UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

A cura di

- Ilenia Sasanelli, 739529, i.sasanelli1@studenti.uniba.it
- Matteo Belforte, 760396, m.belforte1@studenti.uniba.it

REPOSITORY

https://github.com/Isasanelli/code__icon_23-24

PROGETTO:

Classificazione e Raccomandazione di Film e Serie TV su Netflix

AA 2023/2024

Sommario

INTRODUZIONE 3

L'Impatto di Netflix sull'Informatica e il Machine Learning 6

1. *Raccomandazioni Personalizzate* 6
2. *Produzione di Contenuti* 6
3. *Etica e Bias nel Machine Learning* 6
4. *Considerazioni finali* 7

PRESENTAZIONE PROGETTO 8

DESCRIZIONE DEL DOMINIO E DATI UTILIZZATI 8

1. Presentazione del Dataset 8
2. Caratteristiche del Dataset 8
3. Ragioni per la Scelta del Dataset 9
4. Limitazioni del Dataset 9

STRUTTURA E FUNZIONALITÀ DEL PROGETTO 10

Struttura del progetto 10

Classificazione 10

1. *Preprocessing dei dati* 11
2. *Creazione degli embeddings* 13
3. *Analyze_data* 15
4. *Apprendimento Supervisionato* 23
5. *Cross Validation* 34
6. *Confronto tra i modelli* 37

Knowledge Base (KB) 41

1. *Panoramica e Motivazioni* 41
2. *Struttura e Contenuto della KB* 41
3. *Esempio di Regole* 43
4. *Processo di Implementazione* 43
5. *Risultati e Impatto* 45

Ricerca e Raccomandazione 46

1. *Ricerca e raccomandazione* 46
2. *Visualizzazione dei titoli più popolari* 49

Librerie utilizzate 51

CONCLUSIONI 53

RIFERIMENTI BIBLIOGRAFICI 54

Introduzione

Netflix rappresenta una delle piattaforme di streaming più influenti a livello globale. Fondata nel 1997 da Reed Hastings e Marc Randolph a Scott Valley, California[1], l'azienda ha avuto un'evoluzione straordinaria nel corso degli anni.

Figura 1

Immaginate di poter accedere a una vasta biblioteca di film e serie TV con un semplice clic: questo è ciò che Netflix offre oggi ai suoi utenti.

Inizialmente, Netflix era concepita come un servizio di noleggio di DVD via posta. Gli abbonati sceglievano i titoli online e ricevevano i dischi direttamente a casa. Questo modello innovativo già all'epoca eliminava le fastidiose penali per i ritardi nella restituzione, tipiche dei videonoleggi tradizionali. Tuttavia, la vera rivoluzione era ancora in arrivo.

Con l'avvento della banda larga e il miglioramento delle tecnologie di streaming, Netflix ha rapidamente trasformato il proprio modello di business. L'azienda ha intuito il potenziale dello streaming video online e ha investito massicciamente in questa direzione. Questo passaggio ha segnato l'inizio di una nuova era nell'intrattenimento domestico, rendendo Netflix un vero e proprio colosso dello streaming video.

Attualmente, Netflix vanta oltre 200 milioni di abbonati distribuiti in tutto il mondo. Questo numero impressionante testimonia quanto il servizio sia diventato parte integrante della vita quotidiana di molte persone.

Netflix non è più solo una piattaforma di streaming, ma è diventato sinonimo di intrattenimento on-demand[Fig:2], rivoluzionando il modo in cui gli utenti accedono e fruiscono dei contenuti video.

Figura 2

L'impatto di Netflix sul panorama dell'intrattenimento va ben oltre i numeri. La piattaforma ha cambiato le abitudini di visione del pubblico, introducendo il concetto di "binge-watching".

Il "Binge-watching"[Fig 3] significa guardare molti episodi di una serie TV in modo consecutivo, di solito in un breve lasso di tempo.

È come se "divorassi" una serie TV tutta d'un fiato. Spesso questo implica guardare più episodi consecutivi senza fare pause o dedicarsi ad altre attività.

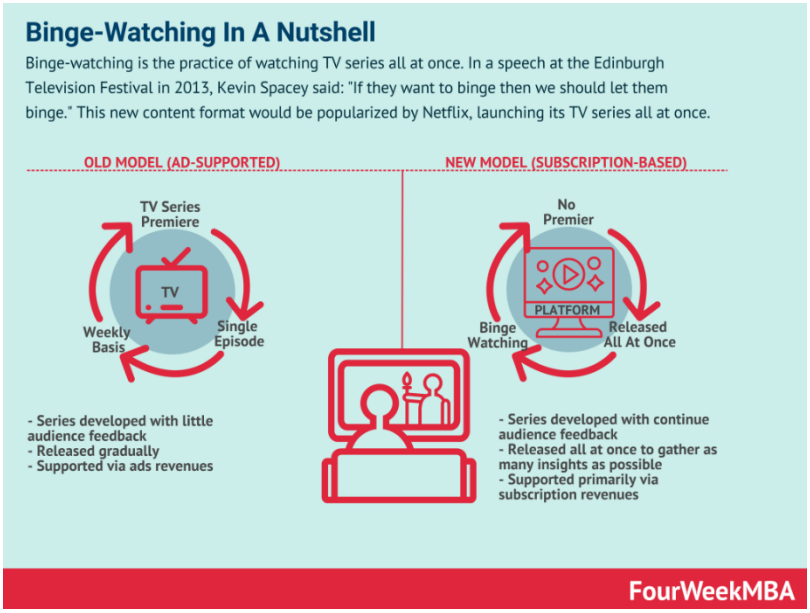


Figura 3

La piattaforma, inoltre, producendo contenuti originali di alta qualità che hanno spesso definito le tendenze culturali globali, con la sua interfaccia user-friendly[Fig 4] e gli algoritmi di raccomandazione personalizzati, Netflix continua a plasmare il futuro dell'intrattenimento digitale, offrendo un'esperienza su misura per ogni utente.

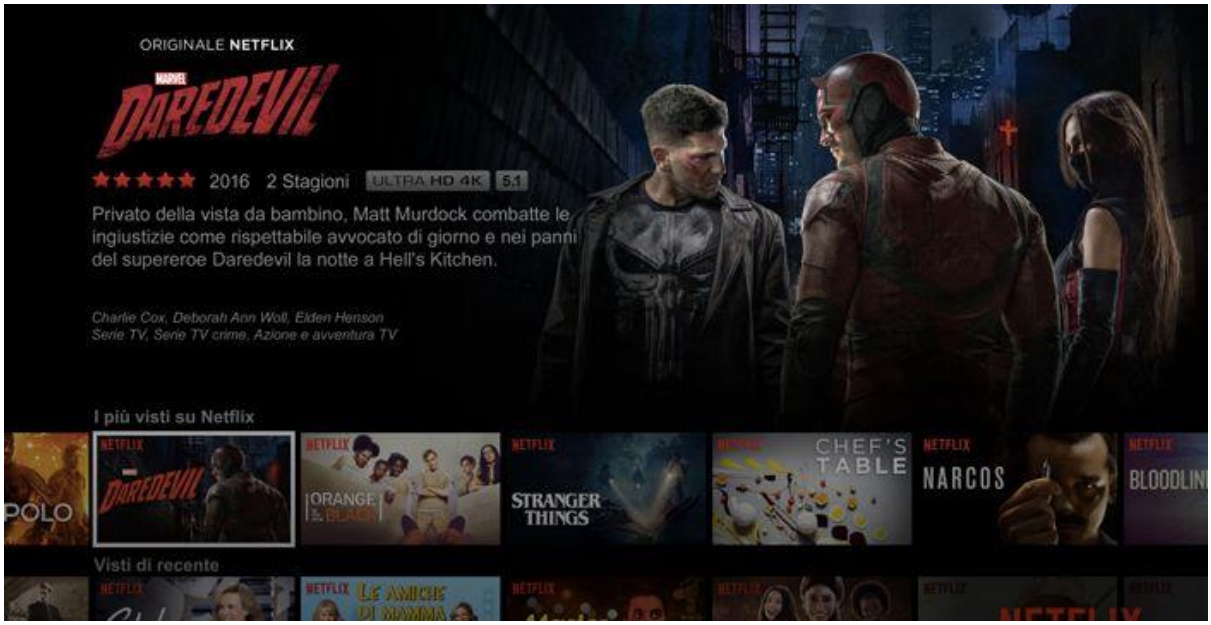


Figura 4

L'Impatto di Netflix sull'Informatica e il Machine Learning

Netflix, uno dei principali servizi di streaming a livello globale, ha avuto un impatto significativo sul campo dell'informatica e, più specificamente, sullo sviluppo e l'applicazione del machine learning (ML). Questo impatto si manifesta attraverso l'implementazione di tecniche avanzate di ML per migliorare l'esperienza utente, ottimizzare le operazioni aziendali e guidare l'innovazione tecnologica.

1. Raccomandazioni Personalizzate

Uno degli esempi più evidenti dell'influenza di Netflix nell'informatica è il suo sistema di raccomandazione, che utilizza algoritmi di machine learning per personalizzare l'esperienza visiva degli utenti.

Utilizzando tecniche di ***collaborative filtering***, ***deep learning*** e ***content based recommender systemstent,*** Netflix analizza il comportamento e le preferenze degli utenti per suggerire contenuti rilevanti. Questo non solo migliora la soddisfazione degli utenti, ma aumenta anche il tempo di visualizzazione e la fidelizzazione del cliente, elementi cruciali per il modello di business di Netflix.

2. Produzione di Contenuti

Netflix utilizza anche machine learning per informare la produzione di contenuti. Attraverso l'analisi dei dati raccolti dalle visualizzazioni, Netflix è in grado di identificare tendenze emergenti, preferenze regionali e gap di mercato, che guidano le decisioni su quali tipi di contenuti produrre. Questo approccio ***data-driven*** alla produzione di contenuti rappresenta un cambiamento significativo rispetto ai metodi tradizionali basati sull'intuizione o sulle opinioni degli esperti.

3. Etica e Bias nel Machine Learning

Infine, l'approccio di Netflix al machine learning solleva anche importanti questioni etiche. L'algoritmo di raccomandazione, ad esempio, potrebbe rinforzare pregiudizi esistenti, promuovendo una selezione di contenuti che potrebbe escludere o marginalizzare alcuni gruppi o opinioni. Netflix ha affrontato questa sfida sviluppando metodi per mitigare i bias

algoritmici e garantire che le raccomandazioni siano equamente rappresentative delle diverse preferenze degli utenti.

4. Considerazioni finali

In conclusione, Netflix si è affermata come una delle piattaforme di streaming più influenti al mondo, rivoluzionando l'industria dell'intrattenimento domestico. Partendo da un servizio di noleggio DVD via posta, l'azienda ha saputo evolversi rapidamente abbracciando le tecnologie di streaming e raggiungendo oltre 200 milioni di abbonati globali. Il suo impatto va oltre i numeri, cambiando le abitudini di visione del pubblico con il "binge-watching" e la produzione di contenuti originali di qualità. Netflix ha anche avuto un'influenza significativa nel campo dell'informatica e del machine learning, implementando sistemi di raccomandazione personalizzati, utilizzando l'analisi dei dati per guidare la produzione di contenuti e affrontando questioni etiche legate ai bias negli algoritmi. Attraverso queste innovazioni, Netflix continua a plasmare il futuro dell'intrattenimento digitale, offrendo un'esperienza su misura e ponendosi all'avanguardia nell'applicazione delle tecnologie informatiche nel settore dei media.

Presentazione progetto

Abbiamo realizzato un sistema innovativo basato su raccomandazione e classificazione per film e serie TV su Netflix, progettato per offrire un'esperienza utente altamente personalizzata e coinvolgente.

Il nostro obiettivo è aiutare gli utenti a navigare in modo efficiente nell'ampio catalogo di contenuti disponibili, facilitando la scoperta di nuovi film e serie TV che rispondano ai loro gusti e preferenze specifiche.

Il cuore del sistema è basato su sofisticati algoritmi di machine learning che analizzano in profondità i dati relativi alle preferenze di visione degli utenti. Attraverso un'interfaccia interattiva e intuitiva, gli utenti potranno esprimere le loro valutazioni sui titoli, indicando il livello di gradimento.

Sulla base di queste valutazioni, il sistema genererà raccomandazioni mirate, che non solo tengono conto dei titoli preferiti, ma anche dei generi e delle categorie di contenuti verso i quali l'utente mostra maggiore interesse.

Descrizione del Dominio e Dati Utilizzati

1. Presentazione del Dataset

Il dataset utilizzato nel nostro progetto proviene dalla piattaforma *Kaggle*, ed è stato scaricato dalla seguente fonte: Netflix Shows Dataset[1]. Questo dataset è stato scelto per la sua rilevanza e completezza in relazione all'obiettivo del progetto, che è la costruzione di un sistema di analisi e raccomandazione per la piattaforma di streaming Netflix.

2. Caratteristiche del Dataset

Il dataset contiene un'ampia gamma di informazioni sui titoli presenti sulla piattaforma Netflix. Le principali caratteristiche del dataset includono:

- **show_id**: Un identificatore univoco per ciascun titolo.
- **type**: Indica se il titolo è un film ("Movie") o una serie TV ("TV Show").
- **title**: Il titolo del contenuto.

- **director**: Il nome del regista del film o della serie.
- **cast**: L'elenco degli attori principali.
- **country**: Il paese di produzione del contenuto.
- **date_added**: La data in cui il titolo è stato aggiunto alla piattaforma Netflix.
- **release_year**: L'anno di rilascio del titolo.
- **rating**: La classificazione del contenuto (ad esempio, PG, R, etc.).
- **duration**: La durata del contenuto, espressa in minuti per i film e in stagioni per le serie TV.
- **listed_in**: Le categorie o i generi in cui il contenuto è classificato.
- **description**: Una breve descrizione del contenuto.

3. Ragioni per la Scelta del Dataset

Il dataset è stato scelto per diverse ragioni:

1. **Completezza**: Fornisce una copertura completa dei contenuti disponibili su Netflix, con informazioni dettagliate che sono fondamentali per costruire un sistema di raccomandazione accurato.
2. **Rilevanza**: Essendo specifico per Netflix, il dataset è perfettamente allineato con l'obiettivo del progetto, che mira a sviluppare un sistema di raccomandazione per una piattaforma di streaming.
3. **Qualità dei Dati**: I dati presenti nel dataset sono strutturati e ben documentati, facilitando le operazioni di preprocessing e l'implementazione di modelli di machine learning.

4. Limitazioni del Dataset

Nonostante le sue numerose qualità, il dataset presenta alcune limitazioni:

- **Dati mancanti**: Alcune colonne presentano valori mancanti, in particolare per campi come "director" e "cast". Questo può ridurre la capacità del modello di fare raccomandazioni basate su tali attributi.

- **Varianza nei Generi**: La categorizzazione dei contenuti sotto "listed_in" può essere ridondante o sovrapposta, il che potrebbe introdurre complessità aggiuntive nella fase di analisi.

