Documentazione progetto Ingegneria della Conoscenza

Collaboratore:

Nome: Alessio
Cognome: Fasano
E-mail: a.fasano32@studenti.uniba.it
Matricola: 738662

Repository github : https://github.com/Afasano32/Games-Succession-Prediction

---

Sommario

1.Introduzione

La crescente popolarità dei videogiochi come forma di intrattenimento e arte interattiva ha portato a un'attenzione sempre maggiore verso la comprensione di ciò che rende un videogioco di successo. Le dinamiche che influenzano il successo di un gioco sono molteplici e complesse, spaziando da aspetti tecnici e creativi a fattori economici e di mercato. Questo progetto si propone di utilizzare modelli di machine learning per analizzare dati storici e predire le probabilità di successo di un videogioco, fornendo così uno strumento di supporto decisionale per sviluppatori e publisher. Il nostro sistema non si limita all'analisi numerica: grazie all'integrazione con ontologie strutturate e una Knowledge Base (KB) dedicata, è in grado di comprendere e rappresentare le relazioni tra elementi come meccaniche di gioco, temi narrativi e preferenze del pubblico. Questa combinazione di machine learning e rappresentazione semantica consente non solo di effettuare previsioni, ma anche di fornire spiegazioni significative sulle correlazioni tra fattori chiave, aiutando gli sviluppatori a prendere decisioni più informate. L'obiettivo principale è quello di colmare il divario tra creatività e dati, supportando il settore dei videogiochi con strumenti innovativi che valorizzano tanto le intuizioni artistiche quanto l'analisi quantitativa.

2.Descrizione del Sistema

Il sistema da noi sviluppato affronta il problema della predizione del successo dei videogiochi attraverso un approccio di regressione, consentendo di stimare in modo quantitativo il livello di successo atteso per un titolo. Le principali funzionalità offerte dal sistema sono:

- **Analisi delle caratteristiche di un videogioco**: il sistema utilizza modelli di machine learning per analizzare dati come genere, piattaforma, players e altri fattori chiave, fornendo una stima continua sul punteggio che gli utenti daranno al videogioco
- **Identificazione di correlazioni significative**: a partire da un insieme di caratteristiche del gioco, il sistema identifica le correlazioni tra i dati, fornendo statistiche utili per comprendere le previsioni.
- **Visualizzazione grafica della conoscenza**: il sistema consente visualizzare i dati relativi a giochi in modo interattivo per comprendere al meglio la natura dei nati.

L'interazione con il sistema avviene tramite un'interfaccia grafica semplice e intuitiva, che permette agli utenti di eseguire previsioni, esplorare i fattori chiave e consultare la Knowledge Base per ricevere nuove informazioni.
Alla base del sistema vi è una Knowledge Base che rappresenta le relazioni tra le caratteristiche dei videogiochi e i risultati di successo. Grazie a questa struttura, gli utenti possono eseguire query per ottenere risposte dettagliate, come ad esempio quali piattaforme sono più in voga tra gli utenti che hanno votato i loro giochi o come ottimizzare una specifica combinazione di fattori.
Il modello di regressione utilizzato è un Ensemble addestrato su un dataset composito di dati storici e recenti, garantendo una stima precisa e aggiornata. Questo approccio permette di trasformare dati grezzi in previsioni utilizzabili, supportando decisioni strategiche nello sviluppo e nel lancio di nuovi titoli.

