Caso di Studio- ICON 21-22

Titolo: Classification system di prodotti videoludici mediante tecniche di apprendimento supervisionato e recommender system basato su clustering

Studente: Enrico Sinisi 662260

Repository: https://github.com/Enrico1398/Icon22

INTRODUZIONE

L'obiettivo del caso di studio è quello di utilizzare le nozioni imparate durante il suddetto insegnamento e di applicarle in un contesto pseudo-reale, più nello specifico si è deciso di utilizzare il dataset "metacritic_game_info", disponibile sul sito Kaggle, condiviso dal portale Metacritic.com, il quale si occupa di pubblicare notizie relative al mondo dell'intrattenimento. Il dataset più nello specifico contiene i titoli videoludici con i più alti valori metacritic, valore ottenuto mediando i voti espressi sul prodotto da parte di tutte le più importanti testate giornalistiche del settore. Ognuno di questi titoli ha svariati attributi

Una precisazione doverosa sta nel dire che questo dataset è stato già utilizzato da me per un altro progetto riguardante un altro insegnamento, ovvero, Data mining. Naturalmente questi progetti pur condividendo lo stesso dataset, hanno obiettivi ben diversi.

Il progetto è stato suddiviso in svariate fasi che verranno mostrate in modo sequenziale.

PREPROCESSING

Questa fase mette alcune modifiche fatte a mano ed altre mediante funzioni messe a disposizione dalla libreria python Pandas.

Il dataset originale si presenta come una lista ordinata di data objects aventi come attributi: position, title, year, publisher,genre,platform,metascore, avg_userscore, no_player, dei quali solo position, year, metascore e avg_userscore sono numerici mentre i restanti sono stringhe.

Esempio Dataset originario:

7,Grand Theft Auto V,2014,RockstarNorth,Action Adventure	Modern	Open-World,XboxOne,97,7.8,Up to 30	
8,Grand Theft Auto V,2013,RockstarNorth,Modern	Action Adventure	Modern	Open-World,PlayStation3,97,8.3,Up to 16
9,Grand Theft Auto V,2013,RockstarNorth,Modern	Action Adventure	Modern	Open-World,Xbox360,97,8.3,Up to 16

Per quanto concerne la qualità del dataset, non sono presenti valori mancanti ma esso pecca di consistenza, poiché molti attributi, essendo stringhe nel dataset iniziale, contengono ridondanze, errori concettuali e di conseguenza sono stati ricondotti successivamente a valori nominali, basati su standard ben definiti. Il focus principale della fase di DATA PREPARATION è l'attributo 'genre'. L'attributo 'genre' è stato modificato poiché risultava ambiguo e confusionario. Come è possibile notare dall'esempio, nel dataset originario, titoli dello stesso publisher e con la stessa identica struttura videoludica avevano attributi differenti e ridondanti. Di conseguenza si è scelto di omologare tutti gli attributi 'genre' verso un unico standard chiaro: "**https://it.wikipedia.org/wiki/Categoria:Tipi_di_videogiochi**", affiancando ad esso il nuovo attributo 'characteristic' contente quella che è definibile una caratteristica distintiva presente all'interno del prodotto (ambientazione, modalità di gioco, etc.…), precedentemente presente in modo erroneo all'intero dell'attributo 'genre'. Infine anche gli attributi 'publisher' e 'no_players' sono stati modificati, infatti anche questi ultimi contenevano ridondanze ed errori, nello specifico per quanto riguarda il primo, i valori di ogni tupla di 'publisher' sono stati sostituiti, visto che erroneamente nel dataset sono stati inseriti gli studi di sviluppo e non i veri publisher dei prodotti, puntualizzazione necessaria poiché sono i publisher che, ad eccezione di studi di sviluppo indipendenti, hanno i diritti su un determinato prodotto e decidono a chi commissionarlo e il budget dello stesso, mentre per quanto concerne il secondo ,alcuni valori come già detto erano ridondanti e scritti in maniera diversi, di conseguenza ricondotti a una forma unica.

