Immagine che contiene Elementi grafici, grafica, testo, clipart

Descrizione generata automaticamente

**Documentazione del caso di studio**

**Gruppo di lavoro:**

- Saracino Lorenzo – 746459 - l.saracino20@studenti.uniba.it

- Rosmarino Fabrizio – 758447 – f.rosmarino@studenti.uniba.it

**Repository github:**

Immagine che contiene emblema, simbolo, Carattere, logo

Descrizione generata automaticamente

**AA 2023-2024**

INDICE

[INTRODUZIONE 3](#_Toc169456884)

[ELENCO DI ARGOMENTI DI INTERESSE 3](#_Toc169456885)

[Apprendimento non supervisionato 3](#_Toc169456886)

[Apprendimento supervisionato 5](#_Toc169456887)

[K-Nearest Neighbors (KNN) 8](#_Toc169456888)

[Random Forest 9](#_Toc169456889)

[Support Vector Machine (SVM) 10](#_Toc169456890)

[Valutazioni dei modelli 11](#_Toc169456891)

[Knowledge Base – Creazione ed utilizzo 12](#_Toc169456892)

# INTRODUZIONE

Per lo sviluppo del progetto gamepy è stato utilizzato il linguaggio di programmazione Python, ponendo l’obiettivo principale sull’analisi dei videogame. I dati relativi sono stati raccolti dal sito Metacritic, il quale è un sito che aggrega recensioni dedicate a videogiochi, e non solo, anche film, musica e serie TV. Il dataset utilizzato per lo sviluppo è stato preso dal sito di kaggle, si riporta il link: <https://www.kaggle.com/datasets/brunovr/metacritic-videogames-data/data>.

I dati sono stati raccolti, così come riportato dalla descrizione dell’autore del dataset, attraverso il “web scraping”, il quale è un processo di automazzato di estrazione di informazioni da siti web. Questo è stato fatto attraverso uno script in Python utilizzando librerie come  requests, bs4.BeautifulSoup, re and pandas. il quale ha raccolto i dati presenti nel csv.

Si riporta la struttura del dataset, in particolare si riportano feature e il loro significato:

- name: il nome del gioco.

- platform: la piattaforma disponibile per il gioco.(playstation, xbox, nintendo..)

- r-date: la data di rilascio del gioco.

- score: media dei voti dati dai critici del sito metascore.

- user score: media dei voti dagli utenti nel sitoweb.

- developer: la casa produttrice del gioco.

- genre: il/i genere/generi del videogioco.

- players: modalità di gioco(1 player, up to 23…)(alcuni giochi potrebbero non avere questa informazione).

- critics: il numero di critiche che il gioco ha ricevuto.

Infine, tutti i dati nel dataset sono stati collezionati il 10 Novembre 2020, ed esso contiene giochi che vanno dal 1996 al 2020.

Inoltre, un altro obiettivo posto, è stato quello di predire e determinare il successo dei videogiochi. Infatti, è presente nel path gamepy/datasets/ un secondo csv, denominato game-data-with-success. Si è costruito un file prolog denominato games\_kb.pl il quale ci permette di interrogare la knowledge base realizzata.

Un'altra caratteristica del progetto è presente nel path gamepy/result dove all’interno è presente un file csv (game-with-constraints), il quale rappresenta una playlist, elenco di giochi che rispettano determinati vincoli.

In chiusura di questo paragrafo si vuole sottolineare che per alcune operazioni la colonna players, la quale indica la modalità di gioco, è stata mappata a causa delle varie modalità che poteva contenere. Il mapping si è fatto stabilendo una gerarchia, le quali colloca le varie informazioni in 4 modalità stabilite:

- singleplayer

- multiplayer

- co-op mode

- no-info, sono quei giochi i quali non si conosce la modalità.

