O(n^3)$ per raccomandazione_forte/3

Backtracking Prolog: Le regole sfruttano il backtracking per esplorare tutte le soluzioni possibili, non solo la prima.

**Dimostrazione del Ragionamento Deduttivo**

L'efficacia di questo approccio è stata dimostrata tramite una suite di query specifiche eseguite dallo script run_prolog_queries.py. I risultati evidenziano la capacità del sistema di andare oltre il semplice recupero di dati.

- **Query di Base (Pattern Matching):** Per verificare la corretta asserzione dei fatti nella Knowledge Base, è stata eseguita una query di pattern matching per recuperare tutti i film appartenenti al Cluster 0. Il sistema ha recuperato correttamente 396 film, dimostrando che i dati sono stati caricati con successo.

```
--- 1. Query di base: Trova film nel Cluster 0 ---

>>> Esecuzione query Prolog: film_info(Titolo, Generi, 0)
Trovati 396 risultati.
Primi 3 risultati:
 - Film: get_shorty, Generi: ['action', 'comedy', 'drama']
 - Film: babe, Generi: ['childrens', 'comedy', 'drama']
 - Film: mighty_aphrodite, Generi: ['comedy']
```

- **Query di Ragionamento Avanzato (Raccomandazione di Qualità):** La regola consiglia_film_qualita/3 implementa un ragionamento con vincoli. Oltre a trovare film nello stesso cluster di pulp_fiction, applica un vincolo aggiuntivo sul rating (deve essere uguale o superiore). Come mostrato nell'output, la regola suggerisce film acclamati come professional_the (Léon) e fargo, filtrando altre opzioni con rating inferiore.

```
--- 2. Raccomandazioni base per 'pulp_fiction' ---

>>> Esecuzione query Prolog: consiglia_film_base('pulp_fiction', FilmConsigliato)
Trovati 33 risultati.
Prime 3 raccomandazioni base:
 - copycat
 - professional_the
 - carlitos_way
```

- **Query con Ragionamento Gerarchico (Cross-Cluster 3 Macro-genreri):** Sfruttando la tassonomia di macro_genere(Background Knowledge), il sistema può eseguire un ragionamento gerarchico. La query consiglia_cross_cluster/3 trova film di alta qualità in cluster diversi ma con generi "semanticamente affini". Inoltre, è possibile interrogare direttamente la gerarchia per trovare tutti i film appartenenti a una macro-categoria, come "azione".

```
--- 3. Raccomandazioni di qualità superiore per 'pulp_fiction' ---

>>> Esecuzione query Prolog: consiglia_film_qualita('pulp_fiction', FilmConsigliato, Motivo)
Trovati 11 risultati.
Raccomandazioni di qualità:
 - professional_the (Motivo: stesso_cluster_qualita_superiore)
 - fargo (Motivo: stesso_cluster_qualita_superiore)
 - godfather_the (Motivo: stesso_cluster_qualita_superiore)
```

```
--- 10. Analisi macro-generi ---
Film di azione nel dataset:

>>> Esecuzione query Prolog: macro_genere(Genere, azione), film_info(Film, Generi, _), member(Genere, Generi)
Trovati 657 risultati.
Trovati 450 film di azione. Primi 5:
 - terminal_velocity
 - willy_wonka_and_the_chocolate_factory
 - surviving_the_game
 - menace_ii_society
 - original_gangstas
```

- **Query di Classificazione Intelligente:** La KB implementa regole euristiche per classificare i film in categorie qualitative astratte, come 'blockbuster' o 'film_d_autore', basandosi su una combinazione di genere e rating. L'output mostra come pulp_fiction venga correttamente identificato come film_d_autore, dimostrando una comprensione del suo contenuto che va oltre i dati grezzi.

```
--- 5. Che tipo di film è 'pulp_fiction'? ---

>>> Esecuzione query Prolog: tipo_film('pulp_fiction', Tipo)
Trovati 1 risultati.
 - pulp_fiction è classificato come: film_d_autore
```