Esempio dataset modificato:

id	position	title	year	publisher	genre	characteri	platform	metascore	user_avg	no_players
1	1	The Legen	1998	Nintendo	Action Adv	Fantasy	Nintendo6	99	9.1	1 Player
2	2	Tony Hawl	2000	Activision	Sports	Skateboar	PlayStatio	98	7.4	not specified
3	3	Grand The	2008	RockstarG	Action Adv	Open-Wor	PlayStatio	98	7.5	1 Player
4	4	SoulCalibu	1999	Bandai	Fighting	3D	Dreamcast	98	8.6	1 to 2
5	5	Grand The	2008	RockstarG	Action Adv	Open-Wor	Xbox360	98	7.9	1 Player
6	6	Super Mar	2007	Nintendo	Platformer	3D	Wii	97	9.0	No Online Multiplaye
7	7	Super Mar	2010	Nintendo	Platformer	3D	Wii	97	9.1	No Online Multiplaye
8	8	Grand The	2014	RockstarG	Action Adv	Open-Wor	XboxOne	97	7.8	1 to 30
9	9	Grand The	2013	RockstarG	Action Adv	Open-Wor	PlayStatio	97	8.3	1 to 16
10	10	Grand The	2013	RockstarG	Action Adv	Open-Wor	Xbox360	97	8.3	1 to 16
11	11	Tony Hawl	2000	Activision	Sports	Skateboar	Dreamcast	97	6.2	not specified

Per quanto riguarda la dimensione del dataset, il dataset iniziale conteneva 5000 data objects, di conseguenza è stato necessario ridurre la dimensione del dataset a 488 data objects, visto che il mio obiettivo di questo caso di studio era di creare un classification system e recommender system che classificasse e consigliasse i prodotti più apprezzati di tutti i tempi, scegliendo così data objects con i valori dell'attributo 'metascore' >= 90, ottenendo così solo le eccellenze del settore.

Quelle elencate precedentemente sono le modifiche fatte a mano direttamente sul file csv.

Invece quelle fatte mediante funzioni della libreria pandas sono le seguenti:
-cancellazione delle colonne id e position
-discretizzazione della colonna 'year' definendo range ben definiti
-associazione di ogni attributo categorico ad un numerico mediante Label Encoder e ulteriore riconduzione di essi in categorico, poiché sia la SVM sia il Decision Tree messi a disposizione da scikit-learn non funzionano su attributi categorici.

Ulteriori operazioni sono state effettuate sul dataset, ma quest'ultime sono riguardanti il sistema che si decide di utilizzare.

SCELTA CLASSIFICATORI

Per creare il classification system in grado di classificare un prodotto dato in input dall'utente, si sono utilizzate tecniche di apprendimento supervisionato e confrontandole tra loro attraverso valutazioni espresse in base a precise metriche, si è scelto il classificatore più performante tra di essi.

ll processo di classificazione è suddiviso in tre fasi:
-training, si produce un modello da un insieme di dati detto training set
-stima dell'accuratezza, mediante metriche utili a stimare le performance su un insieme di dati diverso dal training set, inizialmente validation set mediante cross validation e poi test set classificando istanze di classe ignota
- scelta del classificatore più adatto per il sistema

La classificazione nel caso di studio è stata utilizzata con lo scopo di predire, tramite addestramento sul dataset precedentemente ottenuto, il genere di un videogioco fornito dall'utente. La feature target, ossia la label da predire è la colonna 'genre'.

Per procedere con la classificazione, ovviamente bisogna scegliere il classificatore più adatto, ossia quello performi ottimamente sia un training set e che sappia generalizzare su un test set. Nel machine learning esiste il teorema “No free Lunch” che afferma che non esiste un algoritmo che vada bene per qualsiasi problema e, di conseguenza, sono necessarie varie prove per trovare il modello di predizione più accurato, valutando le performance di ciascuna alternativa, al fine di trovare la più adatta.

Il metodo utilizzato è stato quello di considerare i classificatori le cui caratteristiche sembravano aderenti al caso di studio, per poi valutarne le performance utilizzando la cross validation, evitando che i classificatori scelti vadano in overfitting, ovvero, buonissime performance sul training set ma basse performance sul test set, successivamente effettuare un confronto tra i classificatori presi in esame. Una volta trovato il più performante, quest'ultimo viene stato utilizzato per le predizioni del sistema.