# ELENCO DI ARGOMENTI DI INTERESSE

Gli argomenti che sono stati oggetto del corso di Ingegneria della Conoscenza e che sono stati riproposti e applicati in gamepy sono:

* Apprendimento non supervisionato: è stato utilizzato il K-meas algoritmo di hard-clustering, la quale è una forma di clustering più rigida in cui ogni punto appartiene completamente a un cluster specifico. I cluster hanno fatto emergere particolari caratteristiche dove ad ognuno di esse si è attribuito un risultato. Infine il K-means ha fatto emerge alcune anomalie.
* Apprendimento non supervisionato: sono state adottate diverse tecniche di machine learning supervisionato per affrontare compiti di classificazione, inoltre oltre alla varie tecniche di pre-processing e pulizia dei dati, si addestrano i modelli di machine learning per predire se un videogioco avrà successo, e si visualizzano i risultati delle predizioni attraverso metriche di valutazione. Gli algoritmi utilizzati sono stati K-Nearest Neighbors (KNN), Random Forest, Support Vector Machina (SVM).
* Rappresentazione e ragionamento: in questa soluzione, viene costruita una base di conoscenza, integrando il Prolog in python grazie alla librerie Pyswip. L’approccio utilizzato è stato quello di creare delle query le quali hanno consentito di inferire nuove informazioni a partite da quelle ricavabili in modo standard, arricchendo così la compensione e l’analisi dell’obiettivo di interesse.
* Risoluzione di un CSP: attraverso il Random Walk e il Simulated Annealing i quali sono entrambi algoritmi di ricerca e ottimizzazione che possono essere adattati per risolvere problemi di soddisfacimento dei vincoli, abbiamo creato un elenco variabile che rappresenta una playlist di giochi che rispetta determinati vincoli stabiliti

# Apprendimento non supervisionato

L'obiettivo è stato quello di esplorare e raggruppare i videogiochi in cluster omogenei basati su caratteristiche chiave come le valutazioni dei giocatori (score), il numero di giocatori e la piattaforma di gioco. Utilizzando l'algoritmo K-Means, abbiamo cercato di identificare pattern e segmentazioni all'interno dei dati per comprendere meglio il panorama dei videogiochi in base a queste caratteristiche.

**Caricamento e Pulizia dei Dati**

Inizialmente, abbiamo caricato i dati da un file CSV contenente informazioni dettagliate sui videogiochi, comprese le colonne per piattaforma, score, numero di giocatori e altre informazioni pertinenti. Abbiamo eseguito una pulizia preliminare eliminando le righe con valori mancanti nelle colonne cruciali come score e numero di giocatori. Successivamente, abbiamo convertito la colonna 'players' in un formato numerico, eliminando eventuali caratteri non numerici che potrebbero essersi presentati.

**Unificazione e filtraggio delle Piattaforme**

Nel dataset originale, i giochi sono associati a diverse piattaforme di gioco. Tuttavia, alcune di queste piattaforme sono versioni diverse della stessa famiglia, come ad esempio Xbox One e Xbox 360, o PlayStation 2, PlayStation 3 e PlayStation 4. Quindi, per semplificare l'analisi e migliorare la coerenza nei dati, abbiamo deciso di raggruppare le diverse piattaforme di gioco in categorie più ampie:

Le piattaforme Xbox One e Xbox 360 sono state unite sotto la categoria 'Xbox'.

Le piattaforme PlayStation 2, PlayStation 3 e PlayStation 4 sono state unite sotto la categoria 'PlayStation'.

Le piattaforme Wii, Wii U, Nintendo DS e Nintendo 3DS sono state raggruppate sotto la categoria 'Nintendo'.

La piattaforma PC è rimasta come 'PC'.

Dopo l'unificazione, è stato necessario limitare l'analisi a un sottoinsieme di piattaforme di gioco principali per garantire che i dati fossero più gestibili e focalizzati sulle piattaforme più comuni. Si è deciso quindi di restringere il tipo di piattaforme solo a quelle riportate precedentemente, raggruppate nelle rispettive categorie.

**Selezione delle Caratteristiche**

Dopo aver unificato le piattaforme, abbiamo selezionato le caratteristiche rilevanti per l'analisi: 'platform' (piattaforma di gioco), 'score' (valutazione del gioco) e 'players' (numero di giocatori). Queste caratteristiche sono state considerate per capire come i giochi si differenziano tra loro in base alle piattaforme e alle valutazioni.