Struttura e Funzionalità del progetto

I moduli di codice sviluppati per questo progetto costituiscono il nucleo operativo del sistema di raccomandazione. Ogni file è dedicato a una specifica fase del processo, dalla preparazione dei dati all'implementazione del machine learning, fino alla generazione delle raccomandazioni finali. Questa struttura modulare consente una gestione efficiente e una facile estendibilità del sistema, assicurando al contempo che ogni componente sia ottimizzato per svolgere la propria funzione in modo indipendente e preciso.

Struttura del progetto

Il progetto è stato strutturato in modo da ottimizzare l'organizzazione del codice e facilitare lo sviluppo.

Abbiamo suddiviso il codice in due parti principali: una dedicata alla **classificazione** e l'altra alla **ricerca e raccomandazione**. Questa suddivisione permette di gestire separatamente le diverse fasi del processo, garantendo che ciascuna componente possa essere sviluppata, testata e ottimizzata in modo indipendente, ma al contempo integrato all'interno del sistema complessivo.

Classificazione

La prima parte del progetto è incentrata sull'apprendimento supervisionato e sulla classificazione. In questa sezione, il nostro obiettivo principale è addestrare modelli di machine learning per classificare i contenuti in base a diverse caratteristiche, come le preferenze degli utenti, i generi, e le valutazioni.

I file chiave in questa parte del progetto includono:

- **preprocess_data.py**: Si occupa della pulizia e preparazione dei dati per l'addestramento dei modelli.

- create_embedding.py: Genera embeddings per rappresentare i titoli e le categorie di contenuti in uno spazio vettoriale continuo, che vengono poi utilizzati nei modelli di classificazione.
- supervised.py: Gestisce l'addestramento dei modelli di classificazione utilizzando tecniche come Random Forest, Decision Tree, e XGBoost.
- cross_validation.py: Implementa la validazione incrociata per valutare le performance dei modelli, assicurando che essi generalizzino bene sui dati non visti.

1. Preprocessing dei dati

Il preprocessing dei dati rappresenta una fase critica e preliminare in qualsiasi progetto di data science e machine learning. Esso consiste in una serie di operazioni destinate a trasformare dati grezzi, spesso incompleti o non strutturati, in un formato pulito e coerente, adatto per essere utilizzato nei modelli di machine learning. La qualità dei dati utilizzati è un fattore determinante per le performance e l'accuratezza dei modelli predittivi, e il preprocessing svolge un ruolo chiave in questo processo

Perché il Preprocessing è importante?

L'importanza del preprocessing non può essere sottovalutata. I dati grezzi sono spesso imperfetti: possono contenere errori, valori mancanti, duplicati, o semplicemente essere in un formato non adeguato all'analisi ed è per questo che il preprocessing viene richiamato per migliorare la qualità dei dati in modo da ridurre gli errori, migliorare la capacità del modello di generalizzare e, in definitiva, produrre risultati più affidabili.

Ad esempio, nel nostro progetto, il dataset utilizzato contiene informazioni dettagliate su film e serie TV disponibili su Netflix. Tuttavia, come spesso accade con i dataset reali, questi dati presentavano delle sfide, tra cui valori mancanti in colonne chiave, duplicati, e variabili non strutturate. Senza un adeguato preprocessing, questi problemi avrebbero potuto compromettere la qualità delle raccomandazioni generate dal modello.

Sono state, dunque, applicate un serie di operazioni chiavi, ognuna mirata a risolvere eventuali problematiche nel dataset.

Le operazioni si articolano in:

Pulizia dei Dati:

- Rimozione dei Duplicati: Uno dei primi passi nel preprocessing è stato quello di rimuovere i duplicati dal dataset. La presenza di record duplicati può introdurre bias nel modello, poiché influisce sulla rappresentazione dei dati e può portare a raccomandazioni ridondanti.
- Eliminazione delle Colonne Non Rilevanti: Colonne come **description** e **duration**, pur contenendo informazioni potenzialmente utili, non erano direttamente rilevanti per gli obiettivi del nostro modello di raccomandazione e sono state rimosse per semplificare il dataset

Gestione dei Valori:

- Sostituzione dei Valori Mancanti: Le colonne con valori mancanti, come **title, cast, director, rating** e **release_year**, sono state trattate riempiendo i vuoti con valori predefiniti, come **"Unknown Title"** o "**Unrated**" o con valore 0 (caso di **release_year**). Questa scelta evita che i valori mancanti possano causare errori durante l'addestramento del modello, mantenendo al contempo una certa coerenza nei dati.

Creazione di Nuove Variabili:

- Unione di Colonne: Abbiamo creato una nuova variabile**, content_category,** che combina le informazioni presenti nelle colonne **type** e **listed_in.** Questo ha permesso di semplificare l'analisi e migliorare la capacità del modello di identificare le categorie di contenuti che meglio si adattano alle preferenze dell'utente.

  Inoltre, è stata aggiunta una colonna **preferences,** con valori simulati, per testare il sistema di raccomandazione con un attributo che rappresenta le preferenze degli utenti.Questa preferenza viaggia con valori randomici da 40 a 100.

Creazione del File CSV Preprocessato

Durante la fase di preprocessing, tutte le operazioni applicate ai dati grezzi vengono consolidate e salvate in un nuovo file CSV, denominato **processed_data.csv**.

Questo file rappresenta la versione pulita, trasformata e ottimizzata del dataset originale, ed è il punto di partenza per tutte le analisi successive nel progetto.

Differenze fra dataset

1. *Netflix Titles Dataset*

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
1	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson		United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmmaker Kirsten Johnson stages his death in inventive
2	s2	TV Show	Blood & Water		Ama Qamata, Khosi Ngema, Gail Mabalane, Thabang Molaba, Dillon Windvogel,	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town teen sets out to prove whether a private-school
3	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabiha Akkari, Sofia Lesaffre, Salim		September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Action & Adventure	To protect his family from a powerful drug lord, skilled thief Mehdi and his expert
4	s4	TV Show	Jailbirds New Orleans				September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	Feuds, flirtations and toilet talk go down among the incarcerated women at the
5	s5	TV Show	Kota Factory		Mayur More, Jitendra Kumar, Ranjan Raj, Alam Khan, Ahsaas Channa, Revathi	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV Comedies	In a city of coaching centers known to train India's finest collegiate minds, an earnest but

2. *Processed Data*

Processed Dataset

	show_id	title	director	cast	country	date_added	release_year	rating	content_category	preferences
1	s1	Dick Johnson Is Dead	Kirsten Johnson	Unknown Cast	United States	September 25, 2021	2020	PG-13	Movie - Documentaries	41
2	s2	Blood & Water	Unknown Director	Ama Qamata, Khosi Ngema, Gail Mabalane, Thabang Molaba, Dillon Windvogel,	South Africa	September 24, 2021	2021	TV-MA	TV Show - International TV Shows, TV Dramas, TV Mysteries	75
3	s3	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabiha Akkari, Sofia Lesaffre, Salim		September 24, 2021	2021	TV-MA	TV Show - Crime TV Shows, International TV Shows, TV Action & Adventure	63
4	s4	Jailbirds New Orleans	Unknown Director	Unknown Cast		September 24, 2021	2021	TV-MA	TV Show - Docuseries, Reality TV	40
5	s5	Kota Factory	Unknown Director	Mayur More, Jitendra Kumar, Ranjan Raj, Alam Khan, Ahsaas Channa, Revathi	India	September 24, 2021	2021	TV-MA	TV Show - International TV Shows, Romantic TV Shows, TV Comedies	98

2. Creazione degli embeddings

Il processo di generazione degli embeddings è un passaggio cruciale nell'analisi dei dati testuali di Netflix. Gli embeddings sono rappresentazioni vettoriali dense che catturano le relazioni semantiche tra parole o frasi, permettendo di trasformare dati testuali in formato numerico utilizzabile dai modelli di machine learning.

Gli embeddings contribuiscono a migliorare l'efficienza computazionale riducendo la dimensionalità dei dati. Ad esempio, invece di utilizzare una matrice di dimensioni enormi per rappresentare una grande quantità di categorie (come nel caso del one-hot encoding), gli embeddings permettono di lavorare con vettori di dimensioni molto più ridotte, riducendo il carico computazionale e permettendo l'uso di modelli più complessi senza sacrificare la velocità di calcolo.

Nei sistemi di raccomandazione, gli embeddings svolgono un ruolo cruciale nel personalizzare le raccomandazioni per gli utenti. Ad esempio, nel contesto del nostro progetto, gli embeddings vengono utilizzati per rappresentare film e serie TV in un modo che riflette le loro caratteristiche e relazioni con altri contenuti.

Questo permette al modello di identificare rapidamente contenuti simili e di suggerirli agli utenti in base alle loro preferenze precedenti.

Come Vengono Generati gli Embeddings?

Il processo di generazione degli embeddings nel nostro progetto è stato implementato utilizzando il linguaggio Python e la libreria SpaCy, un potente strumento per il natural language processing (NLP).

Le fasi di generalizzazione sono suddivide in:

Caricamento dei Dati Preprocessati:

- Il primo passo consiste nel caricare il dataset preprocessato (processed_data.csv), che contiene i dati già puliti e trasformati. Questo file è essenziale perché fornisce le basi per generare embeddings coerenti e rilevanti

Caricamento del Modello SpaCy:

- Viene caricato un modello linguistico preaddestrato di SpaCy, come `en_core_web_sm`, che è in grado di trasformare testo in rappresentazioni vettoriali (embeddings). SpaCy è scelto per la sua efficienza e per la qualità degli embeddings che genera, basati su un'ampia gamma di testi

Generazione degli Embeddings:

- Gli embeddings vengono generati per specifiche colonne del dataset, in particolare per **title** e **content_category.**

  Il testo presente in queste colonne viene passato attraverso il modello di SpaCy, che lo trasforma in un vettore numerico. Se un testo non ha un embedding valido (ad esempio, per testi troppo brevi o non riconoscibili), viene assegnato un vettore di zeri

Salvataggio degli Embeddings:

- Una volta generati, gli embeddings vengono salvati in file NumPy (.npy), che possono essere facilmente caricati e utilizzati nelle fasi successive del progetto. Ad esempio, i file "title_embeddings.npy " e "content_category_embeddings.npy" contengono i vettori che rappresentano i titoli e le categorie di ogni film e serie tv.

3. Analyze_data

Il file analyze_data.py è un modulo Python progettato per eseguire l'analisi esplorativa dei dati e generare visualizzazioni che sintetizzano le informazioni chiave presenti nel dataset preprocessato. Questo modulo svolge un ruolo fondamentale nel comprendere le caratteristiche principali del dataset, fornendo insight utili che possono guidare le fasi successive del progetto, come la costruzione di modelli di machine learning o la formulazione di raccomandazioni.

Funzionalità Principali del Modulo

1. Caricamento e Preparazione dei Dati

Il file analyze_data.py inizia caricando il dataset preprocessato (processed_data.csv) utilizzando la libreria Pandas. Questo dataset, generato nella fase di preprocessing, contiene dati puliti e trasformati, pronti per essere analizzati e visualizzati.

2. Generazione di Grafici di Distribuzione

Il modulo offre diverse funzioni per la creazione di grafici di distribuzione, come plot_distribution.

Questi grafici sono essenziali per visualizzare come i dati sono distribuiti rispetto a vari attributi, come l'anno di rilascio (release_year), la categoria di contenuto (content_category), e il tipo di contenuto (type).

- **Distribuzione per Anno di Rilascio**: I grafici possono essere creati per visualizzare la distribuzione dei film e delle serie TV rilasciati per decennio, offrendo una panoramica storica della produzione dei contenuti.
- **Distribuzione per Categoria di Contenuto**: Un altro esempio è la distribuzione delle categorie di contenuto, che mostra la frequenza relativa di diversi generi o tipi di contenuti presenti nel dataset
- **Standardizzazione dei Dati**: La funzione `apply_standard_scaler` utilizza `StandardScaler` dalla libreria Scikit-Learn per standardizzare i dati.

```python
def apply_standard_scaler(X):
    scaler = StandardScaler()
    return scaler.fit_transform(X)
```

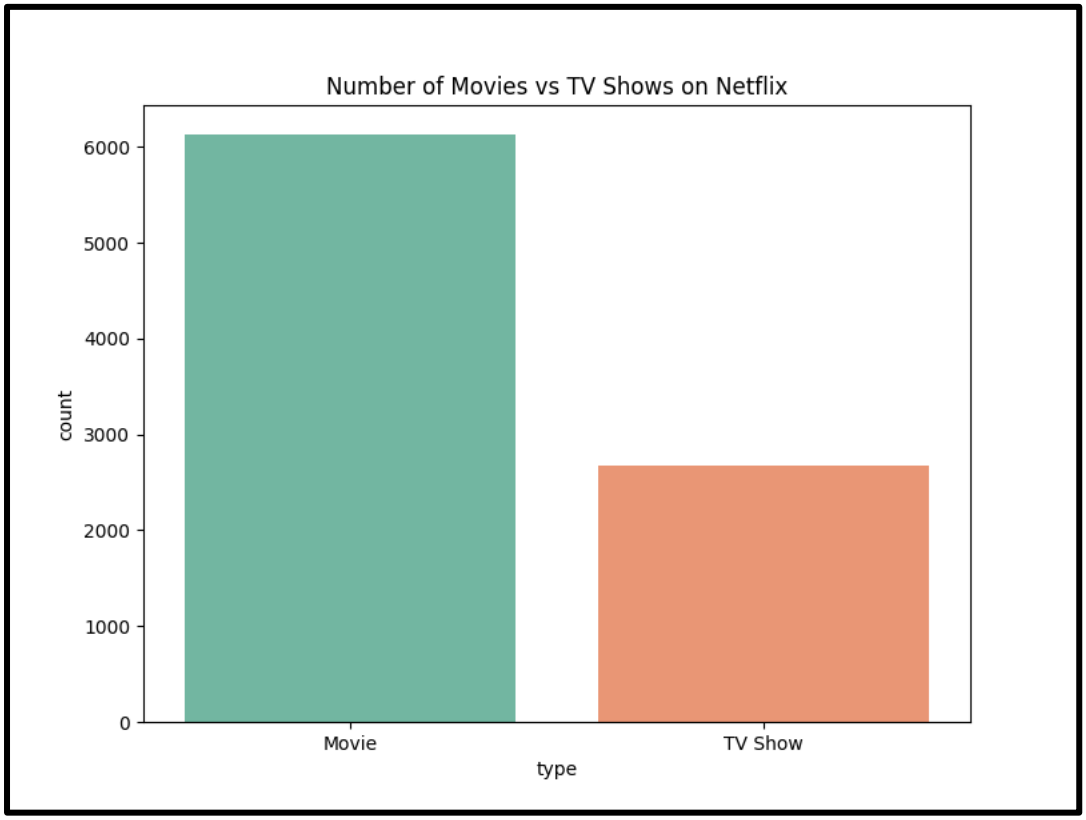
  Questo passaggio è spesso necessario prima di applicare tecniche di machine learning, poiché la standardizzazione garantisce che i dati abbiano una distribuzione con media zero e varianza unitaria, riducendo l'influenza di scale diverse tra le variabili

- **File Generati e Collegamenti con Altri Moduli**

Il file analyze_data.py genera diverse documenti che sono cruciali per il progetto:

1. **Grafici Salvati**: Tutti i grafici generati, come le distribuzioni per anno di rilascio o le categorie di contenuto, vengono salvati in formato PNG nella directory dei risultati (../results/visualizations/analyze_data). Questi grafici possono essere utilizzati per report, presentazioni, o per guidare decisioni durante lo sviluppo del modello.
2. **File CSV di Statistiche**: Il file CSV (../result/visualizations/popular_statistics.csv) contenente le statistiche riassuntive è salvato nella stessa directory e può essere utilizzato in fasi successive del progetto, come l'ottimizzazione dei modelli di machine learning o la personalizzazione delle raccomandazioni.