La scelta degli algoritmi da testare si è basata sulla mia curiosità di confrontare modelli molto diversi tra loro, inizialmente solo SVM e Random Forest a cui successivamente è stato aggiunto il Decision Tree.

Gli algoritmi più nello specifico sono i seguenti: -- **C-Support Vector Classification (SVC)** L’algoritmo SVM, seppur utilizzato per classificazioni binarie, può essere impiegato per problemi di classificazione multi-classe utilizzando la metodologia one-vs-one oppure la one-vs-rest. Nello specifico nella prima, si creano k(k-1) /2 classificatori, dove k è il numero di classi, che effettuano classificazione su coppie di classi, per poi assegnare come classe finale quella con più assegnazioni, mentre nella seconda si creano k classificatori quanto i k feature target e si fa un confronto tra quella selezionata e il resto scegliendo la majority class. Ognuna delle due metodologie è biased riguardo la majority class, problema che verrà discusso in seguito. La SVM si basa sull’idea di trovare un iperpiano che divida il set di dati in due o più classi su x dimensioni spaziali dove x è il numero di classi. I punti dati più vicini all’iperpiano sono detti vettori di supporto e rappresentano le possibili classi di appartenenza. Un iperpiano linearmente separabile è un iperpiano cui è semplice distinguere due classi, il problema è trovare quale tra le infinite rette che rappresentano l’iperpiano risulti ottimale, ossia quella che generi il minimo errore di classificazione su una nuova osservazione. Il metodo del kernel consente di modellare modelli non lineari di dimensioni superiori. Il suo scopo è di prendere i dati come input e trasformarli nella forma richiesta

qualora non sia possibile determinare un iperpiano linearmente separabile. In questo caso useremo il metodo kernel con l'SVM poiché il modello in questione è non lineare. I principali vantaggi di questo algoritmo sono i seguenti: -

- Efficace in dimensioni spaziali elevate
- Efficienza della memoria, poiché solo un sottoinsieme dei punti di allenamento viene utilizzato nel processo decisionale effettivo di assegnazione
- Versatilità, grazie alla capacità di applicare nuovi kernel portando a una maggiore performance di classificazione.

I suoi principali svantaggi sono:

- Interpretazione non semplice
- Metodo non probabilistico, poiché il classificatore funziona posizionando gli oggetti sopra e sotto un iperpiano di classificazione, non esiste un'interpretazione probabilistica diretta per l'appartenenza al gruppo
- Richiede adattamento per tutte quelle variabile non numeriche

-Random Forest Classifier La Random Forest è un bagging di alberi di decisione, i quali vengono uniti per ottenere una previsione più accurata e stabile. Ogni albero in una random forest impara da un campione casuale di dati. I campioni vengono disegnati con la sostituzione, nota come bootstrap, ovvero, alcuni campioni verranno utilizzati più volte in un singolo albero. L'idea è che addestrando ciascun albero su campioni diversi, sebbene ogni albero possa presentare una varianza elevata rispetto a una particolare serie di dati di addestramento, nel complesso la foresta avrà una varianza inferiore, in modo da avere le predizioni finali vicine al risultato.

I vantaggi del Random Forest sono:

-Versatilità, utilizzabile sia per problemi di regressione che di classificazione, funzionando bene con una combinazione di caratteristiche numeriche e categoriche, e l'avere gli iperparametri predefiniti ottimali, perché producono un buon risultato di previsione, consentendo anche un notevole miglioramento di essi, e di conseguenza della previsione. In generale, questi algoritmi sono veloci da addestrare, ma piuttosto lenti nel creare previsioni una volta che sono stati addestrati, poiché richiedono un alto numero di alberi per ottenere risultati accurati

-Decision Tree L'albero di decisione è metodo non parametrico, utilizzabile sia per task di classificazione sia per regressione. L'algoritmo messo a disposizione da scikit-learn si basa una versione ottimizzata di CART, che similmente a C4.5 costruisce alberi in partizioni binarie usando come feature e threshold quella

che porta il più grande valore di information gain per ogni nodo. Tuttavia, benché gli alberi di decisioni siano nati per essere utilizzati su attributi categorici, la versione messa a disposizione da scikit-learn non supporta quest'ultimi e perciò è stato necessario attuare nella fase di preprocessing una conversione di tutti gli attributi categorici a numerici mediante Label Encoder.
I vantaggi del Decision Tree sono:
-Facilmente interpretabile e visualizzabile
-Richiede poca data preparation
-Performante sia su dati categorici e numerici e particolarmente adatto per problemi multy-way
I prinicipali svantaggi sono:
-Alberi troppo complessi potrebbero portare ad overfitting
-Piccole variazioni sui dati possono generare alberi altamente differenti