**Preprocessing dei Dati**

Abbiamo applicato il preprocessing dei dati per preparare le caratteristiche numeriche e categoriche per l'analisi di clustering:

Le caratteristiche numeriche 'score' e 'players' sono state standardizzate utilizzando StandardScaler per garantire che avessero la stessa scala e non dominassero il processo di clustering.

Le caratteristiche categoriche sono state trasformate utilizzando OneHotEncoder per gestire le variabili categoriche come la piattaforma di gioco, trasformandole in una forma numerica adatta all'analisi.

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

Per identificare il numero ottimale di cluster da utilizzare con l'algoritmo K-means, abbiamo eseguito un'analisi della curva a gomito. Questo ci ha permesso di valutare come l'inerzia (somma dei quadrati delle distanze dei punti dai centroidi) varia al variare del numero di cluster K. Sulla base di questa analisi, abbiamo scelto un valore ottimale di K che bilanciasse la complessità del modello con la sua capacità di identificare pattern significativi nei dati.

Immagine che contiene linea, diagramma, Diagramma, testo

Descrizione generata automaticamente

**Clustering con K-means**

Utilizzando il numero ottimale di cluster identificato (3), abbiamo creato una pipeline che include il preprocessing dei dati e l'applicazione dell'algoritmo K-means. Abbiamo adattato questa pipeline ai dati per assegnare ciascun videogioco al suo cluster corrispondente in base alle caratteristiche di score, numero di giocatori e piattaforma.

**Interpretazione dei Risultati**

I risultati dell'analisi di clustering sono stati interpretati visualmente attraverso grafici come lo scatter plot. Questi grafici hanno mostrato la distribuzione dei videogiochi all'interno dei cluster identificati, con punti colorati in base al cluster di appartenenza e con stili diversi per identificare facilmente la piattaforma di gioco. Questo ci ha aiutato a comprendere come i giochi sono raggruppati in base alle caratteristiche di score e numero di giocatori, nonché a identificare eventuali tendenze o segmentazioni significative tra le diverse piattaforme.

**Conclusioni**

L'analisi di clustering dei videogiochi utilizzando K-means ha fornito una panoramica dettagliata su come i giochi possono essere raggruppati in base alle loro caratteristiche principali. Questo approccio ha il potenziale per supportare decisioni strategiche nel settore dei videogiochi, aiutando gli sviluppatori e i distributori a comprendere meglio il comportamento dei consumatori e le preferenze di gioco in diverse piattaforme.

Immagine che contiene testo, schermata, numero, linea

Descrizione generata automaticamente

**Interpretazione del grafico**

1. **Assi:**
   * L'asse delle ascisse (x) rappresenta il punteggio del gioco (score).
   * L'asse delle ordinate (y) rappresenta il numero di giocatori (players).
2. **Colori dei punti:**
   * I punti sono colorati in base al cluster a cui appartengono:
     + **Cluster 0**: Rosso
     + **Cluster 1**: Blu
     + **Cluster 2**: Verde
3. **Stile dei punti:**
   * I punti sono stilizzati in base alla piattaforma del gioco:
     + **PlayStation**: Cerchio
     + **Xbox**: Croce
     + **PC**: Quadrato
     + **Nintendo**: Croce più (+)

Dalla visualizzazione del clustering dei videogiochi in base ai loro punteggi (score) e al numero di giocatori (players), possiamo fare diverse deduzioni:

* **Cluster 0 (Rosso)**:
  + Generalmente contiene giochi con un punteggio medio-alto (score circa tra 60 e 80).
  + La maggior parte dei giochi ha un numero di giocatori basso (players meno di 40).

Questo cluster può rappresentare giochi che sono generalmente accettati ma non eccezionali. Potrebbero essere giochi di nicchia o con un pubblico limitato.

* **Cluster 1 (Blu)**:
  + Include giochi con un punteggio basso-medio (score tra 20 e 60).
  + I giochi in questo cluster hanno un numero di giocatori basso (players meno di 40).

Questi giochi potrebbero essere considerati meno popolari o di qualità inferiore. Potrebbero non avere un grande impatto sul mercato e potrebbero essere titoli che non sono stati bene accolti dai giocatori.