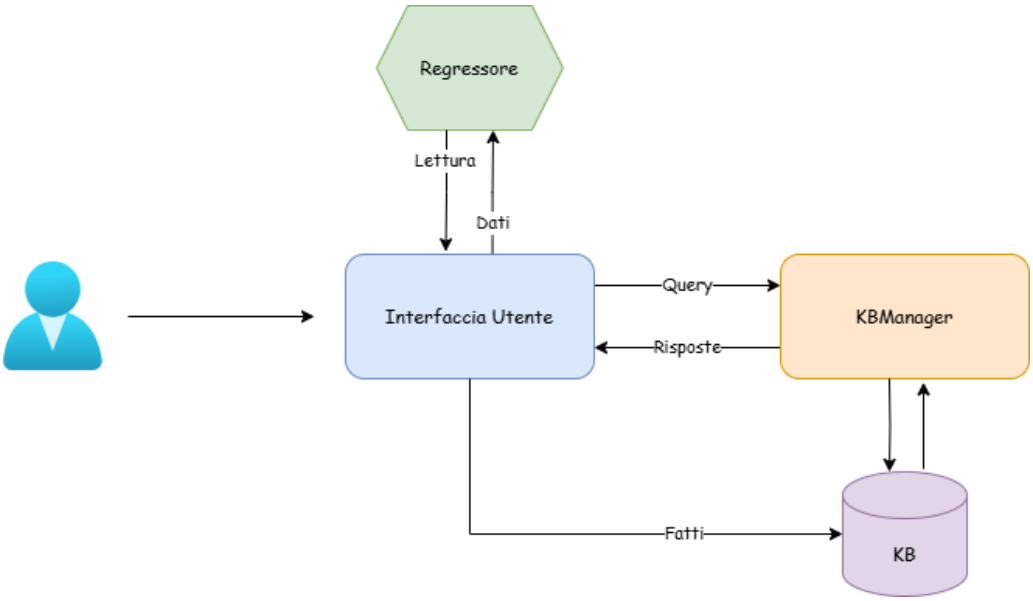


Figura1. Rappresentazione del Sistema

3. Base di conoscenza

Una base di conoscenza è costituita da un insieme di formule espresse mediante un linguaggio di rappresentazione della conoscenza e ammette come formule determinate proposizioni.
Le proposizioni sono frasi che hanno un valore di verità in un mondo, il linguaggio utilizzato per rappresentare la conoscenza è il linguaggio delle clausole definite proposizionali, che è un sotto-linguaggio del calcolo proposizionale, avente la seguente sintassi:

- Un **atomo** o proposizione elementare è costituito da un singolo simbolo che rappresenta una proposizione che può essere vera o falsa.
- Un **corpo** è una congiunzione di più atomi.
- Una **clausola definita** è un atomo (in questo caso chiamata clausola atomica) o è espressa nella forma di regola:

  h :- a1, ..., an.

dove h è la testa della clausola ed è un atomo, mentre a1, ..., an rappresentano il corpo costituito da una congiunzione di atomi.
Il dominio rappresentato è quello dei videogiochi, che include aspetti quali generi, piattaforme, giocatori e altre caratteristiche che influenzano il successo di un titolo. Il dominio è rappresentato in termini di individui e relazioni. Gli individui sono entità del mondo dei videogiochi, come giochi specifici, generi, meccaniche di gioco o piattaforme. Le relazioni specificano cosa è vero riguardo a questi individui, ad esempio quali caratteristiche sono associate a un determinato livello di successo.
Per rappresentare queste informazioni è stato utilizzato il linguaggio logico Prolog, un linguaggio di programmazione basato sulla logica del primo ordine, che si esprime con la seguente sintassi:

- Una **variabile** inizia con una lettera maiuscola.
- Una **costante** inizia con una lettera minuscola, un numero o una stringa.
- Un **predicato** inizia con una lettera minuscola ed è utilizzato per descrivere un atomo.
- Un **atomo** è nella forma p oppure p(t1, ..., tn) dove p è un simbolo di predicato e t1, ..., tn sono termini che possono essere costanti o variabili.

3.1 Fatti

Gli individui nella base di conoscenza in Prolog sono rappresentati e strutturati nel seguente modo:

Game(game_name, developer, user_score, metacritic, critics, playtime, achievements_count, released_year, released_month, released_day, is_in_top_100_publisher, is_in_top_100_developer genre, platform, publisher).

Di seguito viene mostrata una tabella con le costanti associate agli individui rappresentati, insieme al significato dei simboli.