SCELTA IPERPARAMETRI

Alcuni dei classificatori scelti necessitano di un settaggio per quanto riguarda gli iperparametri e la loro scelta è importante poiché la miglior combinazione di iperparametri determina anche la miglior performance del classificatore stesso. Gli iperparametri sono parametri che non vengono appresi direttamente all'interno dei classificatori ma devono essere forniti in input prima dell'apprendimento. È possibile e consigliato cercare nello spazio iperparametrico il miglior punteggio di cross validation per determinarli..

Per ogni classificatore c'è stata la ricerca della miglior accoppiata di iperparametri in base alle metriche che verranno spiegate in seguito:
Random Forest:
-n_estimator = [10,100,1000] , numeri di alberi nella foresta
- max_features = [sqrt,log2] , numero di feature da considerare per il miglior split
SVC:
-C = [50,10,1.0], parametro regolarizzatore
-kernel = ['poly','rbf'], tipo di kernel scelto
-gamma = [auto], coefficiente per il kernel
Il Decision Tree è un algoritmo non parametrico che non usa iperparametri.

Le miglior combinazioni sono state:
- **Random Forest**: n_estimator = 100 , max_features = sqrt
- **SVC**: C = 1.0 ,kernel = rbf, gamma = auto

Queste combinazioni sono state scelte dopo aver ottenuto le migliori metriche e sfruttando la cross validation, più nello specifico k-fold. Le metriche utilizzate per la valutazione sono le seguenti:

Precision: $\mathrm{TP}/\left(\mathrm{TP}+\mathrm{FP}\right)$ rapporto tra le istanze ritrovate e definite come rilevanti e le istanze ritrovate;
Recall: TP/(TP+FN) rapporto tra le istanze ritrovate e definite come rilevanti e le istanze effettivamente rilevanti;
Accuratezza: $(\mathbf{TP} + \mathbf{TN})\ (\mathbf{TP} + \mathbf{FP} + \mathbf{TN} + \mathbf{FN})$,è il rapporto tra le istanze correttamente predette e il totale delle istanze presenti. Testa l'abilità del modello nel predire correttamente le due classi