* **Cluster 2 (Verde)**:
  + Comprende giochi con un punteggio alto (score sopra 60, molti tra 80 e 100).
  + I giochi in questo cluster tendono ad avere un numero di giocatori molto alto (players sopra 60, alcuni oltre 160).

Questo cluster rappresenta i giochi di grande successo, sia in termini di qualità che di popolarità. Questi titoli hanno sia un alto punteggio che un grande numero di giocatori, indicando un forte apprezzamento e una vasta base di utenti.

**Rilevazione di Dati anomali**

Ci sono giochi che, nonostante punteggi bassi, hanno un numero alto di giocati, o viceversa, questi potrebbero essere considerati anomali. Per esempio, giochi nel cluster 1, presentano un numero di giocatori molto più alto del previsto, questo potrebbe indicare un’anomalia nel modo in cui i dati sono stati raccolti o potrebbe essere un particolare titolo che attira i giocatori nonostante le recensioni negative.

# Apprendimento supervisionato

**Introduzione**

Il file sorgente apprendimentoSupervisionato.py utilizza diverse tecniche di machine learning per classificare i videogiochi in base al loro successo, utilizzando dati come valutazioni dei giocatori, valutazioni critiche e numero di utenti. Sono stati implementati tre modelli di classificazione supervisionata: K-Nearest Neighbors (KNN), Random Forest e Support Vector Machine (SVM). L'obiettivo è valutare e confrontare le prestazioni di questi modelli utilizzando misure di accuratezza, report di classificazione, matrici di confusione e curve ROC.

**Caricamento e Pulizia dei Dati**

Il dataset dei videogiochi è stato caricato da un file CSV e successivamente pulito per rimuovere righe con valori mancanti nelle colonne critiche come 'user score', 'genre', 'critics' e 'users'. La colonna 'user score' è stata gestita convertendo il valore 'tbd' in NaN e successivamente sostituendo i valori NaN con la media delle valutazioni dei giocatori disponibili. È stata inoltre definita e aggiunta al file CSV una nuova colonna 'success' basata su criteri specifici di successo del gioco.

Immagine che contiene testo, Software multimediale, schermata

Descrizione generata automaticamente

**Preprocessing dei Dati**

Le variabili categoriali sono state codificate utilizzando LabelEncoder, mentre le variabili numeriche sono state standardizzate utilizzando StandardScaler. I dati sono stati poi suddivisi in training set e test set utilizzando train\_test\_split, con una proporzione del 70% per l’addestramento e del 30% per il test.

**Addestramento e Valutazione dei Modelli**

Per effettuare l’addestramento, abbiamo deciso di utilizzare la funzione train\_and\_evaluate\_model, la quale svolge un ruolo cruciale. Questa funzione prende in input un modello specifico (come KNN, Random Forest o SVM), addestra il modello utilizzando il set di addestramento (X\_train e y\_train), esegue le previsioni sul set di test (X\_test), calcola l'accuratezza delle previsioni e genera un report dettagliato delle prestazioni del modello.

Per visualizzare e valutare i risultati ottenuti dal modello, abbiamo utilizzato la funzione print\_results, la quale stampa diversi risultati:

Accuracy: l'accuratezza del modello sui dati di test.

Classification Report: una panoramica dettagliata di precision, recall, F1-score e support per ciascuna classe di output.

Training e Test Accuracy: l'accuratezza del modello sui dati di addestramento e test.

Feature Importances: se disponibile, mostra l'importanza delle caratteristiche utilizzate dal modello per prendere decisioni.

Confusion Matrix: una matrice che mostra i veri positivi, i falsi positivi, i veri negativi e i falsi negativi.

ROC Curve: una curva che visualizza il tasso di veri positivi rispetto al tasso di falsi positivi, con l'Area Under the Curve (AUC) indicata nel titolo del grafico.

**Menu di Selezione del Modello**

All'avvio del programma, viene visualizzato un menu che consente all'utente di selezionare il modello da utilizzare (KNN, Random Forest, SVM) o di uscire dall'applicazione. Questa interfaccia utente rende il processo di selezione del modello interattivo e accessibile.