Spiegazione grafici



Il grafico presentato è un istogramma che confronta il numero di film (Movie) rispetto al numero di serie TV (TV Show) disponibili sulla piattaforma Netflix. Il grafico ha l'obiettivo di visualizzare la distribuzione di questi due tipi di contenuti all'interno del catalogo di Netflix.

Dati Presentati

- Asse X (Tipo di Contenuto): L'asse orizzontale del grafico rappresenta le due categorie principali di contenuti presenti su Netflix: *Movie* (Film) e *TV Show* (Serie TV).
- Asse Y (Conteggio): L'asse verticale indica il numero di titoli disponibili per ciascuna categoria.

Risultati

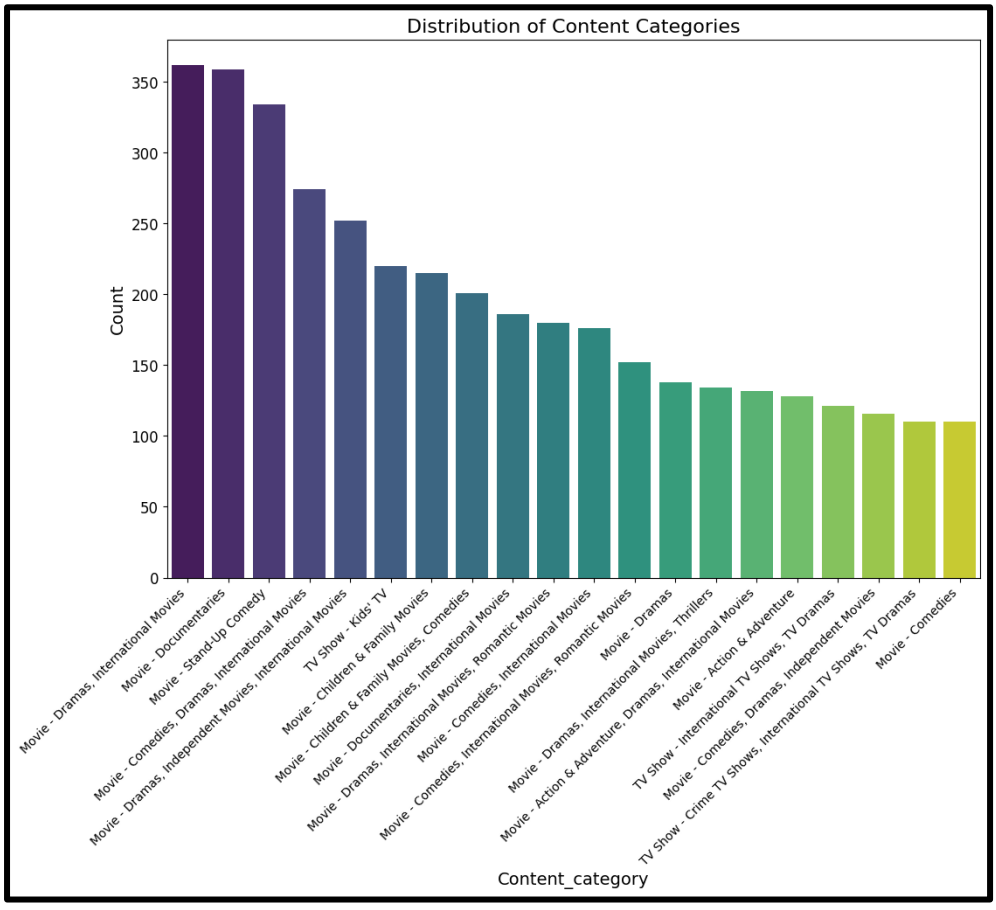
- Film (Movie): Il grafico mostra che i film costituiscono la maggior parte del catalogo di Netflix, con un numero di titoli che supera le 6.000 unità.
- Serie TV (TV Show): Le serie TV sono meno numerose rispetto ai film, con un conteggio che si aggira intorno alle 3.000 unità.

Dal grafico emerge chiaramente che il numero di film disponibili su Netflix è significativamente maggiore rispetto al numero di serie TV. Questo risultato suggerisce che, sebbene Netflix offra una vasta gamma di serie TV popolari e di successo, i film continuano a rappresentare la parte preponderante del catalogo. Questo dato può essere utile per comprendere le preferenze di programmazione della piattaforma e per analizzare come Netflix potrebbe bilanciare la sua offerta di contenuti per soddisfare le diverse preferenze degli utenti.

Distribuzione delle Categorie di Contenuto

Il grafico rappresentato è un istogramma che visualizza la distribuzione dei contenuti presenti su Netflix in base alla loro categoria.

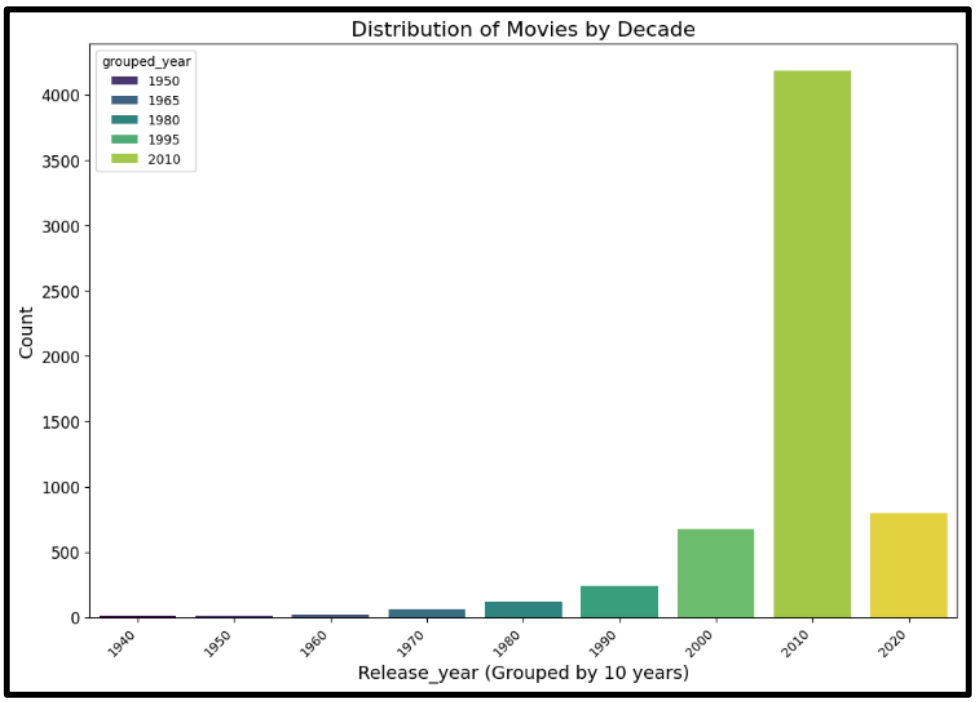
Le categorie combinano informazioni sul tipo di contenuto (ad esempio, Film o Serie TV) e il genere (come Dramma, Commedia, Documentario).



- Osservazioni:
    - Le categorie più rappresentate sono "Movie - Dramas, International Movies" e "Movie - Documentaries", con ciascuna che conta oltre 350 titoli. Questo indica un'abbondanza di film drammatici e documentari, specialmente con un focus internazionale.
    - Altre categorie come "Stand-Up Comedy" e "TV Show - Kids' TV" sono anch'esse ben rappresentate, indicando che Netflix offre una vasta gamma di contenuti per diverse fasce d'età e preferenze.
    - Le categorie con meno titoli, come "TV Show - Crime " e "Movie - Comedies", suggeriscono che queste aree sono meno esplorate o meno richieste dagli utenti.

Distribuzione dei Film per Decennio

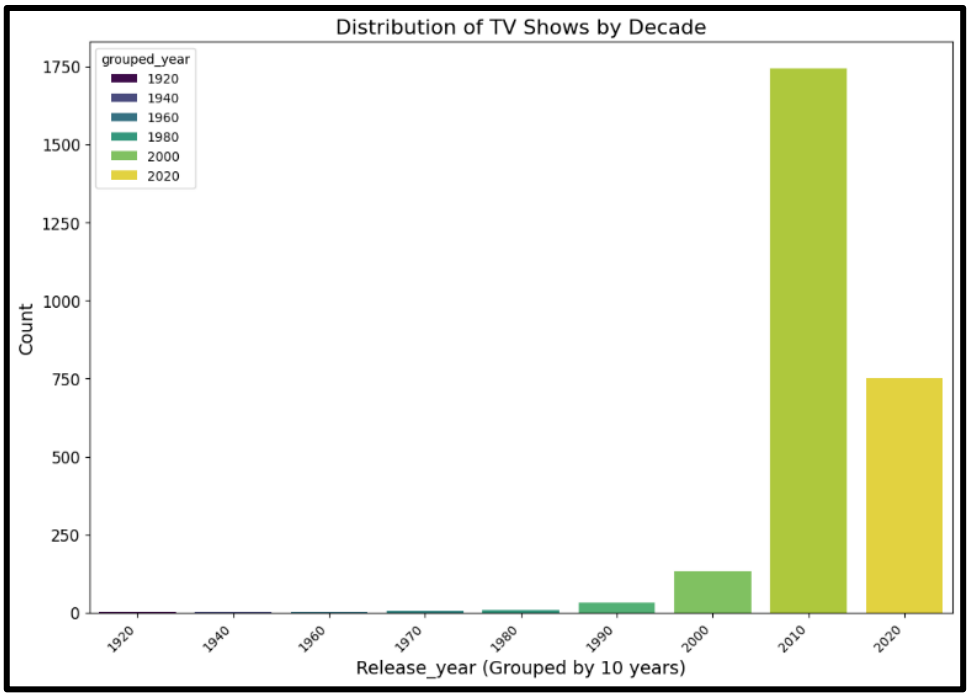
Il grafico mostra la distribuzione dei film disponibili su Netflix raggruppati per decennio di rilascio.



- Osservazioni:
    - La maggior parte dei film disponibili su Netflix è stata rilasciata nel decennio 2010-2020, con oltre 4.000 titoli. Questo riflette una concentrazione su contenuti moderni e recenti.
    - C'è una crescita significativa nel numero di film a partire dal decennio 2000-2010, che segna l'inizio di una produzione cinematografica più intensa.
    - I decenni precedenti al 2000 hanno un numero molto ridotto di titoli, il che suggerisce che Netflix è principalmente orientato verso la distribuzione di film recenti piuttosto che classici.

Distribuzione delle Serie TV per Decennio

Il grafico rappresenta la distribuzione delle serie TV su Netflix, raggruppate anch'esse per decennio di rilascio



- Osservazioni:
    - Analogamente ai film, il decennio 2010-2020 domina la distribuzione delle serie TV, con circa 1.750 titoli. Questo evidenzia l'importanza crescente delle serie TV nella programmazione di Netflix.
    - Le serie TV mostrano una crescita notevole a partire dagli anni 2000, segnalando una tendenza relativamente recente verso la produzione di questo tipo di contenuto.
    - I decenni precedenti agli anni '80 presentano pochissime serie TV, sottolineando l'accento su contenuti più moderni e la relativa scarsa rappresentazione delle serie storiche nel catalogo.

Confronto tra i Grafici

Categorie vs Anni di Rilascio

- Distribuzione Tematica vs Temporale:
    - Mentre il primo grafico si concentra sulla varietà di contenuti offerti da Netflix in termini di categorie (combinazione di genere e tipo di contenuto), i successivi due grafici analizzano la distribuzione temporale dei film e delle serie TV.
    - La distribuzione per categorie mostra che Netflix offre una vasta gamma di contenuti, con una forte presenza di film drammatici e documentari, ma include anche una rappresentanza significativa di serie TV dedicate a specifiche fasce di età, come i bambini. Dall'altra parte, i grafici temporali evidenziano un catalogo pesantemente orientato verso titoli moderni, soprattutto quelli rilasciati dopo il 2000.

Film vs Serie TV

- Predominanza dei Film:
    - In entrambi i grafici temporali, il decennio 2010-2020 è il più rappresentato, sia per i film che per le serie TV. Tuttavia, il numero totale di film è

notevolmente superiore a quello delle serie TV, specialmente negli anni recenti.

- Le serie TV hanno iniziato a guadagnare importanza solo di recente rispetto ai film, come indicato dalla loro distribuzione più concentrata nei decenni successivi al 2000. Ciò suggerisce una crescita relativamente nuova ma significativa delle serie TV come formato preferito per il consumo di contenuti.

Conclusione del Confronto

I grafici indicano che Netflix offre una vasta gamma di contenuti, con una predominanza di film drammatici e documentari, nonché una crescente attenzione alle serie TV, specialmente nelle produzioni più recenti. La piattaforma sembra privilegiare contenuti moderni, con una forte concentrazione di titoli rilasciati negli ultimi dieci anni, il che riflette le tendenze attuali nel consumo di media. Mentre i film costituiscono ancora la maggior parte del catalogo, le serie TV hanno rapidamente guadagnato terreno, indicando un cambiamento nelle preferenze degli utenti e nella strategia di contenuto di Netflix.

4. Apprendimento Supervisionato

L'apprendimento supervisionato è una delle principali tecniche del machine learning, dove un modello viene addestrato su un dataset etichettato. Ciò significa che ogni esempio di input nel dataset è associato a un output (etichetta) conosciuto. L'obiettivo dell'apprendimento supervisionato è costruire un modello che possa predire l'etichetta corretta per nuovi dati non visti, basandosi sulle informazioni apprese durante l'addestramento.

In un contesto come quello della classificazione dei titoli Netflix, l'apprendimento supervisionato permette di costruire un modello che può distinguere tra i titoli che saranno più o meno visti dagli utenti. Il modello, addestrato su dati storici etichettati, può quindi prevedere se un nuovo titolo sarà probabilmente tra i più visti o meno, migliorando la raccomandazione personalizzata e aumentando l'engagement.