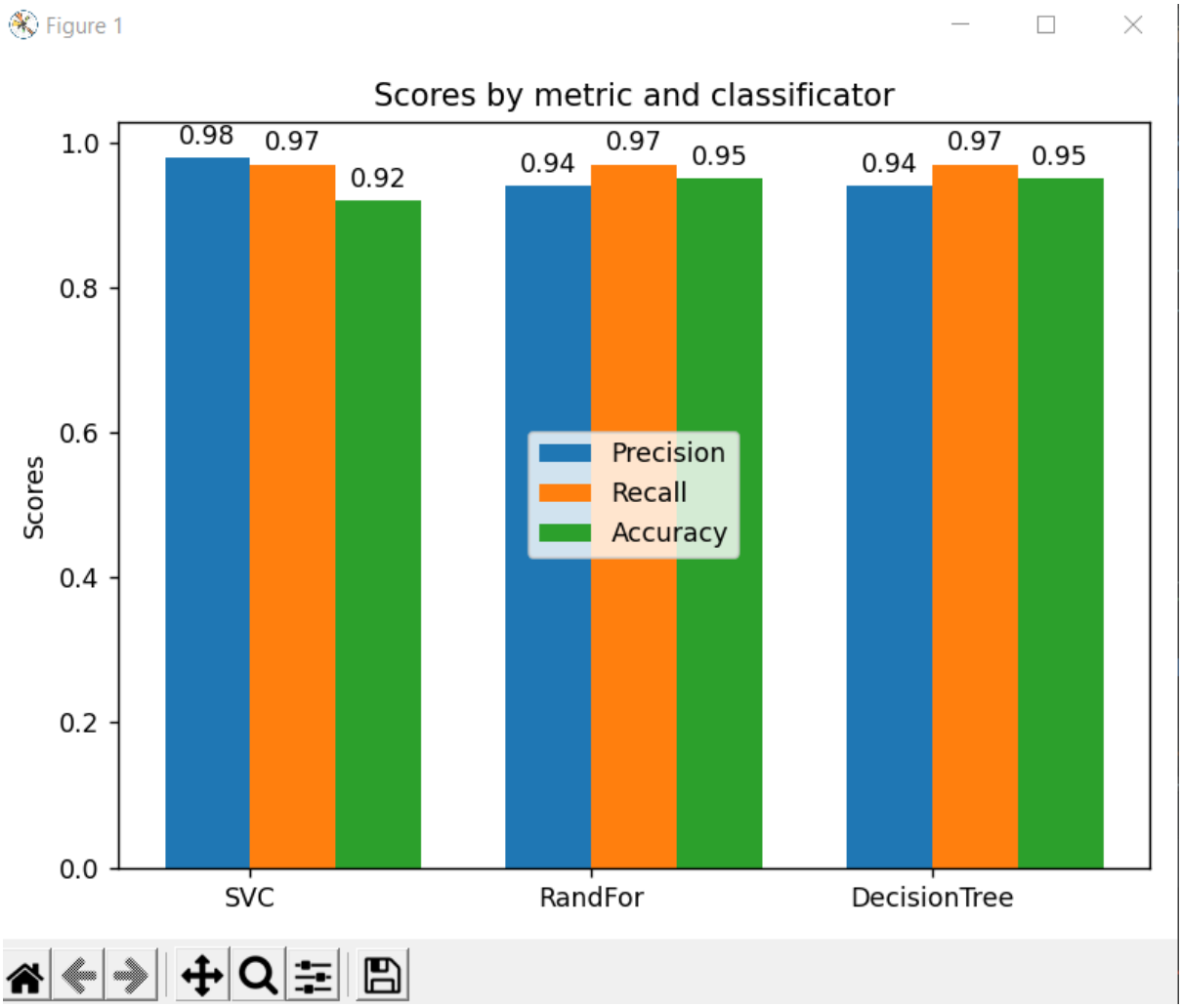
CROSS VALIDATION

Per evitare l'overfitting, ossia l'adattamento eccessivo del modello al training set che in fase di testing non permette al classificatore di generalizzare ed effettuare correttamente la classificazione, è essenziale utilizzare parte dei dati training come surrogato dei dati di test chiamato validation set. La procedura più comune utilizzata è chiamata cross validation. Nell'approccio di base, chiamato k-fold CV, il training set è suddiviso in k insiemi più piccoli. Per ciascuno del k sottoinsiemi si segue la seguente procedura:
1. Un modello viene addestrato utilizzando k-1 sottoinsiemi come dati di allenamento
2. il modello risultante viene convalidato sulla parte restante dei dati
La misura delle prestazioni riportata dalla convalida incrociata k-fold è quindi la media dei valori calcolati nel ciclo. Come scegliere il parametro k dipende dal tempo e dalle risorse disponibili. I valori usuali per k sono 3, 5, 10 o anche N, dove N è la dimensione dei dati.

RISULTATI OTTENUTI

Per scegliere il miglior classificatore più adatto, si sono confrontati i vari classificatori antecedentemente scritti con la loro migliore combinazione di iperparametri, secondo le metriche già descritte. Il tutto è stato fatto sfruttando la Cross Validation.

I risultati emersi sono i seguenti:



Alla luce delle performance molto simile di tutti e tre i classificatori, ho deciso di provarli tutti e tre mediante la fase di test immettendo come input un prodotto non presente nel dataset. È emerso che diversamente dal Decision Tree e Random Forest, la SVC classificava diversi esempi con lo stesso valore target, ovvero “Action Adventure”. La motivazione risiede nel fatto che, i metodi One-vs-One e One-vs-Rest, dediti alla classificazione multi-way, favoriscono la majority class presente nel dataset rispetto alle altre. Pur cercando di penalizzare gli errori fatti dal classificatore su attributi target diversi dalla majority class, quest’ultimo continuava a non riuscire a generalizzare. Di conseguenza la scelta si è ristretta ai classificatori Random Forest e Decision Tree. Alla fine, ho scelto di utilizzare il Decision Tree come classificatore per il classification system, per una mera questione di performance legate al tempo di esecuzione, poiché ambe due i classificatori riescono a generalizzare bene.