Campo	Significato
game_name	Nome del gioco
developer	Sviluppatore del gioco
user_score	Voto dato al gioco dagli utenti
metacritic	Punteggio Metacritic
critics	Numero di critici che hanno votato il gioco
playtime	Tempo medio di gioco
achievements_count	Numero di achievement disponibili nel gioco
released_year	Anno di rilascio del gioco
released_month	Mese di rilascio del gioco
released_day	Giorno di rilascio del gioco
is_in_top_100_publisher	Indica se l'editore è tra i top 100
is_in_top_100_developer	Indica se lo sviluppatore è tra i top 100
genre	Genere del gioco
platform	Piattaforma supportata
publisher	Editore del gioco

3.2 Regole

Le regole rappresentate mediante simboli di predicato sono di seguito riportate in tabella con i rispettivi significati

Regola	Descrizione
random_select/3	Seleziona N elementi casuali da una lista.
platform_with_most_games/1	Trova le piattaforme con il maggior numero di giochi con User_Score > 8.5.
developer_with_most_games/1	Trova gli sviluppatori con il maggior numero di giochi con User_Score > 8.5.
most_popular_genres/1	Trova i generi più popolari basati su giochi con User_Score > 8.5.
top_rated_games/1	Trova i 5 giochi con il punteggio più alto degli utenti (User_Score > 8.5).

most_played_games/1	Trova i 5 giochi più giocati (con Playtime > 50).
most_achievements_games/1	Trova i 5 giochi con il maggior numero di achievement (Achievements > 50).
most_recent_games/1	Trova i 5 giochi più recenti (con ReleasedYear >= 2020).

Le relazioni definiscono un legame tra un gioco e un particolare attributo, indicando per quali giochi la relazione è vera. Ad esempio, il predicato movement con l'argomento straight — movement(Game, straight) — rappresenta una relazione che è vera quando il tipo di movimento del gioco è rettilineo.

3.3 Interrogare la KB

La base di conoscenza viene utilizzata per consentire all'utente di ottenere informazioni su un gioco, specificando le sue caratteristiche. Per fare ciò, l'utente può fornire una delle caratteristiche di un videogioco, ponendo domande al sistema per ottenere informazioni riguardanti il videogioco e le sue caratteristiche, tramite query con variabili.

Ad esempio, se l'utente vuole conoscere i generi di un gioco, il sistema può eseguire una query utilizzando una delle regole predefinite. Supponiamo che l'utente sia interessato a trovare i generi di giochi che hanno un punteggio utente superiore a 8.5. La query per ottenere i generi più popolari potrebbe essere:

most_popular_genres(X)

Questa query restituisce i generi più popolari in base ai giochi con un punteggio utente maggiore di 8.5. In questo caso, l'istanza della query potrebbe restituire una lista come **['Action', 'Adventure', 'RPG'].**

L'utente non interagisce direttamente con la base di conoscenza, ma lo fa tramite un'interfaccia grafica. Il sistema, tramite il modulo **kb_manager**, gestisce le interazioni e traduce le richieste dell'utente in query verso la base di conoscenza. Le funzioni del modulo eseguono query come **platform_with_most_games(), developer_with_most_games(),** o **most_popular_genres(),** restituendo i risultati filtrati in base ai criteri definiti.