Pipeline

In questa fase viene seguita una precisa sequenza di passi: per prima cosa viene caricato il file CSV dei dati preprocessati; successivamente viene effettuato un mapping sui rating dei titoli. Questo mapping è necessario per convertire i rating, che sono stringhe categoriali, in numeri che possono essere usati come feature numeriche nei modelli. In seguito, mostriamo la tabella con spiegazione dei rating e della propria assegnazione numerica.

Rating	Valore Numerico
G	1
PG	2
PG-13	3
R	4
NC-17	5
Unrated	NaN
13+	3
16+	4
18+	5
7+	2
ALL	1

Dove:

- "G" significa "General Audience" (per tutti), quindi è stato assegnato il valore più basso, 1, per indicare che è adatto a tutti i gruppi di età.
- **"PG" (2):** "Parental Guidance" suggerisce che alcuni contenuti potrebbero non essere adatti per bambini molto piccoli senza la supervisione di un adulto. Questo rating è

leggermente più restrittivo rispetto a "G", quindi è stato assegnato un valore superiore, 2.

- **"PG-13" (3):** Questo rating indica che il contenuto potrebbe non essere adatto per bambini sotto i 13 anni. Per riflettere questa restrizione più severa, è stato assegnato un valore di 3.
- **"R" (4):** Il rating "R" indica che il contenuto è per adulti e potrebbe essere inappropriato per i minori. Questo giustifica un ulteriore aumento nel valore, portandolo a 4.
- **"NC-17" (5):** "No Children 17 and Under Admitted" è il rating più restrittivo, indicato per contenuti destinati esclusivamente a un pubblico adulto. È stato quindi assegnato il valore massimo, 5.
- **"Unrated" (np.nan):** Quando un contenuto non ha una classificazione specifica, viene considerato "Unrated" e non viene assegnato un valore numerico.
- **"13+" (3):** Analogamente a "PG-13", questo rating indica che il contenuto è adatto a persone di età pari o superiore a 13 anni.
- **"16+" (4) e "18+" (5):** Questi rating riflettono restrizioni più severe simili a "R" e "NC-17".
- **"7+" (2) e "ALL" (1):** "7+" è simile a "PG", suggerendo che il contenuto è più adatto a bambini più grandi, mentre "ALL" è equiparato a "G", adatto a tutte le età.

Seguentemente, avviene la fase di caricamento e filtraggio degli embedding, che consiste nel verificare se il numero degli embedding risulta maggiore del numero di righe nel DataFrame.

Le caratteristiche derivanti dal DataFrame preprocessato (numeric_rating, release_year) vengono combinate con gli embedding per formare il dataset finale (**features**) utilizzato per l'addestramento.
Successivamente, viene applicata la tecnica **SMOTE** (Synthetic Minority Over-sampling Technique) per la gestione del dataset sbilanciato. Questa tecnica, derivante dal classico oversampling, genera esempi sintetici per la classe minoritaria, aumentando la sua rappresentazione nel dataset e prevenendo che i modelli siano sbilanciati verso la classe maggioritaria.

Dopodichè, una volta bilanciato il dataset, esso viene suddiviso in un set di addestramento (70%) e un set di test (30%).
Alla fine della pipeline, avviene l'addestramento e la valutazione dei modelli.

Criteri di Giudizio

Quando si valuta un modello, è fondamentale utilizzare una serie di criteri e metriche di giudizio che permettano di comprendere quanto bene il modello sta performando. Le principali metriche considerate includono l'accuratezza, la precisione, il richiamo (recall), l'F1-score e la ROC-AUC. Queste metriche vengono spesso interpretate utilizzando la matrice di confusione, uno strumento essenziale per analizzare la performance di un modello.

Scelta dei Metodi: Motivazioni e confronto

Nel progetto sono stati scelti vari modelli di classificazione supervisionata, ciascuno con caratteristiche uniche che lo rendono adatto a diverse tipologie di dati e problemi:

Decision Tree Classifier

Il modello *Decision Tree* garantisce una facile interpretabilità e trasparenza. Questi metodi possono essere visualizzati e interpretati facilmente, rendendoli utili per capire quali caratteristiche influenzano maggiormente le decisioni del modello. Inoltre, essi sono capaci di gestire sia feature numeriche che categoriche.
In linea del tutto teorica, il modello Decision Tree presenta alcuni punti deboli:

- Propenso all'*overfitting*, specialmente con alberi molto profondi.
- Sensibile a piccole variazioni nei dati.

Parametri applicati:

```
'DecisionTree': DecisionTreeClassifier(max_depth=10, min_samples_split=10, min_samples_leaf=5),
```

- **max_depth = 10**: Limita la profondità massima dell'albero per prevenire l'overfitting. Una profondità eccessiva può portare il modello a imparare troppo dai dati di addestramento, perdendo la capacità di generalizzare.
- **min_samples_split = 10**: Numero minimo di campioni richiesti per dividere un nodo. Questo parametro aiuta a prevenire la creazione di nodi che dividono dati con un numero molto basso di campioni.

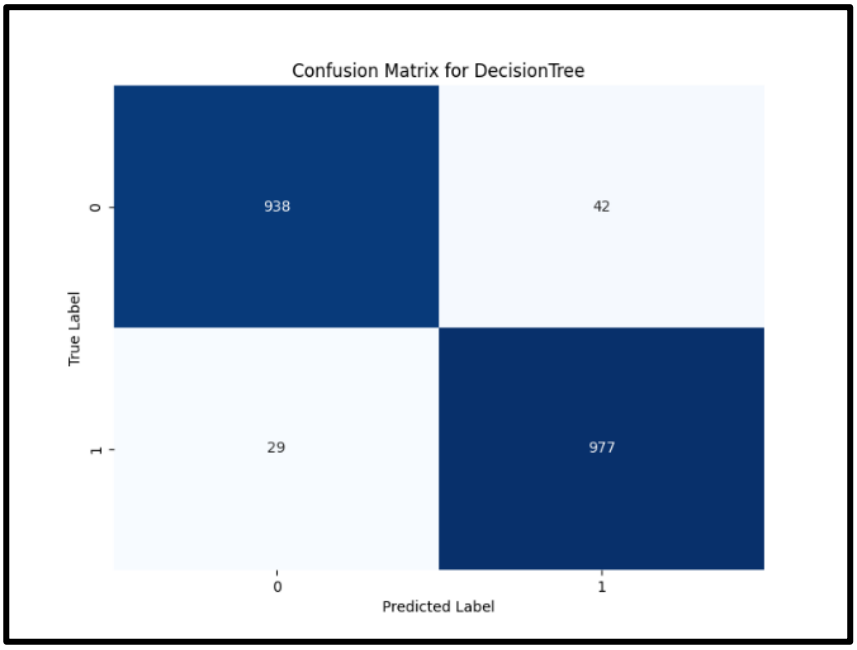
- **min_samples_leaf = 5**: Numero minimo di campioni richiesti in un nodo foglia. Valori bassi riducono il rischio di overfitting.

Risultati

	precision	recall	f1-score
False	0.9700103412616339	0.9571428571428572	0.9635336414997432
True	0.9587831207065751	0.9711729622266402	0.9649382716049383
accuracy	0.9642497482376636	0.9642497482376636	0.9642497482376636
macro avg	0.9643967309841045	0.9641579096847487	0.9642359565523407
weighted avg	0.9643232396108841	0.9642497482376636	0.9642451510092227

Nel nostro caso di studio, i risultati del Decision Tree mostrano valori attorno al 96%, dimostrando di aver fornito ottime prestazioni. L'incremento di accuracy suggerisce che la struttura ad albero cattura efficacemente le relazioni nel dataset. La coerenza tra precision e recall indica un buon bilanciamento tra falsi positivi e falsi negativi.

La matrice di confusione mostra valori bassi di FN e FP (rispettivamente 30 e 43), suggerendo una ottima precisione nella classificazione dei titoli più visti e indicando una capacità comparabile di identificare correttamente i titoli più popolari.



Per quanto concerne la curva di apprendimento, è possibile vedere un divario ampio tra errore di training e testing. Questo può suggerire la presenza di overfitting.

Random Forest Classifier

Il modello *Random Forest* garantisce una riduzione del rischio di overfitting rispetto ai singoli alberi decisionali, grazie all'aggregazione di molti alberi deboli (bagging) e una elevata accuratezza, specialmente su dataset complessi e con molti feature. Tuttavia, essi richiedono una maggiore complessità a livello computazionale rispetto ai singoli alberi decisionali e risultano meno interpretabile rispetto a un singolo albero decisionale.

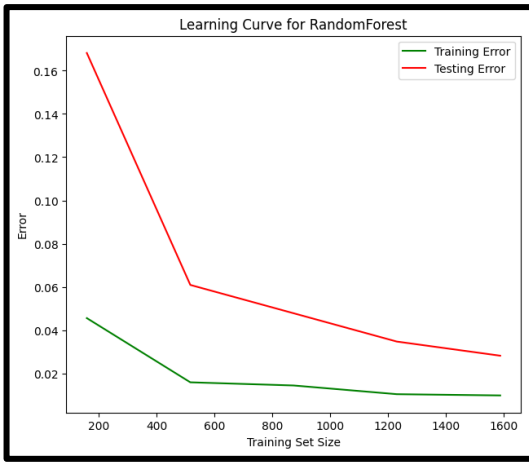
Parametri applicati:

```
'RandomForest': RandomForestClassifier(n_estimators=20, max_depth=10, min_samples_split=10, min_samples_leaf=5),
```

- **n_estimators = 20**: Numero di alberi nella foresta. Un numero maggiore di alberi tende a migliorare la stabilità e l'accuratezza del modello, ma aumenta anche il tempo di calcolo.
- **max_depth = 10**: Limita la profondità massima degli alberi, come nel Decision Tree.
- **min_samples_split = 10 e min_samples_leaf = 5**: Gli stessi parametri utilizzati nel Decision Tree sono applicati qui, estendendo i benefici a una foresta di alberi.

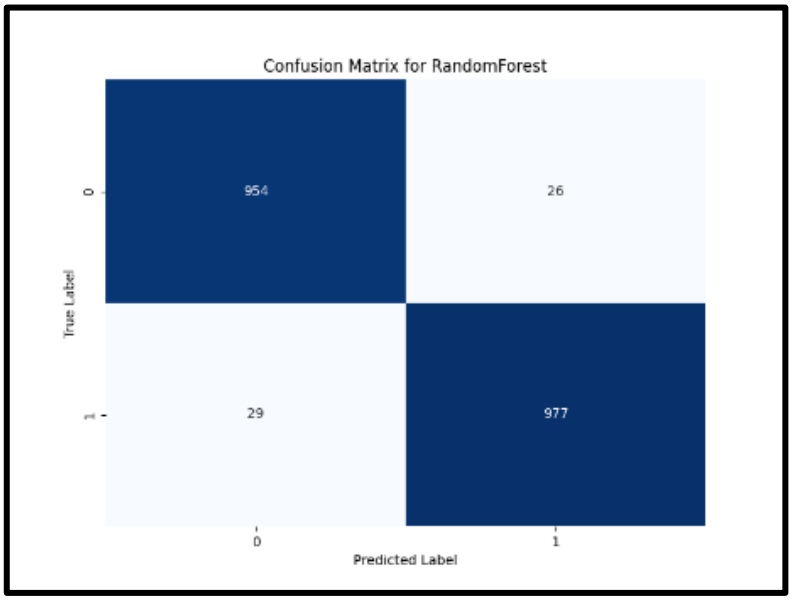
Risultati

I risultati del Random Forest mostrano valori attorno al 97%. In particolare, l'elevata percentuale dell'accuratezza suggerisce che il modello è molto efficace nel classificare correttamente i titoli Netflix.



	precision	recall	f1-score
False	0.970498474059003	0.9734693877551021	0.9719816607233825
True	0.9740777666999003	0.9711729622266402	0.9726231956197113
accuracy	0.972306143001007	0.972306143001007	0.972306143001007
macro avg	0.9722881203794517	0.9723211749908711	0.9723024281715469
weighted avg	0.9723115497874736	0.972306143001007	0.9723066275439801

La matrice di confusione mostra prestazioni eccellenti, con un alto numero di veri positivi (977) e veri negativi (949). Il numero di falsi positivi (31) e falsi negativi (29) è molto basso, indicando una classificazione molto accurata per entrambe le classi.



La curva di apprendimento mostra un gap di errore tra training e test abbastanza ridotto. Questo indica un buon equilibrio tra bias e varianza, con il modello che generalizza bene sui dati non visti. Nel complesso il Random Forest ha mostrato eccellenti risultati.

AdaBoost Classifier

Il modello *AdaBoost Forest* migliora la performance di modelli deboli, come i piccoli alberi decisionali, combinandoli in una serie di modelli iterativi ed è un modello robusto contro l'overfitting in molti casi. Un problema dell'AdaBoost Classifier consiste nella sua sensibilità ai dati rumorosi e agli outlier, poiché concentra l'attenzione sui casi difficili.

Parametri applicati:

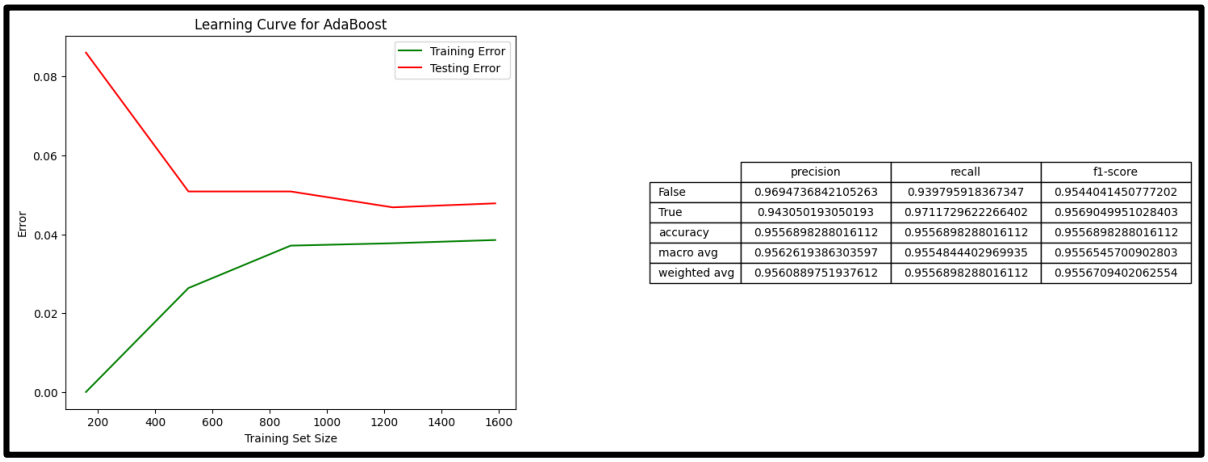
```
'AdaBoost': AdaBoostClassifier(algorithm='SAMME', n_estimators=50, learning_rate=1),
```

- **algorithm = 'SAMME'**: Specifica l'algoritmo da utilizzare. SAMME (Stagewise Additive Modeling using a Multiclass Exponential loss function) è una variante di AdaBoost per classificazione multiclass. È adatto per migliorare la performance di modelli deboli.