Di seguito sono riportati alcuni esempi di utilizzo del sistema:

```
Benvenuto utente

Premere 1 per scegliere il sistema di classificazione , o 2 per scegliere il sistema di raccomandazione
1
Benvenuto nel sistema di classificazione
Inserisci il nome di un titolo che ti è piaciuto ->
FIFA 22
Inserisci l'anno di uscita ->
2021
Inserisci il publisher ->
EA
Inserisci l'intero corrispondete a una caratteristica distintiva del gioco:1 Open-World,2 Baseball,3 Golf,4 Skateboarding,5 Basketball,6 Soccer,7 Football,8 Snowboarding,9 Tennis,1
0
Inserisci l'intero corrispondente alla console di gioco:1 Nintendo64,2 PlayStation,3 Dreamcast,4 Wii,5 GameCube,6 PlayStation2,7 Xbox,8 GameBoyAdvance,9 PlayStation3,10 Xbox360,11
10
Inserisci il voto che daresti a questo gioco ->
8.1
```

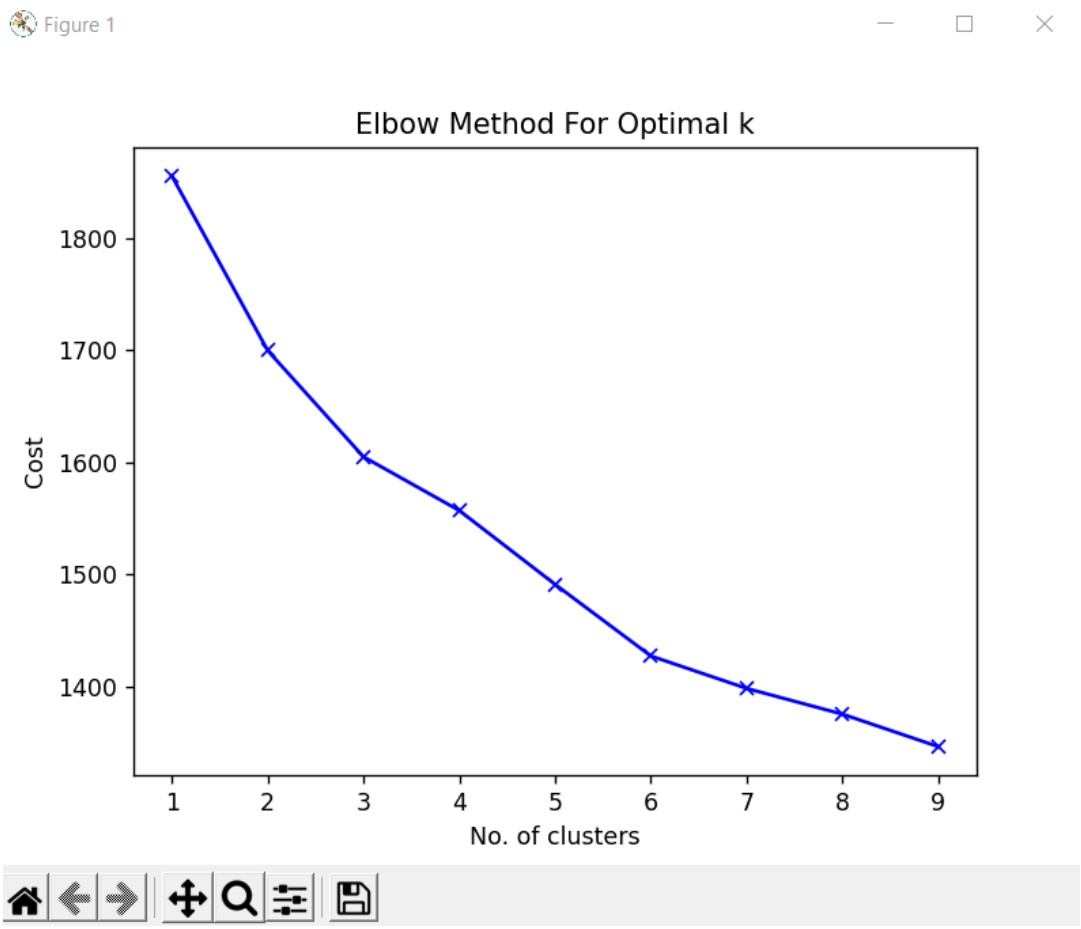
```
Il genere del videogioco da te inserito è Sports


Process finished with exit code 0
```