**Esecuzione e Visualizzazione dei Risultati**

Il menu principale guida l'utente attraverso la selezione del modello e l'esecuzione dell'addestramento e valutazione del modello scelto. I risultati vengono visualizzati utilizzando grafici per la matrice di confusione e la curva ROC, mentre i report di classificazione e altre metriche vengono stampati direttamente nella console.

**Conclusioni**

Questo approccio integrato per l'analisi e la valutazione dei modelli di classificazione dei videogiochi fornisce una panoramica completa delle prestazioni dei modelli selezionati. L'uso di tecniche avanzate di machine learning e visualizzazione dei dati permette di comprendere meglio quali caratteristiche influenzano il successo dei videogiochi e come i modelli possono essere utilizzati per prendere decisioni informative nel settore dei videogiochi

### K-Nearest Neighbors (KNN)

Immagine che contiene testo, schermata

Descrizione generata automaticamente

Immagine che contiene testo, schermata, diagramma, Rettangolo

Descrizione generata automaticamenteImmagine che contiene testo, linea, schermata, Diagramma

Descrizione generata automaticamente

L'AUC per la curva ROC è pari a 0.9236, che misura la capacità del modello di discriminare tra classe positiva e classe negativa.

Un valore di AUC vicino a 1 indica un modello con buone prestazioni discriminative.

Nella matrice di confusione fornita:

- 4705 casi di gioco non di successo sono stati correttamente classificati.

- 65 casi di gioco non di successo sono stati classificati erroneamente come gioco di successo.

- 423 casi di gioco di successo sono stati correttamente classificati.

- 191 casi di gioco di successo sono stati classificati erroneamente come gioco non di successo.

### Random Forest

Immagine che contiene testo, schermata, menu

Descrizione generata automaticamenteImmagine che contiene testo, schermata, diagramma, Rettangolo

Descrizione generata automaticamenteImmagine che contiene testo, schermata, linea, Diagramma

Descrizione generata automaticamente

L’AUC è pari a 0.9975 ~ 1, il che indica un’elevata capacità discriminante del modello.

Nella matrice di confusione fornita:

- 4732 casi di gioco non di successo sono stati correttamente classificati.

- 38 casi di gioco non di successo sono stati classificati erroneamente come gioco di successo.

- 607 casi di gioco di successo sono stati correttamente classificati.

- 7 casi di gioco di successo sono stati classificati erroneamente come gioco non di successo.

### Support Vector Machine (SVM)

Immagine che contiene testo, schermata

Descrizione generata automaticamenteImmagine che contiene testo, schermata, diagramma, design

Descrizione generata automaticamenteImmagine che contiene testo, linea, Diagramma, schermata

Descrizione generata automaticamente

L’AUC è pari a 0.949 , il che indica una buona capacità discriminante tra classi.

Nella matrice di confusione fornita:

- 4723 casi di gioco non di successo sono stati correttamente classificati.

- 47 casi di gioco non di successo sono stati classificati erroneamente come gioco di successo.

- 372 casi di gioco di successo sono stati correttamente classificati.

- 242 casi di gioco di successo sono stati classificati erroneamente come gioco non di successo.

### Valutazioni dei modelli

**KNN**

Il modello KNN ha un'accuracy complessiva elevata sia sul set di addestramento che su quello di test, indicando una buona capacità di generalizzazione.

La precision per entrambe le classi è abbastanza alta, con un'eccezione nel recall per la classe "gioco di successo", che è inferiore rispetto alla classe "gioco non di successo".

L'F1-score riflette un buon compromesso tra precision e recall per la classe "gioco non di successo", mentre per la classe "gioco di successo" c'è spazio per miglioramenti.

La matrice di confusione indica che il modello tende a predire correttamente i giochi non di successo, ma ha un recall più basso per i giochi di successo.

L'AUC della curva ROC suggerisce che il modello ha una buona capacità discriminativa.

In generale, il modello KNN sembra essere efficace nella predizione del successo dei giochi sulla base delle caratteristiche fornite, ma potrebbe beneficiare di ulteriori ottimizzazioni per migliorare il recall nella classe "gioco di successo" e quindi migliorare l'F1-score complessivo.