- **n_estimators = 50**: Numero di stime (modelli deboli) da combinare. Un numero maggiore può migliorare le prestazioni, ma potrebbe aumentare il rischio di overfitting.
- **learning_rate = 1**: Pondera il contributo di ciascun modello debole. Un valore più basso riduce l'importanza dei modelli deboli, mentre un valore più alto aumenta l'apprendimento, ma potrebbe anche rendere il modello più instabile.

Risultati

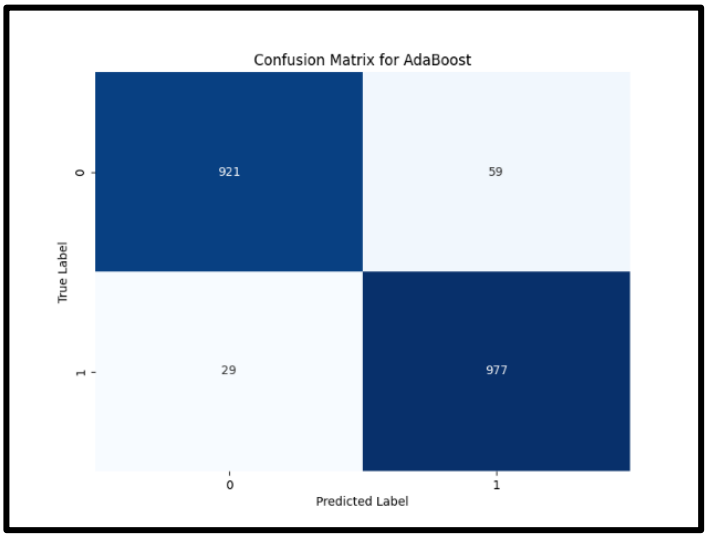
I risultati di AdaBoost si aggirano sul 95,56%. L'accuracy del 95.57% indica che il modello classifica correttamente la grande maggioranza dei titoli; la precision elevata suggerisce che quando il modello predice un titolo come "più visto", è generalmente affidabile. Il recall alto



	precision	recall	f1-score
False	0.9694736842105263	0.939795918367347	0.9544041450777202
True	0.943050193050193	0.9711729622266402	0.9569049951028403
accuracy	0.9556898288016112	0.9556898288016112	0.9556898288016112
macro avg	0.9562619386303597	0.9554844402969935	0.9556545700902803
weighted avg	0.9560889751937612	0.9556898288016112	0.9556709402062554

indica che il modello identifica correttamente una grande porzione dei titoli effettivamente più visti e L'F1-score bilanciato conferma una buona performance complessiva.

La matrice di confusione mostra una forte performance, con un alto numero di veri positivi e veri negativi e valori bassi di FN e FP (rispettivamente 29 e 59). Il numero di falsi positivi è poco superiore a quelli del Decision Tree, ma mostrano una buona precisione nella classificazione dei titoli più visti. I falsi negativi sono abbastanza bassi, indicando una capacità comparabile di identificare correttamente i titoli più popolari.



La curva di apprendimento mostra una rapida convergenza con un gap minimo tra errore di training ed errore di testing. Questo indica un buon equilibrio tra bias e varianza, suggerendo robustezza e generalizzazione.

K-Nearest Neighbors (KNN)

Il modello *K-Nearest Neighbors* è noto per la sua semplicità e non parametrica, che non assume una forma particolare per la funzione che genera i dati. Inoltre, esso risulta particolarmente adatto a problemi dove la relazione tra le feature e la classe non è lineare.

Parametri applicati:

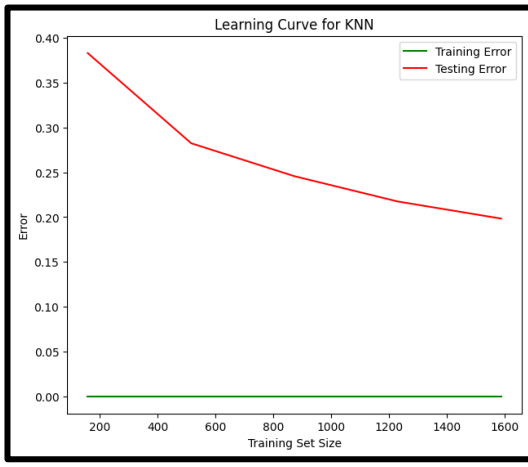
```
'KNN': KNeighborsClassifier(n_neighbors=5, weights='distance'),
```

- **n_neighbors = 5**: Numero di vicini più prossimi da considerare per la classificazione. Questo parametro bilancia la complessità del modello; valori troppo bassi possono portare a un modello troppo sensibile, mentre valori troppo alti potrebbero rendere il modello meno reattivo.
- **weights = 'distance'**: Pondera i vicini in base alla loro distanza. I vicini più vicini hanno più peso, rendendo il modello più preciso.

Risultati

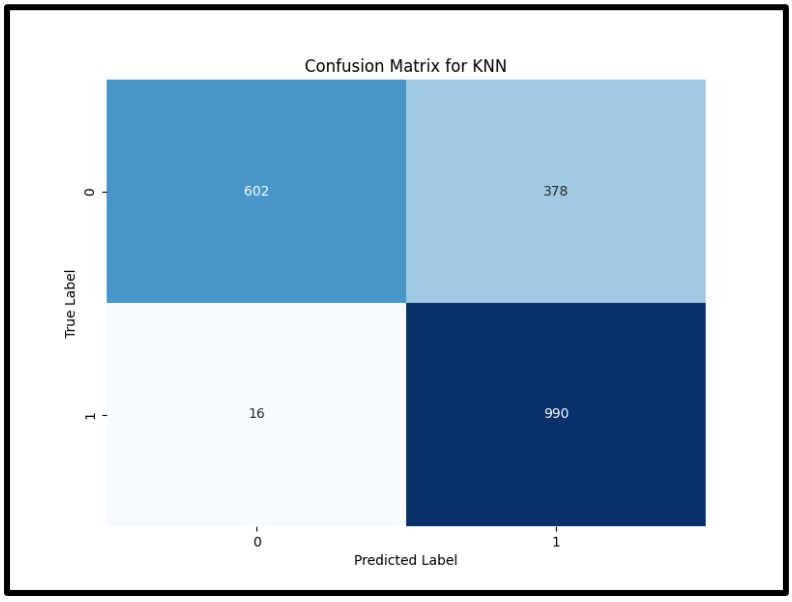
Nel nostro caso di studio, i risultati del KNN si aggirano attorno tra il 79% e 84,7%. La precision per la classe negativa (0) è alta (0.9741), ma il recall è relativamente bassa (0.6143), indicando che il modello tende a classificare erroneamente molti esempi negativi come positivi.

L'accuracy complessiva dell'80.16% suggerisce che il modello ha una performance discreta, ma c'è spazio per miglioramenti.



	precision	recall	f1-score
False	0.9741100323624595	0.6142857142857143	0.753441802252816
True	0.7236842105263158	0.9840954274353877	0.8340353833192923
accuracy	0.8016112789526687	0.8016112789526687	0.8016112789526687
macro avg	0.8488971214443877	0.799190570860551	0.7937385927860541
weighted avg	0.8472578789046747	0.8016112789526687	0.7942661439209303

Rispetto alla matrice di confusione, il modello KNN mostra una buona capacità di identificare i veri positivi (990), ma ha un numero significativo di falsi positivi (378).



Per quanto concerne la curva di apprendimento, invece, è possibile osservare un divario significativo tra l'errore di training (quasi zero) e l'errore di testing. Questo indica un forte overfitting: il modello si adatta troppo bene ai dati di training, ma non generalizza altrettanto bene sui dati di test e l'errore di testing diminuisce all'aumentare dei dati di training, ma resta comunque elevato.

XGBoost Classifier

Il modello *XGBoost Classifier* risulta essere estremamente efficiente, sia in termini di tempo che di risorse computazionali. Inoltre, questo modello è capace di gestire feature sparse e dati con strutture complesse ed è considerato uno dei modelli di boosting più potenti, spesso vincente in competizioni di machine learning.

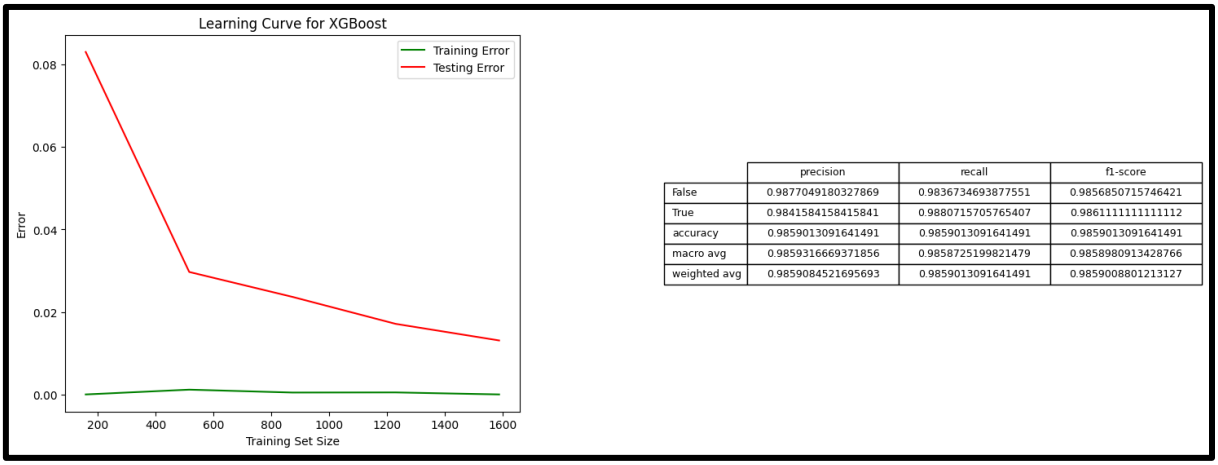
Parametri applicati:

```
'XGBoost': XGBClassifier(eval_metric='logloss', verbosity=0)
```

- **eval_metric = 'logloss'**: Metriche di valutazione per il modello. logloss è una misura comune utilizzata per valutare la probabilità delle previsioni. Una minore log-loss indica una migliore capacità predittiva del modello.
- **verbosity = 0**: Controlla il livello di output dei messaggi di debug. 0 significa silenzioso, riducendo il rumore durante l'addestramento.
- **Grid Search**: Per XGBoost, è stata effettuata una ricerca degli iperparametri (n_estimators, learning_rate, max_depth) per trovare la combinazione ottimale:
  - **n_estimators = [50, 100, 200]**: Numero di alberi da addestrare. Aumentare il numero di alberi tende a migliorare la performance, fino a un certo punto.
  - **learning_rate = [0.01, 0.1, 0.2]**: Tasso di apprendimento, che bilancia la velocità e l'accuratezza dell'apprendimento.
  - **max_depth = [3, 5, 7]**: Profondità massima degli alberi, con un impatto diretto sulla complessità del modello e il rischio di overfitting.

Risultati

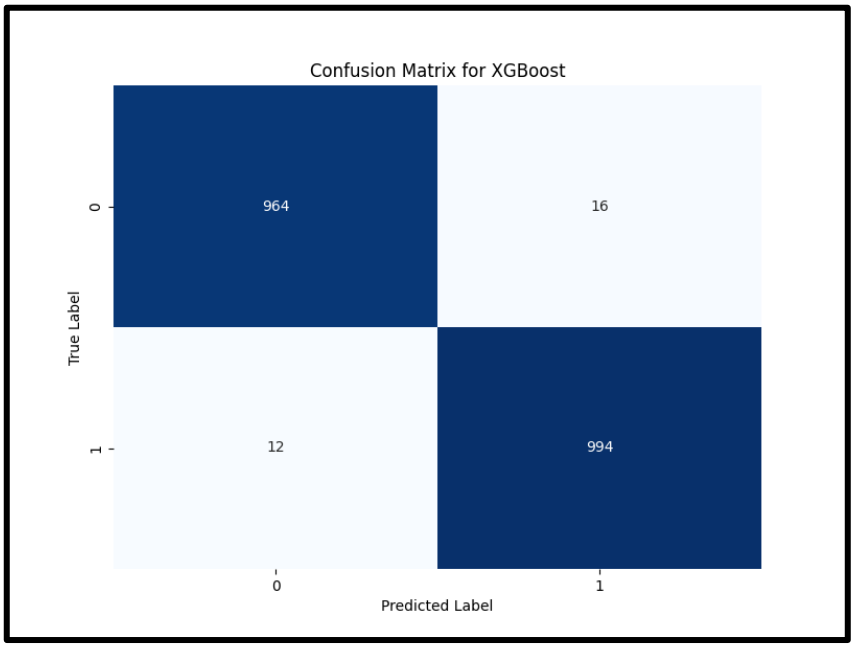
Nel nostro caso di studio, i risultati del XGBoost si aggirano attorno al 98,6%. La capacità del modello di classificare correttamente sia i titoli più visti che quelli meno visti è eccelso e XGBoost mostra le migliori prestazioni tra i modelli, con il minor numero di falsi positivi (16) e falsi negativi (12).



	precision	recall	f1-score
False	0.9877049180327869	0.9836734693877551	0.9856850715746421
True	0.9841584158415841	0.9880715705765407	0.9861111111111112
accuracy	0.9859013091641491	0.9859013091641491	0.9859013091641491
macro avg	0.9859316669371856	0.9858725199821479	0.9858980913428766
weighted avg	0.9859084521695693	0.9859013091641491	0.9859008801213127

L'errore di addestramento diminuisce significativamente all'aumentare delle dimensioni del training set, indicando che il modello sta apprendendo efficacemente dai dati. La curva tende ad appiattirsi verso la fine, suggerendo che il modello sta saturando e non sta più migliorando significativamente.

L'errore di test diminuisce inizialmente e poi si stabilizza a un valore relativamente basso. Questo comportamento indica che il modello sta generalizzando bene sui dati non visti durante l'addestramento e che il rischio di overfitting è contenuto.



5. Cross Validation

Il cross-validation è una tecnica fondamentale per la valutazione delle performance di un modello di machine learning. Consiste nel suddividere il dataset disponibile in un certo numero di "fold" (sottogruppi), allenando il modello su alcuni di questi fold e testandolo sui rimanenti. Questo processo viene ripetuto più volte, cambiando i fold di training e testing, per garantire che ogni dato venga utilizzato sia per l'allenamento che per il testing. Una delle versioni più comuni è la **Stratified K-Fold Cross-Validation**, che mantiene la distribuzione delle classi nel dataset per ogni fold, riducendo il rischio di bias.