4. Sperimentazione

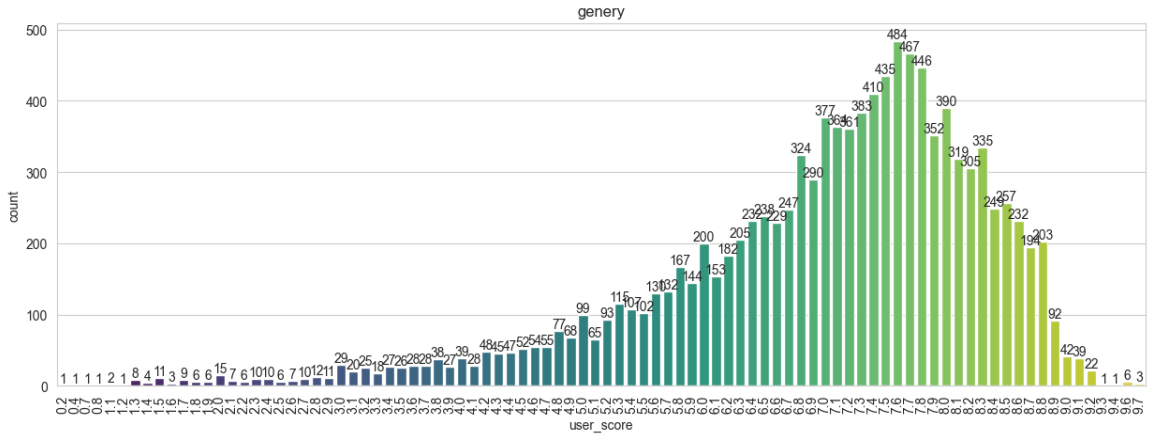
Apprendimento Supervisionato

Nell’apprendimento supervisionato dato un insieme di esempi, coppie input + output, si deve predire l'output per nuovi casi di cui si conosce solo l’input.
I dati consistono in un insieme di feature di input, insieme di feature obbiettivo (target), e gli esempi devono essere suddivisi in training e test con lo scopo di imparare (creare quindi un modello) a predire i valori delle feature obiettivo degli esempi di test, se le feature obiettivo sono valori discreti, il task è di classificazione; se le feature obiettivo sono valori continui, il task è di regressione.
lo schema tipo degli algoritmi supervisionati è quella di definire la rappresentazione compatta di una funzione utile a fare predizioni dai valori delle feature, ottenendo un valore di output Y.

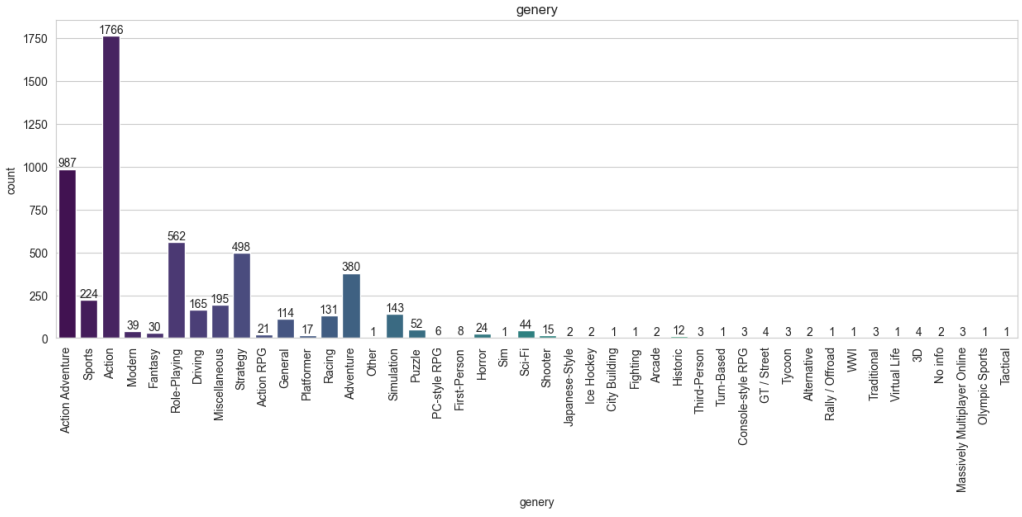
4.1 Dataset

Il dataset utilizzato per l'addestramento e il test del regressore è composto da una raccolta di caratteristiche legate all'entità videogioco.
È composto da due dataset uniti insieme, sono stati analizzati i dati raggruppati, eliminate variabili che avevano una percentuale di dati nulli superiore al 90%.
È stata effettuata un'opportuna procedura di preprocessing sui dati (gestione dei valori mancanti, del tipo di valore continuo o discreto e la normalizzazione dei valori)
Il dataset contiene in tutto 11150 esempi che è stato diviso in set di training e un set di test, precisamente 70% per il training e 30% per il test.

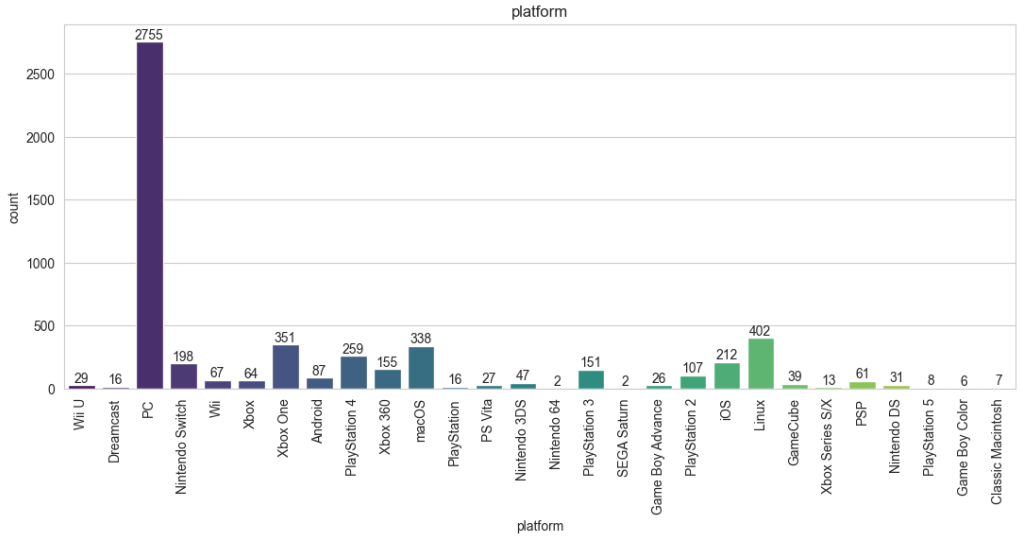
Distribuzione della variabile Target: User_score