- 

- **Query di Meta-Ragionamento (Punteggio di Raccomandazione):** La regola raccomandazione_forte/3 esegue un meta-ragionamento, ovvero un ragionamento sulla qualità della raccomandazione stessa. Assegna un punteggio di "forza" a ogni film suggerito, basandosi sul rating e sul numero di generi in comune. Questo permette di non solo trovare raccomandazioni, ma anche di ordinarle in base alla loro pertinenza.

```
--- 9. Raccomandazioni più forti per 'pulp_fiction' (con punteggio) ---

>>> Esecuzione query Prolog: raccomandazione_forte('pulp_fiction', FilmConsigliato, Punteggio)
Trovati 39 risultati.
Top 3 raccomandazioni forti:
 - copycat (Punteggio: 2)
 - professional_the (Punteggio: 4)
 - carlitos_way (Punteggio: 2)
```

**Analisi Critica dei Risultati**

I risultati dimostrano la capacità della KB di eseguire ragionamenti complessi, integrando i dati del clustering per fornire risposte intelligenti e contestualizzate. L'analisi degli output rivela due caratteristiche importanti del ragionamento logico.

In primo luogo, alcune query producono risultati duplicati (es. get_shorty nelle raccomandazioni cross-cluster). Questo non è un errore, ma una caratteristica del motore di Prolog che, grazie al backtracking, trova tutte le possibili vie deduttiveper arrivare a una soluzione. Un film può essere raccomandato più volte se soddisfa i criteri attraverso percorsi di ragionamento diversi (es. tramite due generi affini distinti).

In secondo luogo, la combinazione dei dati di input con la conoscenza di fondo può portare a classificazioni inaspettate ma logicamente corrette. Ad esempio, Willy Wonka & the Chocolate Factory viene classificato nella macro-categoria "azione" perché è etichettato come "Adventure" nel dataset originale, e la nostra tassonomia raggruppa "Adventure" sotto "Azione".

Questi comportamenti, sebbene possano richiedere un post-processing per l'utente finale (es. rimozione dei duplicati), evidenziano la trasparenza e la completezza del ragionamento logico. In sintesi, la componente Prolog trasforma l'output della pipeline di machine learning in un sistema di conoscenza attivo, capace di inferenze flessibili e spiegabili che sarebbero impossibili per i soli modelli statistici.

# Capitolo 6: Conclusioni e Sviluppi Futuri

Questo progetto ha dimostrato con successo la fattibilità e l'efficacia di una pipeline integrata che combina diverse tecniche di IA per analizzare e organizzare un catalogo di film. I passaggi cruciali sono stati:

- **Creazione di cluster tematicamente coerenti** tramite K-Means, fornendo la base per la successiva organizzazione.
- **Adozione di una metodologia di validazione robusta** (train/test split, baseline, analisi degli errori) che ha permesso di addestrare classificatori con un'elevata e realistica accuratezza (~98%), validando la qualità dei cluster.
- **Integrazione di modelli interpretabili** come la Rete Bayesiana e una Knowledge Base Prolog che, insieme ai classificatori, costituiscono un potente e sfaccettato sistema di conoscenza.

*Sviluppi Futuri Possibili:*

- **Esplorare Algoritmi di Clustering Alternativi:** Testare algoritmi come DBSCAN, che non richiede di pre-specificare il numero di cluster, potrebbe rivelare strutture diverse nei dati.
- **Apprendimento Online o Periodico:** Trasformare la pipeline in un sistema dinamico che si aggiorna periodicamente con nuovi dati, mantenendo i modelli sempre attuali.

- **Interfaccia Utente:** Sviluppare una semplice interfaccia grafica (es. con Streamlit o Flask) per permettere a un utente di interagire con il sistema, cercando film e ricevendo raccomandazioni generate da Prolog.

## Riferimenti bibliografici

Poole, D., & Mackworth, A. (2017). Artificial Intelligence: Foundations of Computational Agents. Cambridge University Press.

Link al dataset MovieLens 100k: https://grouplens.org/datasets/movielens/100k/

Scikit-learn documentation: https://scikit-learn.org/

pgmpy documentation: https://pgmpy.org/

pyswip documentation: https://github.com/yuce/pyswip