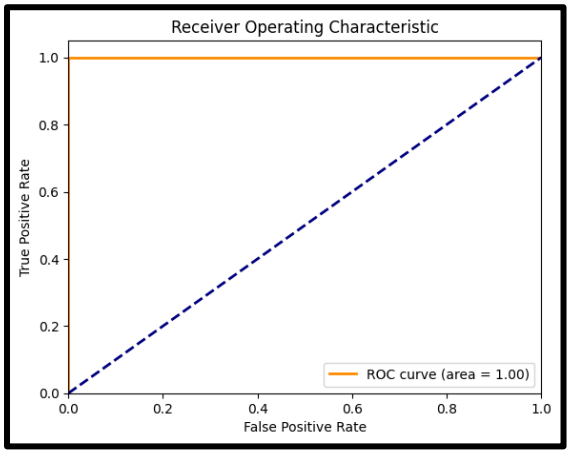
Perché Implementare il Cross-Validation?

Il cross-validation è cruciale per diversi motivi:

1. **Stima delle Performance del Modello:** Permette di avere una stima più robusta delle performance del modello su dati non visti, evitando di basarsi solo su una singola suddivisione tra training e test set.
2. **Riduzione del Variance nelle Stime:** L'utilizzo di più fold riduce la varianza nelle stime delle performance, fornendo una visione più completa di come il modello si comporterà su dati futuri.
3. **Ottimizzazione dei Parametri:** Consente di ottimizzare i parametri del modello in modo più accurato, grazie a una valutazione continua su diverse configurazioni di dati.

Descrizione dei Grafici

ROC Curve



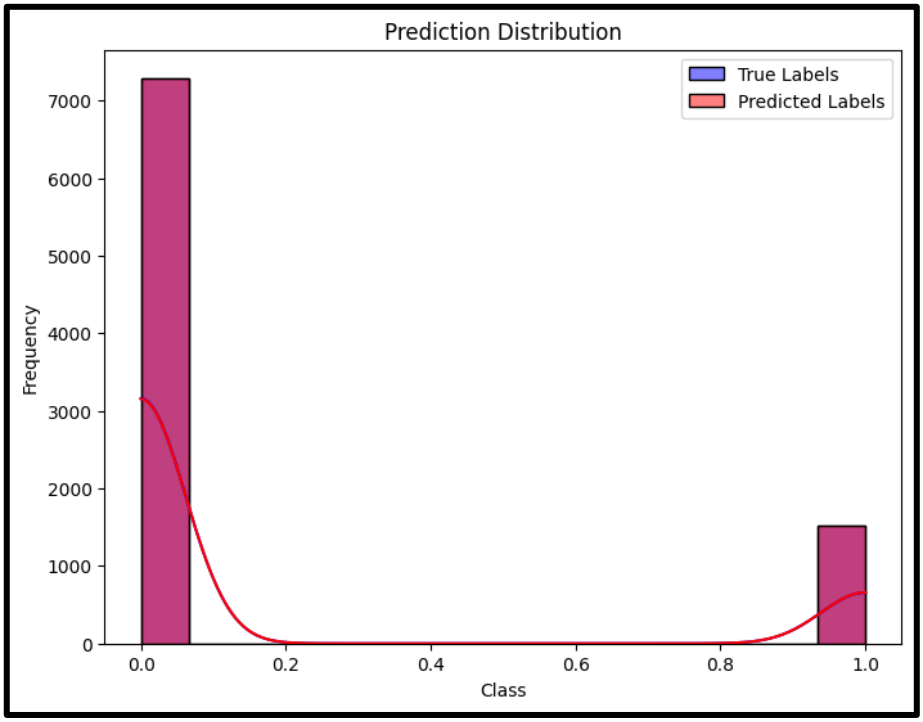
Dati Presentati: La curva ROC mostra il rapporto tra il tasso di veri positivi (True Positive Rate - TPR) e il tasso di falsi positivi (False Positive Rate - FPR) su vari livelli di soglia.

Risultati: La curva ROC ha un'area sotto la curva (AUC) pari a 1.00, il che indica una capacità perfetta del modello di distinguere tra classi.

Questo significa che il modello riesce a classificare correttamente tutti gli esempi come positivi o negativi senza errore.

Osservazioni: Un AUC di 1.00 è raramente raggiunto nei modelli pratici e suggerisce che il modello potrebbe essere sovra-addestrato o che i dati su cui è stato valutato sono relativamente facili da classificare. Questo livello di performance è indicativo di un modello molto robusto, ma potrebbe anche essere il risultato di un dataset con caratteristiche molto distinte tra le classi.

Distribuzione delle Predizioni



Dati Presentati: Questo grafico mostra la distribuzione delle etichette vere (True Labels) confrontata con la distribuzione delle predizioni del modello (Predicted Labels).

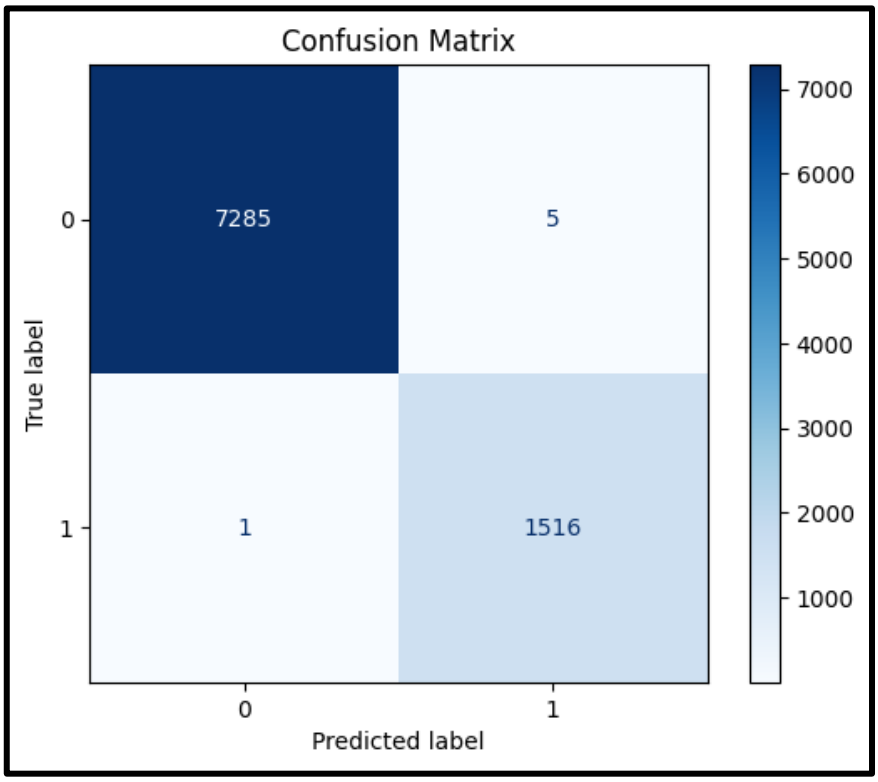
Risultati:

La distribuzione delle predizioni (etichettata in rosso) si sovrappone quasi completamente alla distribuzione delle etichette reali (etichettata in blu), indicando che il modello prevede con elevata precisione le classi del dataset.

Osservazioni: Una sovrapposizione quasi perfetta delle distribuzioni indica che il modello non solo fa previsioni accurate, ma anche che la probabilità predetta riflette correttamente la

distribuzione reale delle classi. Questo tipo di allineamento è raro e rappresenta un'indicazione forte che il modello non è solo accurato, ma anche ben calibrato.

Matrice di Confusione



Dati Presentati: La matrice di confusione rappresenta i numeri esatti delle predizioni corrette ed errate suddivisi per ciascuna classe.

Risultati:

- **Vero Negativi (TN):** 7285
- **Falsi Positivi (FP):** 5
- **Falsi Negativi (FN):** 1
- **Vero Positivi (TP):** 1516

Osservazioni: Il numero estremamente basso di falsi positivi e falsi negativi indica un'elevata precisione del modello. Questo, combinato con i numeri elevati di veri negativi e veri positivi, suggerisce che il modello ha una precisione e un richiamo molto alti, soprattutto considerando il bilanciamento delle classi.

Confronto tra i grafici

Tutti i grafici indicano che il modello è estremamente accurato. L'area sotto la curva ROC di 1.00, la matrice di confusione quasi perfetta, e la perfetta sovrapposizione tra distribuzione delle etichette vere e predette suggeriscono che il modello potrebbe essere considerato ideale per la classificazione in questione.

6. Confronto tra i modelli

Alla luce dei risultati ottenuti, stiliamo dei confronti tra i modelli presentati sulla base di Accuratezza, Generalizzazione, Bilanciamento delle classi e Complessità computazionale.

Accuratezza dei Modelli:

- **XGBoost:** Mostra la migliore accuratezza complessiva tra i modelli testati. Questo suggerisce che XGBoost ha una forte capacità di classificazione sui dati di addestramento.
- **Random Forest**: Segue XGBoost con prestazioni leggermente inferiori ma ancora molto elevate. Random Forest è noto per la sua robustezza e la capacità di gestire dati complessi.
- **AdaBoost**: Sebbene le prestazioni siano elevate, le curve di apprendimento indicano che potrebbe generalizzare meglio su dati non visti rispetto ad alcuni altri modelli.
- **Decision Tree**: Offre un vantaggio leggero rispetto a AdaBoost in termini di accuratezza, ma non raggiunge le prestazioni di XGBoost e Random Forest.
- **KNN**: Risulta significativamente meno accurato rispetto agli altri modelli. La performance può variare fortemente con il parametro kkk e tende a deteriorarsi con dati ad alta dimensionalità o con rumore.
- La **Cross-validation** è stata utilizzata per valutare le prestazioni dei modelli in modo più robusto, fornendo una stima più affidabile della loro accuratezza su nuovi dati non visti.

I risultati della cross-validation mostrano che **XGBoost** e **Random Forest** hanno prestazioni consistenti attraverso i vari fold, suggerendo una buona generalizzazione. Questo è indicativo di una bassa varianza nei modelli, ossia, i modelli sono stabili e non dipendono fortemente dal set di dati specifico. Invece, il **KNN** ha mostrato una varianza più alta tra i fold, indicando una performance meno consistente e quindi una maggiore sensibilità ai dati di addestramento, il che può portare a risultati non affidabili su nuovi dati.

Generalizzazione:

- XGBoost e Random Forest: Entrambi i modelli mostrano una buona capacità di generalizzazione, come indicato dalle curve di apprendimento che mostrano un basso rischio di overfitting. Questo li rende adatti per applicazioni in cui la capacità di generalizzare su dati nuovi è cruciale.
- AdaBoost: Ha indicato una buona capacità di generalizzazione, suggerendo che potrebbe essere più robusto ai dati non visti rispetto a modelli come Decision Tree.
- KNN: Mostra segni di forte overfitting, il che suggerisce che potrebbe non performare bene su nuovi dati non visti. Questo è dovuto alla sua natura non parametrica e alla sensibilità ai dati di addestramento.
- La cross-validation è essenziale per valutare la capacità di generalizzazione dei modelli. Durante il processo, ogni modello è stato testato su diversi fold, ognuno dei quali contiene una diversa parte del dataset, simulando così come il modello si comporterebbe su dati nuovi. XGBoost e Random Forest hanno mostrato prestazioni coerenti su tutti i fold, indicando una buona capacità di generalizzazione. Questo suggerisce che sono modelli affidabili per applicazioni in cui la generalizzazione è cruciale.

Bilanciamento delle Classi:

- XGBoost: Gestisce il bilanciamento delle classi in modo eccellente, risultando il miglior modello per questo aspetto. È particolarmente utile quando le classi sono sbilanciate.

Gli Altri Modelli: Tutti gli altri modelli sembrano gestire il bilanciamento delle classi bene, ma XGBoost eccelle grazie alla sua capacità di trattare le classi minoritarie in modo efficace.
La cross-validation aiuta anche a verificare come i modelli gestiscono il bilanciamento delle classi in diverse sezioni del dataset. XGBoost, grazie alla sua robusta implementazione di tecniche di boosting, mantiene alte performance anche su fold con distribuzioni di classe sbilanciate, mostrando una bassa sensibilità ai problemi di bilanciamento delle classi. I risultati di cross-validation hanno confermato che XGBoost è meno soggetto a variazioni nelle prestazioni legate allo sbilanciamento delle classi rispetto agli altri modelli.

Complessità Computazionale:

- KNN: È il più semplice e veloce da addestrare, ma la sua complessità computazionale aumenta notevolmente con la dimensione del dataset durante la fase di previsione.
- Random Forest: Richiede più risorse rispetto a KNN ma è meno complesso rispetto a XGBoost. Il suo addestramento può essere relativamente lungo, ma gestisce bene grandi quantità di dati.
- XGBoost: Tipicamente il più complesso e computazionalmente intensivo. Tuttavia, la sua capacità di ottimizzare le prestazioni tramite parametri e tecniche di boosting giustifica questo costo.

  Durante la cross-validation, la complessità computazionale diventa un fattore importante poiché ogni modello viene addestrato e valutato più volte.

  XGBoost richiede più tempo per eseguire la cross-validation a causa della sua complessità, ma la sua capacità di ottimizzare le prestazioni attraverso la ricerca di iperparametri lo rende vantaggioso rispetto ai modelli meno complessi. La cross-validation ha dimostrato che nonostante la complessità computazionale, XGBoost offre un compromesso eccellente tra tempo di esecuzione e accuratezza.

Considerazioni Finali:

La cross-validation è stata fondamentale per valutare le prestazioni di ogni modello in modo robusto e affidabile. I risultati ottenuti da questa tecnica di valutazione hanno confermato che XGBoost è il miglior modello complessivo, eccellendo in accuratezza, capacità di generalizzazione e gestione del bilanciamento delle classi. Tuttavia, è anche il più complesso in termini di risorse computazionali.

Random Forest si è dimostrato un ottimo secondo, offrendo un buon equilibrio tra accuratezza, generalizzazione e complessità computazionale, rendendolo una scelta solida in scenari con risorse limitate.

KNN, nonostante la sua semplicità e rapidità di addestramento, ha mostrato prestazioni inferiori, specialmente in scenari con dati complessi o sbilanciati, suggerendo che potrebbe non essere la scelta migliore per questo tipo di problemi.

In sintesi, la cross-validation ha confermato e rafforzato le osservazioni fatte durante l'analisi dei modelli, fornendo un quadro chiaro delle performance dei modelli in un contesto applicativo reale.

Knowledge Base (KB)

1. Panoramica e Motivazioni

In questo progetto, la Knowledge Base (KB) implementata in Prolog rappresenta un componente fondamentale per migliorare il sistema di classificazione e raccomandazione dei titoli Netflix. L'integrazione di una KB è motivata dalla necessità di sfruttare il ragionamento simbolico accanto alle tecniche di machine learning. Mentre i modelli di machine learning offrono potenti metodi per la classificazione e la raccomandazione, spesso operano come "scatole nere," rendendo opaca la logica decisionale. La natura dichiarativa di Prolog consente la rappresentazione esplicita della conoscenza, permettendo un ragionamento basato su fatti e regole chiaramente definiti.