**Random Forest**

Il modello Random Forest ha dimostrato eccellenti capacità predittive, con un'accuracy molto alta su entrambi i set di dati (addestramento e test). Le metriche nel Classification Report indicano una buona capacità del modello di distinguere tra le due classi di interesse ("gioco non di successo" e "gioco di successo"). Inoltre, la matrice di confusione e l'AUC confermano la robustezza e l'efficacia del modello nelle sue predizioni.

Questi risultati suggeriscono che il modello Random Forest è ben addestrato e può essere utilizzato con fiducia per predire il successo dei videogiochi basandosi sulle caratteristiche fornite.

Inoltre, il modello ha attribuito l’importanza delle features nel fare le predizioni e si sono ottenute le seguenti percentuali:

- users: 56.62%

- user score: 28.42%

- critics: 12.36%

- genre: 2.60%

Questo indica che il numero di utenti che hanno recensito il gioco (users) è stata la caratteristica più influente nelle predizioni del modello, seguita da user score, critics e genre.

**SVM**

Il modello SVM ha una buona precisione nel predire i giochi non di successo, ma mostra una performance inferiore nel predire i giochi di successo, come indicato da una recall più bassa. Questo potrebbe essere dovuto a uno sbilanciamento nelle classi dei dati o alla necessità di ulteriori ottimizzazioni del modello.

In sintesi, il modello SVM mostra una buona accuratezza generale, ma potrebbe beneficiare di miglioramenti nella capacità di predire correttamente i giochi di successo, soprattutto aumentando il recall per questa classe.

# Knowledge Base – Creazione ed utilizzo

**Creazione della Knowledge Base (createKB)**

Il file sorgente createKB.py è stato implementato per la creazione della Knowledge Base si concentra sull'elaborazione di un dataset di videogiochi in formato CSV e sulla sua trasformazione in fatti utilizzabili in Prolog.

Iniziamo con un'analisi dettagliata di ciò che il codice fa e come contribuisce alla creazione della KB.

**1. Normalizzazione dei Nomi dei Giochi e dei Generi:** Il primo passo è la normalizzazione dei nomi dei giochi e dei generi associati. Questo processo include la sostituzione degli apostrofi con underscore e la conversione di punti e due punti in trattini per uniformare il formato.

**2. Pulizia e Trasformazione dei Generi:** Ogni gioco può appartenere a più generi separati da virgole nel dataset CSV. Il codice separa questi generi e li pulisce da spazi vuoti e caratteri non desiderati, come virgolette singole.

**3. Scrittura dei Fatti Prolog:** I dati puliti vengono quindi utilizzati per generare fatti Prolog che rappresentano l'associazione tra i nomi dei giochi e i loro generi. Se un fatto è già stato scritto, viene ignorato per evitare duplicati.

**4. Selezione e Scrittura dei Giochi di Alta Qualità:** Il codice identifica i giochi con un punteggio superiore a 80 e almeno 100 critiche e li trasforma in fatti Prolog aggiuntivi per rappresentare giochi di successo nel tempo.

**5. Salvataggio della KB:** Infine, tutti i fatti sono scritti in un file di testo Prolog (games\_kb.pl) per essere utilizzati successivamente, quest’ultimo viene salvato nella directory /src per dei conflitti riscontrati in fase di implementazione della KB.

**Utilizzo della Knowledge Base (useKB)**

Dopo aver creato la KB, si è implemantato il file sorgente useKB.py per utilizzare questa base di conoscenza interattivamente, per effettuare query e ottenere informazioni specifiche sui giochi.

**1. Verifica dell'Installazione di SWI-Prolog:** Il primo passo nella parte di utilizzo è la verifica dell'installazione di SWI-Prolog sul sistema dell'utente. Questa verifica è cruciale perché il sistema deve essere pronto per eseguire query Prolog e non generare errori.