CLUSTERING E RECOMMENDATION

Il clustering è una metodologia di apprendimento non supervisionato che consente di identificare e raggruppare elementi simili, creando cluster ovvero, gruppi di questi ultimi che risultano conformi ad elementi medi, detti centroidi. Si è scelto di adottare questa tecnica al fine di poter individuare similarità e correlazioni tra i dati che non dipendessero unicamente dall'attributo "genre", in modo da per poterle poi sfruttare per creare recommender system. Poiché l'algoritmo per fare clustering, chiamato KMeans, si basa su distanza Euclidea, metrica non adatta a similarità tra attributi categorici, si è optato per la sua variante KModes. Infatti, quest'ultimo utilizza una misura di similarità per elementi categorici, sostituendo l'utilizzo della media con l'utilizzo della moda, utilizzando un metodo frequencybased utile per minimizzare la funzione di costo. Sono stati individuati sette clusters, e quindi usando "l'elbow method", metodo empirico utile a trovare il numero ottimale di cluster per un set di dati all'interno di un range determinato – nello specifico, il range scelto è rimasto limitato al di sotto del numero di generi dei videogiochi presenti, in modo tale da poter individuare correlazioni non fortemente legate a questi ultimi.

Esempio elbow method:



Per quanto riguarda il funzionamento del sistema di raccomandazione, viene richiesto all'utente di immettere i dati riguardati un videogioco di suo gradimento, similmente a quanto avviene con il sistema di classificazione. Gli attributi immessi vengono utilizzati per trovare una correlazione tra quest'ultimo e i suoi elementi più simili. Per far ciò si creano i cluster, se ne prendono i medoidi, ovvero gli elementi più rappresentativi dei vari clusters, e mediante la libreria FuzzyWuzzy, che utilizza come metrica la distanza di Levenshtein per determinare la similarità tra stringhe, si calcola la similarità tra l'input e i vari medoidi per ogni attributo. Successivamente si ordinano i medoidi in ordine decrescente, si trova il cluster corrispondente al medoide con il più alto valore di similarità, e a sua volta si calcola la similarità tra l'input e ogni elemento del cluster corrispondente al top medoid, infine il cluster viene ordinato in senso decrescente e vengono presi i primi cinque elementi con il valore di similarità più alto.

Esempio funzionamento Recommender System:

```
Benvenuto utente

Premere 1 per scegliere il sistema di classificazione , o 2 per scegliere il sistema di raccomandazione
2
Benvenuto nel sistema di raccomandazione
Inserisci il nome di un titolo che ti è piaciuto ->
FIFA 22
Inserisci l'anno di uscita ->
2021
Inserisci il publisher ->
EA
Inserisci l'intero corrispondente al genere del gioco :1 Action,2 Action Adventure,3 Action RPG,4 RPG,5 JRPG,6 Rhythm,7 Sports,8 FPS,9 TPS,10 Fighting,11 Racing,12 Platformer,13 Pu
7
Inserisci l'intero corrispondete a una caratteristica distintiva del gioco:1 Open-World,2 Baseball,3 Golf,4 Skateboarding,5 Basketball,6 Soccer,7 Football,8 Snowboarding,9 Tennis,1
6
Inserisci l'intero corrispondente alla console di gioco:1 Nintendo64,2 PlayStation,3 Dreamcast,4 Wii,5 GameCube,6 PlayStation2,7 Xbox,8 GameBoyAdvance,9 PlayStation3,10 Xbox360,11
16
Inserisci il voto che daresti a questo gioco ->
8.1
```

```
I videogiochi raccomandati in base alle tue preferenze sono:

FIFA Soccer 10
NHL 2002
Madden NFL 2002
SSX 3
SSX Tricky

Process finished with exit code 0
```