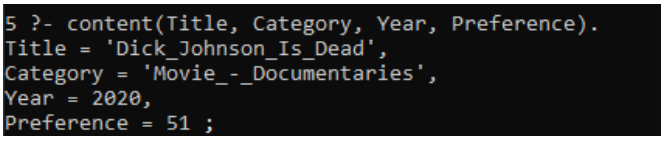
L'integrazione di una KB consente di colmare il divario tra approcci basati sui dati e approcci basati su regole. Questo metodo ibrido potenzia la capacità del sistema di fornire raccomandazioni non solo basate su pattern appresi dai dati, ma anche su inferenze logiche derivate dalla conoscenza esplicita codificata nella KB. Questo approccio è particolarmente utile per generare raccomandazioni che rispettano criteri complessi e predefiniti, come le preferenze di genere o attributi specifici dei contenuti.

2. Struttura e Contenuto della KB

La KB è composta da due file principali:

- File `facts.pl`

  Questo file contiene fatti in Prolog. I fatti rappresentano dati statici su cui si possono eseguire query. Nel contesto di un sistema di raccomandazione, i fatti potrebbero rappresentare contenuti disponibili, con dettagli come il titolo, la categoria, l'anno di rilascio e una misura della preferenza dell'utente.

  Questo fatto indica che esiste un contenuto con titolo (esempio "Dick_Johnson_Is_dead"), appartenente ad una certa categoria (esempio "Movie_-_Documentaries"), rilasciato in un determinato anno (esempio “2020”) e possiede una certa preferenza (esempio “51”).

- File `rules.pl`

Il file delle regole (`rules.pl`) contiene le regole di inferenza che Prolog utilizza per dedurre nuove informazioni dai fatti. Le regole sono dichiarazioni che stabiliscono condizioni sotto le quali certe conclusioni sono vere.

Ad esempio, una regola per raccomandare contenuti con alta preferenza può essere:

```
recommend(Content) :-
    content(Content, _, _, Preference),
    Preference >= 80.
```

Questa regola dice che un contenuto è raccomandato se la sua preferenza è maggiore o uguale a 80 (concetto che può essere esteso anche per preferenze minori di 80).

Altre regole potrebbero includere:

- similar_content: Per trovare contenuti simili in base alla categoria.
- recommend_similar: Per raccomandare contenuti simili con alta preferenza.
- recommend_by_genre: Per raccomandare contenuti basati su un genere specifico.
- recommend_by_year: Per raccomandare contenuti basati sull'anno di rilascio.
- recommend_by_genre_year: Per raccomandare contenuti basati su una combinazione di genere e anno.

Esempio console prolog:

```
9 ?- content(Title, Category, Year, Preference), Preference > 80.
Title = 'Blood_&_Water',
Category = 'TV_Show_-_International_TV_Shows,_TV_Dramas,_TV_Mysteries',
Year = 2021,
Preference = 94 ;
Title = 'Midnight_Mass',
Category = 'TV_Show_-_TV_Dramas,_TV_Horror,_TV_Mysteries',
Year = 2021,
Preference = 81 ;
Title = 'The_Starling',
Category = 'Movie_-_Comedies,_Dramas',
Year = 2021,
Preference = 84 ;
Title = 'Vendetta:_Truth,_Lies_and_The_Mafia',
Category = 'TV_Show_-_Crime_TV_Shows,_Docuseries,_International_TV_Shows',
Year = 2021,
Preference = 87 .
```

Contenuto con preferenza maggiore di 80

```
8 ?- content(Title, Category, Year, Preference), Preference < 80.
Title = 'Dick_Johnson_Is_Dead',
Category = 'Movie_-_Documentaries',
Year = 2020,
Preference = 51 ;
Title = 'Ganglands',
Category = 'TV_Show_-_Crime_TV_Shows,_International_TV_Shows,_TV_Action_&_Adventure',
Year = 2021,
Preference = 72 ;
Title = 'Jailbirds_New_Orleans',
Category = 'TV_Show_-_Docuseries,_Reality_TV',
Year = 2021,
Preference = 59 .
```

Contenuto con preferenza minore di 80

3. *Esempio di Regole*

Una regola in Prolog è una dichiarazione logica che descrive come un certo fatto può essere dedotto sulla base di altri fatti già noti. Si compone di una testa e di un corpo, separati da `:-`.

La testa rappresenta il fatto che si vuole dimostrare, mentre il corpo contiene le condizioni che devono essere soddisfatte affinché la regola sia vera. In sostanza, una regola definisce una relazione condizionale tra fatti, permettendo a Prolog di eseguire inferenze logiche e dedurre nuove informazioni dai dati esistenti. Ad esempio:

```
# Comandi Prolog Console

## Comandi di avvio e caricamento
?- cd('D:/code__icon_23-24/results/prolog').
?- [facts].  % Carica il file dei fatti
?- [rules].  % Carica il file delle regole

## Raccomandare Contenuti con Alta Preferenza UQuesta query restituisce tutti i contenuti con una preferenza pari o superiore a 80):
?- recommend(Content).


## Visualizzare Tutti i Fatti sui Contenuti:
?- content(Title, Category, Year, Preference).

## Trovare Contenuti con una Specifica Categoria/Titolo/ Anno ( non serve mettere '' )/Preferenza ( anche qui non bisogna mettere ' ' ):
?- content(Title, 'TV_Show_-_International_TV_Shows,_Romantic_TV_Shows,_TV_Comedies', Year, Preference).

## Trovare Contenuti con una Preferenza Superiore( o inferiore) a un Valore:
?- content(Title, Category, Year, Preference), Preference < 80.

?- content(Title, Category, Year, Preference), Preference > 80.

## Trovare Contenuti  appartententi alla stessa categoria :
?- similar_content('Glee', SimilarTitle). % riporta la lista dei titoli simili al titolo scelto.

se vogliamo verificare che due titoli sono simili, basa solo inserire due titoli.


## Raccomandare Contenuti Basati su Genere ( questa funzionalità permette all'utente di visualizzare i generi dei film o serie tv):
?- recommend_by_genre(Genre, Content).

## Raccomandare Contenuti Basati su Anno di Rilascio:
?- recommend_by_year(2021, Content).

## Raccomandare Contenuti Basati su Genere e Anno di Rilascio ( ricerca specifica):
?- recommend_by_genre_year('Movie_-_Comedies', 2010, Content).

inserendo nelle parentesi anche solo (Genre, 2010, Content), verrà mostrata la lista dei vari generi e contenuti appartenenti a quell'anno
```

4. *Processo di Implementazione*

La KB viene generata automaticamente da uno script Python (generate_prolog_files.py), che elabora i dati preprocessati e li converte in fatti e regole Prolog. Questa automazione garantisce che la KB sia aggiornata con i dati più recenti, consentendo raccomandazioni coerenti e accurate.

Il processo prevede i seguenti passaggi:

a. Raccolta e Preparazione dei Dati

Il primo passo nell'implementazione di una Knowledge Base (KB) in Prolog consiste nella raccolta e preparazione dei dati. Questi dati sono tipicamente organizzati in un formato strutturato, come un file CSV, contenente informazioni rilevanti sui contenuti, quali titolo, categoria, anno di rilascio e un valore che rappresenta la preferenza dell'utente. Una volta

raccolti, questi dati vengono caricati in un DataFrame utilizzando la libreria Pandas, il che facilita le successive operazioni di manipolazione e aggiornamento dei dati.

b. Generazione dei Fatti Prolog

il passo successivo è la generazione dei fatti Prolog. Ogni riga del DataFrame viene convertita in un fatto Prolog, che rappresenta un'asserzione semplice su un contenuto, includendo informazioni come il titolo, la categoria, l'anno di rilascio e la preferenza. Questi fatti vengono poi salvati in un file chiamato **facts.pl.** Questo file diventa una parte essenziale della KB, fornendo la base su cui si costruiscono le inferenze e le decisioni del sistema.

c. Definizione delle Regole Prolog

Oltre ai fatti, una Knowledge Base efficace deve includere regole che permettano al sistema di ragionare sui fatti esistenti. Le regole in Prolog descrivono le condizioni sotto le quali certe conclusioni possono essere tratte. Ad esempio, si possono definire regole per raccomandare contenuti con alte preferenze, per identificare contenuti simili basati sulla categoria, o per fare raccomandazioni basate su combinazioni di genere e anno di rilascio. Queste regole vengono scritte in un file separato, `rules.pl`, e rappresentano la logica decisionale del sistema.

d. Generazione e Aggiornamento dei File Prolog

L'ultima fase del processo prevede la generazione automatica dei file Prolog contenenti i fatti e le regole. Il sistema crea e aggiorna i file facts.pl e rules.pl, assicurando che la KB sia sempre allineata con i dati più recenti e con le logiche di raccomandazione definite. Questa automazione facilita la manutenzione della KB, permettendo al sistema di evolversi e adattarsi senza richiedere interventi manuali costanti

5. Risultati e Impatto

L'ibridazione di metodi basati sui dati e metodi basati su regole dimostra l'efficacia della combinazione di diverse tecniche di intelligenza artificiale per ottenere una maggiore accuratezza nei sistemi di raccomandazione. La KB non solo fornisce un quadro chiaro per la generazione di raccomandazioni, rendendo questo sistema più robusto e versatile, ma consente anche future estensioni in cui possono essere facilmente integrate ulteriori regole o logiche.

Nb:

Per eventuali problematiche con prolog è possibile consultare i file presenti nella directory " ../documenti" :

- `lista_categoria_prolog`: documento in .txt che contiene la lista di tutte le categorie presenti nel dataset da utilizzare sulla console prolog nella sezione 'Categoria'

- `comandi_console_prolog`: documento in .txt che contiene la lista delle query da porre a prolog

Ricerca e Raccomandazione

La seconda parte del progetto si concentra sulla ricerca e la raccomandazione di contenuti. Qui, l'obiettivo è fornire agli utenti suggerimenti personalizzati basati sui loro interessi e comportamenti, utilizzando le classificazioni e le informazioni derivate dalla prima parte del progetto.

I file principali in questa sezione includono:

- **`search_and_recommendation.py`**: Implementa funzionalità di ricerca e raccomandazione, utilizzando tecniche come il content-based filtering e la similarità dei coseni per suggerire contenuti simili a quelli già apprezzati dagli utenti.
- **`main.py`**: Coordina l'integrazione delle funzionalità di ricerca e raccomandazione con le altre componenti del sistema, gestendo il flusso di lavoro complessivo.

1. Ricerca e raccomandazione

Nella seconda fase di questo caso di studio, ci siamo concentrati sull'implementazione di un sistema di ricerca e raccomandazione per la piattaforma Netflix. La ricerca consente agli utenti di trovare contenuti specifici o categorie di interesse all'interno dell'ampia libreria di titoli, mentre la raccomandazione suggerisce contenuti rilevanti basati sulle preferenze e interazioni dell'utente.

Abbiamo scelto di sviluppare questo sistema di ricerca e raccomandazione perché migliora significativamente l'esperienza dell'utente, aiutandolo a scoprire nuovi contenuti in modo più efficiente e personalizzato. In un contesto in cui la quantità di contenuti disponibili è in costante crescita, la capacità di fornire raccomandazioni pertinenti e immediate diventa essenziale per mantenere l'utente coinvolto e soddisfatto.

Per realizzare un sistema in grado di fornire raccomandazioni ed effettuare ricerche da parte di un utente, abbiamo adottato il modello di *content-based filtering*.

Questo approccio utilizza gli embeddings per calcolare la somiglianza tra i vari contenuti nel database.

Il *content-based filtering* si distingue per la sua capacità di generare raccomandazioni analizzando direttamente le caratteristiche dei contenuti, come il genere, il cast, e altre informazioni contestuali.

L'algoritmo di *content-based filtering* viene applicato per creare un profilo utente basato sui contenuti selezionati o apprezzati dall'utente stesso. Questo profilo viene poi confrontato con tutti i contenuti disponibili nel database, utilizzando una metrica di somiglianza come la *cosine similarity*, per identificare e ordinare i contenuti più rilevanti da raccomandare.

In un dataset ricco di metadati come quello che abbiamo scelto, questo approccio ci consente di sfruttare appieno la varietà e la profondità delle informazioni disponibili.

Ad esempio, se un utente mostra una preferenza per il genere “Comico”, il sistema è in grado di identificare e raccomandare altri film che condividono queste caratteristiche.

Raccomandazioni personalizzate basate su preferenze espresse dall’utente

Il sistema è progettato per fornire raccomandazioni personalizzate basate sulle preferenze espresse dall'utente.

Inizia con la raccolta del feedback dell'utente su specifici contenuti (come valutare un titolo su una scala da 1 a 5), e utilizza queste informazioni per migliorare continuamente la qualità delle raccomandazioni.

Ad esempio: Quando l'utente esprime una valutazione positiva per un titolo, il sistema utilizza il *content-based filtering* per identificare titoli simili basandosi sulle caratteristiche intrinseche del contenuto, rappresentate attraverso embeddings.

Ad esempio, consideriamo un caso in cui l'utente valuta positivamente il titolo di una serie tv come "Glee" (5 su 5).

```
Come valuti il titolo 'Glee' per 'Intrattenimento' su una scala da 1 a 5? 5

Raccomandazioni basate su 'Glee':
- Refugee (Movie - Documentaries, Preferenze: 99)
- Brainchild (TV Show - Kids' TV, Preferenze: 63)
- Darra (Movie - Dramas, International Movies, Preferenze: 86)
- Backcountry (Movie - Horror Movies, Independent Movies, Thrillers, Preferenze: 48)
- Ishqiya (Movie - Comedies, International Movies, Romantic Movies, Preferenze: 83)
```

In base alla tipologia di valutazione, il sistema ha generato una serie di raccomandazioni basate sulla somiglianza tra l'embedding di "Glee" e quelli di altri titoli nel database.

Questo processo sfrutta la *cosine similarity* per misurare quanto ogni titolo sia simile a "Glee" e ordina i titoli raccomandati in base a questa somiglianza.