**2. Interfaccia Interattiva con l'Utente:** Una volta verificata l'installazione, l'utente viene presentato con un menu interattivo che offre due opzioni principali:

* **Mostrare 10 giochi di un genere scelto:** L'utente può selezionare un genere tra quelli disponibili e ottenere una lista di giochi appartenenti a quel genere. Nel caso di inserimento di un genere presente nella lista e avente, un numero di giochi ad esso associati minore di 10, all’ utente verrà mostrato un messaggio in cui viene messo al corrente di questa condizione, seguito dalla stampa dei giochi relativi a quel genere.
* **Mostrare 10 giochi con maggiore successo a partire da un certo anno:** L'utente può specificare un anno compreso tra il 1996 e il 2020 e ricevere una lista di giochi con un punteggio superiore a 80 e un numero significativo di critiche, usciti da quell'anno in poi. Nel caso di inserimento scorretto, verrà richiesto all’ utente di reinserire nuovamente un anno compreso nell’ intervallo definito.
* **Mostrare 5 migliori giochi di un developer scelto:** L’ utente può inserire il nome di una casa produttrice tra quelle disponibili e ottenere una lista dei 5 migliori giochi prodotti da questa. I giochi presentati saranno quelli prodotti dalla casa produttrice inserita, e aventi un successo molto alto sul mercato (aventi uno score maggiore o uguale a 90, basato su un numero esiguo di recensioni e votazioni ricevute dagli utenti).Nel caso di inserimento di una casa produttrice avente un minor numero di giochi di grande successo prodotti , verrà mostrato all’utente un messaggio con cui viene messo al corrente della situazione, seguito dalla stampa dei top giochi disponibili della casa produttrice.
* **Mostrare 10 giochi di una modalità di gioco scelta:** L’ utente può inserire il tipo di modalità di gioco, scegliendo tra Singleplayer, Multiplayer, co-op mode e no mode, e ottenendo una lista di giochi della modalità inserita.

**3. Implementazione delle Query:** Le query sono implementate utilizzando la libreria pyswip, che permette di eseguire query Prolog direttamente dal codice Python. Le risposte alle query sono presentate all'utente nella console.

**4. Gestione delle Eccezioni:** Il codice gestisce anche possibili errori, come input non validi o input scritti usando caratteri maiuscoli o minuscoli.

**Risultati e Considerazioni Finali.**

Attraverso il processo di creazione e utilizzo della KB, abbiamo raggiunto diversi obiettivi importanti:

* **Organizzazione dei Dati:** Il dataset di videogiochi è stato trasformato in una forma più strutturata e utilizzabile attraverso la normalizzazione e la pulizia dei dati.
* **Creazione di Fatti Prolog:** I dati trasformati sono stati rappresentati come fatti Prolog, consentendo interrogazioni efficienti e specifiche sulla base di conoscenza.

**//possiamo scrivere di più qui in base alle query create successivamente**

In conclusione, la combinazione di creazione e utilizzo della KB per i videogiochi non solo facilita l'organizzazione dei dati, ma fornisce anche uno strumento potente per l'analisi e l'esplorazione dei dati stessi. Questo approccio non solo migliora l'efficienza nella gestione delle informazioni, ma anche la comprensione e l'interpretazione dei dati stessi, sottolineando l'importanza di un'implementazione accurata e dettagliata di una KB in contesti applicativi reali.

## RISOLUZIONE DI UN CSP – soddisfazione dei vincoli

L’obiettivo era quello di creare una playlist di giochi con particolari caratteristiche. Per conseguire tale scopo, l’approccio adottato ha trattato il problema come una questione di ottimizzazione dei vincoli, che includono:

1. Esattamente 10 giochi nella selezione
2. Numero totale di critics non superiore a 300.
3. Media del punteggio degli utenti almeno 7.0.
4. Non più di due giochi dello stesso genere.

Per ottenere la seguente playlist sono stati utilizzati due algoritmi di ottimizzazione:

**Random Walk**, il quale seleziona casualmente 10 giochi dal dataset e valuta il numero di violazioni dei vincoli. Se trova una selezione con meno violazioni rispetto alla migliore selezione trovata finora, aggiorna la migliore selezione. L'algoritmo continua per un numero fisso di iterazioni (max\_iter) o fino a quando non trova una selezione che soddisfa tutti i vincoli (zero violazioni).