N.B:

Questo discorso è appicato anche con tutte le tipologie di votazioni che ruotano su scala da 3 a 5 su 5

Se consideriamo un secondo caso in cui l'utente esprime una valutazione negativa per "Glee" (2 su 5). Il sistema, riconoscendo che il titolo non è stato gradito, chiede all'utente di specificare un genere preferito. Una volta fornito (in questo caso "stand-up"), il sistema modifica il suo approccio e genera una nuova serie di raccomandazioni basate sul nuovo genere specificato dall'utente. Questo meccanismo di adattamento è implementato nel codice attraverso la funzione `user_rating_flow_with_decision`, che raccoglie le valutazioni e decide il prossimo passo, come suggerire un genere diverso se la valutazione è bassa.

```
Come valuti il titolo 'Glee' per 'Intrattenimento' su una scala da 1 a 5? 2
Sembra che questo titolo non ti sia piaciuto. Che genere preferisci? stand-up

Ecco alcune raccomandazioni nel genere 'Stand-Up Comedy':
- Donald Glover: Weirdo (Movie - Stand-Up Comedy, Preferenze: 100)
- Nikki Glaser: Bangin' (Movie - Stand-Up Comedy, Preferenze: 100)
- Todo lo que sería Lucas Lauriente (Movie - Stand-Up Comedy, Preferenze: 100)
- Felipe Esparza: Bad Decisions (TV Show - Stand-Up Comedy & Talk Shows, TV Comedies, Preferenze: 100)
- Bill Burr: I'm Sorry You Feel That Way (Movie - Stand-Up Comedy, Preferenze: 100)
- Jo Koy: Comin' In Hot (Movie - Stand-Up Comedy, Preferenze: 100)
- Jandino: Whatever it Takes (Movie - Stand-Up Comedy, Preferenze: 100)
- Rodney Carrington: Here Comes the Truth (Movie - Stand-Up Comedy, Preferenze: 100)
- Sam Kinison: Live in Vegas (Movie - Stand-Up Comedy, Preferenze: 100)
- Cristela Alonzo: Lower Classy (Movie - Stand-Up Comedy, Preferenze: 99)
```

Da come possiamo dedurre, il sistema è costruito per essere interattivo, permettendo all'utente di esplorare vari contenuti e categorie attraverso un processo iterativo di ricerca e raccomandazione. Ogni iterazione offre l'opportunità di affinare le raccomandazioni in base alle preferenze specificate in tempo reale. Questo è evidente nel flusso delle immagini, dove l'utente è guidato a esplorare nuovi generi se i contenuti precedentemente raccomandati non soddisfano le aspettative.

2. Visualizzazione dei titoli più popolari

Abbiamo introdotto una funzionalità aggiuntiva nel sistema di raccomandazione che non si limita solo a suggerire contenuti, ma permette anche di visualizzare in modo interattivo e dinamico i titoli più popolari attraverso grafici. Questa funzionalità è stata sviluppata per offrire agli utenti e agli analisti uno strumento visivo che rappresenta la popolarità dei contenuti in una forma facilmente interpretabile.

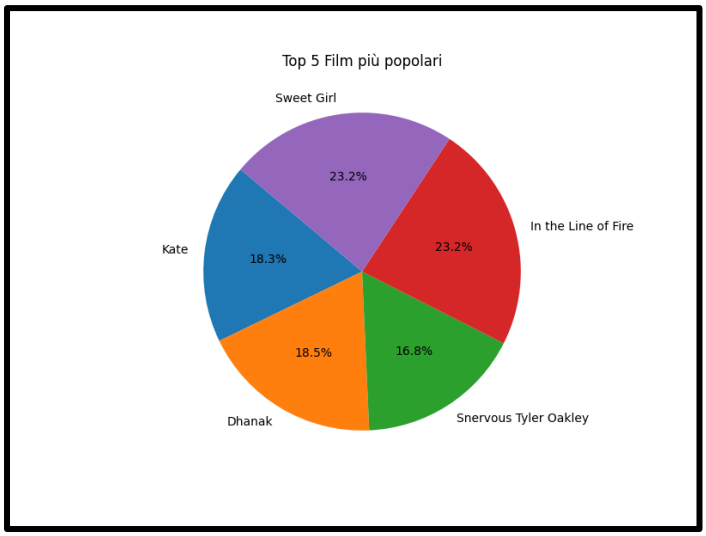
```
==============================================
 Menu di Selezione
----------------------------------------------
 1. Classificazione dei Film e Serie Tv
 2. Ricerca e Raccomandazioni
 3. Visualizza Titoli Più Popolari
 4. Generazione della KB
 5. Uscita
==============================================
 Seleziona un'opzione (1-5): 3

Top 5 Film con preferenze variabili:
82. Kate (Movie - Action & Adventure, Preferenze: 75)
117. Dhanak (Movie - Comedies, Dramas, Independent Movies, Preferenze: 76)
117. Dhanak (Movie - Comedies, Dramas, Independent Movies, Preferenze: 76)
174. Snervous Tyler Oakley (Movie - Documentaries, LGBTQ Movies, Preferenze: 69)
184. In the Line of Fire (Movie - Action & Adventure, Classic Movies, Preferenze: 95)
248. Sweet Girl (Movie - Action & Adventure, Dramas, Preferenze: 95)
Grafico salvato in: D:\code__icon_23-24\scripts\..\results\visualizations\statistic_recommander\top_movies_pie.png

Top 5 Serie TV con preferenze variabili:
3. Ganglands (TV Show - Crime TV Shows, International TV Shows, TV Action & Adventure, Preferenze: 73)
544. Undercover (TV Show - Crime TV Shows, International TV Shows, TV Action & Adventure, Preferenze: 60)
545. Workin' Moms (TV Show - International TV Shows, TV Comedies, Preferenze: 60)
640. Godzilla Singular Point (TV Show - Anime Series, International TV Shows, Preferenze: 68)
1015. PJ Masks (TV Show - Kids' TV, Preferenze: 90)
Grafico salvato in: D:\code__icon_23-24\scripts\..\results\visualizations\statistic_recommander\top_tv_shows_pie.png
```

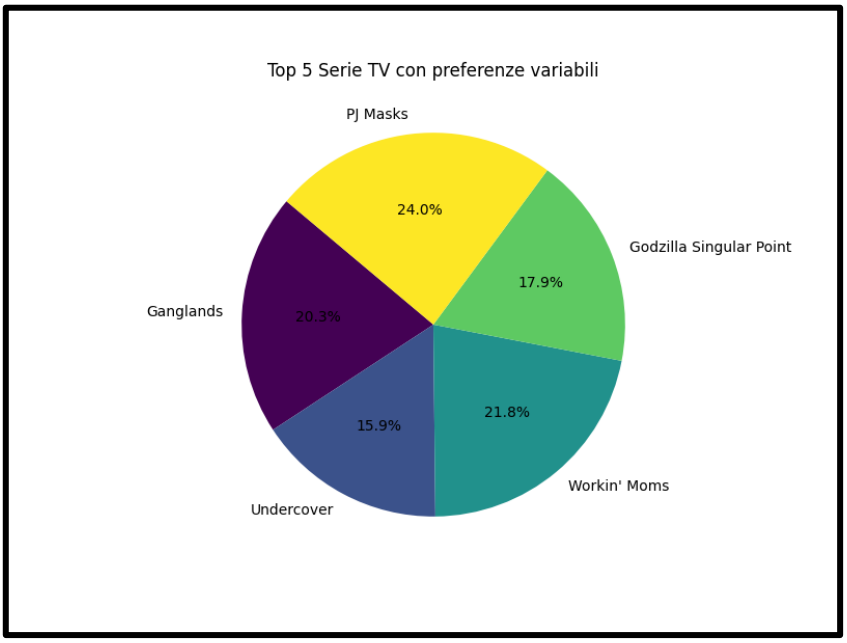
Abbiamo implementato un meccanismo per selezionare i titoli più popolari basandoci su un campo denominato "preferences", che riflette il gradimento generale degli utenti verso i diversi contenuti. Questa selezione non è statica, ma è progettata per essere aggiornata dinamicamente, in modo che i titoli visualizzati riflettano le tendenze più recenti o i comportamenti osservati degli utenti.

La logica di selezione è stata implementata in modo tale che si possa filtrare per categoria, permettendo all'utente di vedere i film o le serie TV più apprezzati in base a preferenze aggregate.

L'introduzione di questa funzionalità di visualizzazione aggiunge un livello di interattività e offre un modo più immediato per comprendere i dati relativi alla popolarità dei contenuti. Non solo arricchisce l'esperienza utente, ma fornisce anche agli analisti e ai decisori uno strumento potente per monitorare e interpretare le preferenze del pubblico nel tempo.

Inoltre, questa funzionalità può essere estesa in futuro per includere altre metriche diverse, come l'evoluzione della popolarità nel tempo o la comparazione tra diverse categorie di contenuti.





Librerie utilizzate

Sono state utilizzate le seguenti librerie:

- Pandas: libreria Python per la manipolazione e l'analisi dei dati. ("Python per Data Analyst") È particolarmente utile per lavorare con strutture di dati etichettate come DataFrame, che sono tabelle bidimensionali con etichette di riga e colonna. Consente operazioni complesse su dati strutturati, come filtraggio, raggruppamento, fusione, e aggregazione.
- NumPy: libreria fondamentale per il calcolo scientifico in Python. Fornisce supporto per array e matrici multidimensionali, oltre a una vasta gamma di funzioni matematiche per operare su questi array.
- spaCy: libreria open-source per l'elaborazione del linguaggio naturale (NLP) in Python, progettata per essere veloce ed efficiente.
- TensorFlow: libreria open-source sviluppata da Google per il machine learning e il deep learning. Supporta l'addestramento e l'esecuzione di reti neurali su grandi dataset. Fornisce strumenti per definire, addestrare e ottimizzare modelli di reti neurali.
- NLTK (Natural Language Toolkit): libreria Python per l'elaborazione del linguaggio naturale. ("NLTK: Libreria Python per l'Elaborazione del Linguaggio Naturale") È particolarmente utile per la ricerca e l'educazione nell'NLP.
- Scikit-learn: libreria Python per il machine learning, costruita su NumPy, SciPy e Matplotlib. Fornisce implementazioni di algoritmi comuni come regressione, classificazione, clustering, e riduzione della dimensionalità.
- Matplotlib: libreria Python per la creazione di grafici e visualizzazioni. ("Creare Grafici in Pandas | Codegrind") È molto flessibile e può generare una vasta gamma di grafici. Può creare grafici a linee, a barre, istogrammi, scatter plot, ecc.
- RDFlib: libreria Python per lavorare con RDF (Resource Description Framework), un modello di dati utilizzato per rappresentare informazioni sul web in formato semantico.

- PySWIP: interfaccia Python per SWI-Prolog, che consente di integrare la logica di programmazione in progetti Python. Permette di eseguire query Prolog e ricevere risposte direttamente in Python.
- Pgmpy: libreria Python per creare e manipolare modelli grafici probabilistici (PGM), come reti bayesiane e reti markoviane.
- SPARQLWrapper: libreria Python che facilita l'accesso a endpoint SPARQL, un linguaggio di query utilizzato per interrogare i grafi RDF.
- Openpyxl: libreria Python per leggere e scrivere file Excel (xlsx).
- WordCloud: libreria Python per generare word cloud, ovvero rappresentazioni visive di testo dove le parole più frequenti appaiono più grandi.
- Seaborn: libreria Python per la visualizzazione dei dati, costruita su Matplotlib. Fornisce interfacce di alto livello per creare grafici statistici. Offre strumenti per creare grafici avanzati come heatmap, distribuzioni, e relazioni tra variabili.
- Imbalanced-learn: libreria Python per trattare con dataset sbilanciati, tipici in molti problemi di classificazione. Include metodi come oversampling (ad esempio, SMOTE) e undersampling per riequilibrare classi minoritarie e maggioritarie. Si integra facilmente con Scikit-learn per applicare tecniche di bilanciamento durante il preprocessing dei dati.

Conclusioni

Con lo sviluppo di tale caso di studio, abbiamo dimostrato la capacità di saper creare un sistema adatto per l'analisi, la classificazione e la raccomandazione di contenuti multimediali su una piattaforma di streaming come Netflix.

Il processo di **classificazione** è stato un elemento fondamentale del progetto. Dopo il preprocessing dei dati, che ha incluso la pulizia e la trasformazione delle informazioni grezze, sono stati generati degli embeddings per rappresentare le caratteristiche latenti dei contenuti. Questi embeddings hanno costituito la base per la successiva analisi dei dati e per le attività di apprendimento supervisionato. La classificazione è stata eseguita utilizzando diversi modelli di machine learning e, dai risultati ottenuti, i modelli ***Random Forest*** e ***XGBoost*** hanno riportato performance ottimali per il soddisfacimento del nostro obiettivo.

Un altro pilastro del progetto è stato lo sviluppo di un sistema di **raccomandazione**. Utilizzando tecniche di ***content-based filtering***, il sistema è stato in grado di suggerire contenuti basati sulle preferenze espresse dagli utenti e sulle caratteristiche intrinseche dei titoli già apprezzati. Questo approccio ha garantito raccomandazioni personalizzate e rilevanti, migliorando l'esperienza di scoperta dei contenuti sulla piattaforma. Inoltre, il sistema ha dimostrato un'alta capacità di adattarsi dinamicamente al feedback degli utenti, permettendo di affinare continuamente le raccomandazioni in base ai loro gusti in evoluzione.

Lo sviluppo di una base di conoscenza (KB) in Prolog ha arricchito il sistema di raccomandazione con una componente di ragionamento logico. Questa KB ha memorizzato dati e regole estratti dal dataset, per migliorare la coerenza delle raccomandazioni.

In conclusione, il progetto realizzato introduce un sistema integrato che combina efficacemente tecniche di machine learning e logica basata su regole per l'analisi, la classificazione e la raccomandazione di contenuti, migliorando la scoperta di nuovi contenuti, offrendo un valore aggiunto significativo attraverso la sua capacità di evolversi e adattarsi continuamente, garantendo un'esperienza di fruizione dei contenuti più coinvolgente e soddisfacente.

Riferimenti Bibliografici

Link del dataset:

https://www.kaggle.com/datasets/shivamb/amazon-prime-movies-and-tv-shows?resource=download

1. Gomez-Uribe, C. A., & Hunt, N. (2015). "The Netflix Recommender System: Algorithms, Business Value, and Innovation". *ACM Transactions on Management Information Systems (TMIS)*, 6(4), 1-19.
2. Amatriain, X., & Basilico, J. (2012). "Netflix Recommendations: Beyond the 5 stars". *The Netflix Tech Blog*.
3. Kumar, A., Sarwat, M., & Briscoe, E. (2017). "Data-driven Content Creation: Leveraging Machine Learning at Netflix". *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1-5.
4. Gupta, U., & Leiserson, C. E. (2019). "The Netflix Edge: How Data-Driven Decision-Making Is Changing the Entertainment Industry". *IEEE Computer*, 52(10), 38-45.
5. Chen, C., & Guestrin, C. (2016). "XGBoost: A Scalable Tree Boosting System". *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794.
6. Ciocca, F., & Franceschelli, M. (2020). "Ethical Implications of Machine Learning in Content Recommendation Systems: The Case of Netflix". *Journal of Business Ethics*, 163(4), 635-648.
7. *Creare Grafici in Pandas | Codegrind*, https://codegrind.it/documentazione/pandas/grafici.
8. *NLTK: Libreria Python per l'Elaborazione del Linguaggio Naturale*, https://www.riccardodebernardinis.com/blog/nltk-libreria-python-per-lelaborazione-del-linguaggio-naturale/.