**Simulated Annealing**, L'algoritmo Simulated Annealing inizia con una soluzione casuale e utilizza una tecnica di annealing per esplorare nuove soluzioni. La temperatura iniziale è impostata ad un valore elevato (temp), fattore di energia(1000), e viene gradualmente ridotta ad ogni iterazione moltiplicandola per un fattore (alpha), fattore di raffreddamento (0.99). Questo consente all'algoritmo di accettare occasionalmente soluzioni peggiori per evitare di rimanere bloccati in minimi locali. Se una nuova soluzione ha meno violazioni rispetto alla migliore soluzione trovata finora, questa viene aggiornata come la migliore soluzione. L'algoritmo termina dopo un numero fisso di iterazioni (max\_iter) o quando la temperatura (che controlla la probabilità di accettazione di soluzioni peggiori) raggiunge un valore molto basso.

Inoltre si è deciso di eseguire 10 iterazioni per ogni algoritmo per ottenere una stima più accurata delle prestazioni dei due algoritmi e delle loro capacità di trovare soluzioni che soddisfano i i vincoli. Poiché gli algoritmi utilizzano elementi casuali nella selezione dei giochi (in particolare il Random Walk), eseguire più iterazioni riduce la variabilità nei risultati e fornisce una valutazione media che è più rappresentativa delle capacità dell'algoritmo.

Un esempio di output del seguente file sorgente è:

l'utente può scegliere tra Random Walk e Simulated Annealing. Dopo 10 esecuzioni

dell'algoritmo selezionato, il programma stamperà:

* La media del tempo di esecuzione dell'algoritmo.
* La media delle violazioni dei vincoli.
* Salverà la migliore selezione trovata in un file CSV.
* Verificherà e stamperà se i vincoli sono stati soddisfatti per la migliore selezione trovata.

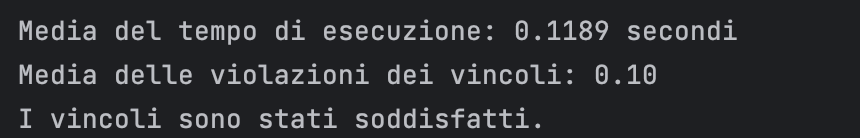
E’ importante soffermarsi sulla media delle violazioni, quest’ultima è una metrica per valutare la qualità delle soluzioni prodotte dagli algoritmi di ottimizzazione in termini di soddisfacimento dei vincoli. In particolare si ha:

* **Violazioni dei vincoli**: Ogni volta che una selezione di giochi non soddisfa uno dei vincoli specificati (ad esempio, superare il numero massimo di critici, avere un punteggio medio degli utenti inferiore a 7.0, ecc.), si conta come una violazione. Ogni violazione aumenta il conteggio totale delle violazioni.
* **Media delle violazioni**: Dopo aver eseguito l'algoritmo di ottimizzazione più volte (ad esempio, 10 volte), si calcola la media delle violazioni dei vincoli su tutte le esecuzioni.

Esempi di risultati ottenuti: Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente

In genere durante i vari test fatti la media delle valutazioni non hai mai prodotto risultati lontani dallo zero, questo significa che, in media, le soluzioni trovate dagli algoritmi di ottimizzazione violano i vincoli in misura molto ridotta. In altre parole, su 10 esecuzioni dell’algoritmo, il numero medio di violazioni è ad esempio 0.10, il che implica che le soluzioni sono generalmente molto vicine a soddisfare tutti i vincoli. Una media vicino a 0 è un indicatore di buone prestazioni dell’algoritmo, suggerendo che la maggior parte delle soluzioni trovate sono valide e rispettano quasi tutti i vincoli e quindi l’algoritmo ha buone performance.

**Conclusioni**

Con l’implementazione di due algoritmi avremo sicuramento casualità nella scelta delle soluzioni grazie al Random Walk, ma avremo anche un graduale miglioramento grazie al Simulated Annealing il quale incorpora una strategia più sofisticata che permette una graduale miglioramento delle soluzioni infatti quest’ultimo è in grado di superare minimi locali grazie alla sua capacità di accettare soluzioni peggiori, migliorando le possibilità di trovare una soluzione globalmente ottima rispetto a molti problemi di ottimizzazione.