Variabile categorica: Genere del videogioco



Variabile categorica: Platform del videogioco



4.2 Ensemble Learning

Nell'ensemble learning, si combinano le predizioni di un certo numero di modelli appresi da learner di base, combinando in questo modo Regressori differenti all'interno di un meta Regressore, che offre prestazioni di generalizzazioni migliori rispetto a ciascun singolo regressore.

Nel machine learning l'apprendimento ensemble si divide in tre tecniche fondamentali:

- Bagging: La tecnica del Bagging (Bootstrap Aggregating) mira a creare un insieme di regressori con uguale importanza. Durante la fase di previsione, ciascun modello fornisce una stima del valore target, e l'output finale è ottenuto combinando le previsioni di tutti i regressori. Nel caso della regressione, si utilizza tipicamente la media delle previsioni dei singoli modelli per determinare il risultato complessivo.
- Boosting: A differenza del bagging, ciascun regressore influisce sulla previsione finale con un certo peso. Tale peso è calcolato in base all'errore commesso dal modello durante la fase di apprendimento, dando maggiore importanza ai modelli che producono previsioni più accurate.
- Stacking: Mentre nel bagging l'output è il risultato di una combinazione delle previsioni (ad esempio, la media), nello stacking viene introdotto un ulteriore regressore (detto meta regressore) che utilizza le predizioni di altri sotto-modelli per effettuare un ulteriore apprendimento e produrre la previsione finale.

4.3 Stacking

La tecnica di Ensemble Learning utilizzata in questo caso è lo Stacking, nel quale si utilizzano un gruppo di learner eterogenei, sono presenti due tipi di learner, Base Learners e un Meta Learner, entrambi i tipi possono essere normali algoritmi di apprendimento automatico come Random Forest, SVM, Perceptron etc. Dato un determinato dataset, i Base Learners vengono addestrati su questo dataset, il Meta Learner viene poi addestrato su un nuovo dataset costruito sulla base delle previsioni dei Base Learner.

L'addestramento mediante tecnica di stacking avviene nel seguente modo:

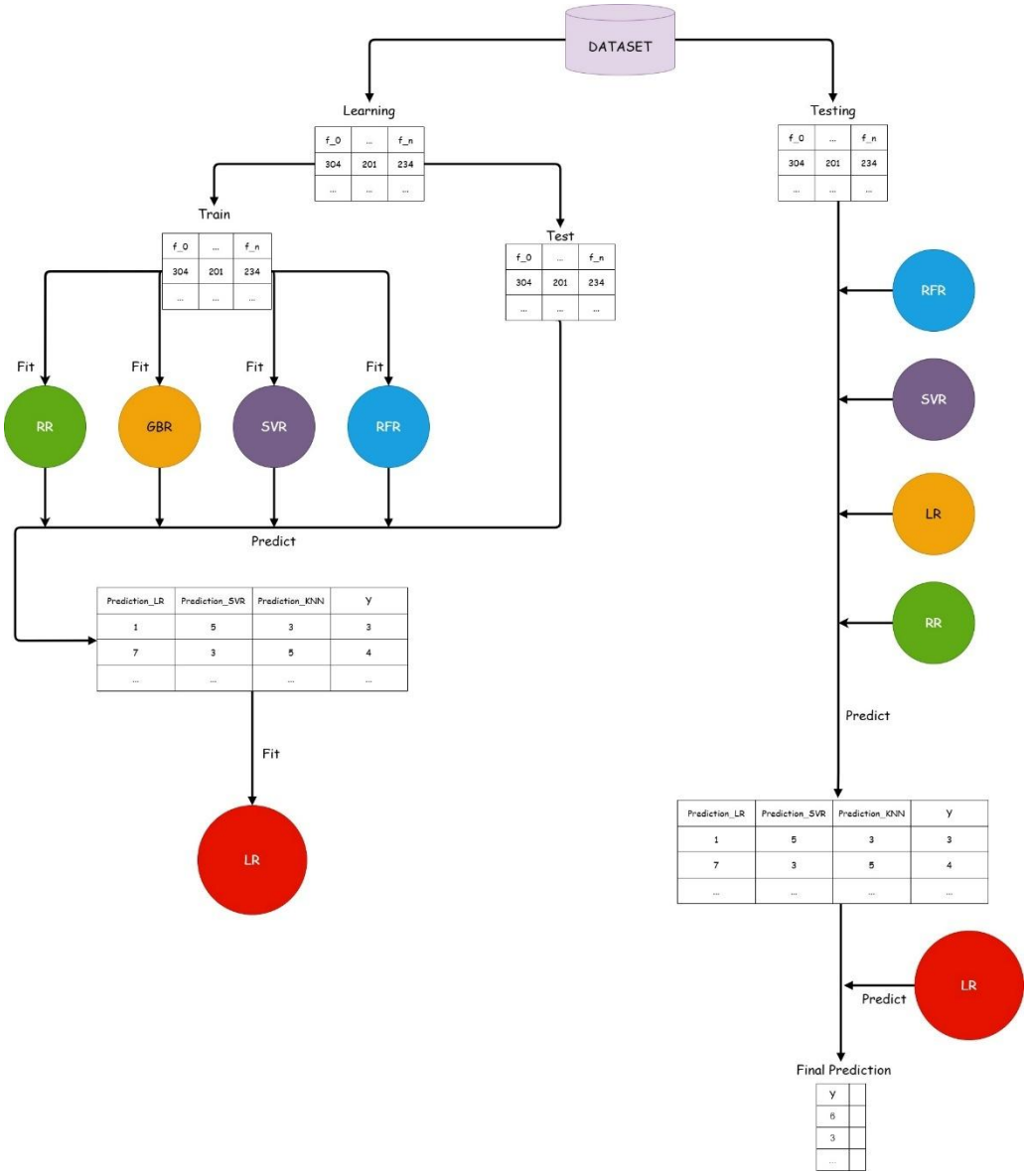
- Il set di addestramento viene diviso in due parti disgiunte
- Ogni base learner viene addestrato sulla prima parte e testato sulla seconda

- Dalle predizioni ottenute dai base learner si crea un nuovo dataset, in cui le meta-feature sono le predizioni di ogni base learner e la feature target è il valore reale dell'input.

Utilizzando questa tecnica si ottengono predizioni più accurate, rispetto all'utilizzo dei singoli modelli.
Una veloce selezione di diversi modelli di regressione ha prodotto queste considerazioni:
scartare modelli inadatti ai dati utilizzati e che aggiungevano poca diversità
- Linear SVR, Ada Boost, Lasso, Elastic Net, KNN diversità.

Si è scelto di utilizzare come base learners i seguenti modelli:
- Random Forest, LGBM, e Gradient Boosting sono stati i più performanti.

E si sono scelti i modleli Ridge Regression e SVR per diversificare l'approcio
Come meta-learner invece si è scelta una rete neurale Multilayer Perceptron Regression(MLPR)



Per ogni base learner è stata effettuata una fase di Tuning e validazione per poter scegliere gli iperparametri, La procedura è stata effettuata utilizzando la tecnica del k-fold con k = 10.
Sono stati salvati i dati relativi al Tuning e validazione effettuate con i modelli che hanno ottenuto i risultati migliori.

5. (Base Learners) Regressori

5.1 Random Forest Regressor

Il **Random Forest Regressor** è un modello di ensemble basato su alberi decisionali. Combina più alberi decisionali, ciascuno addestrato su un sottoinsieme casuale dei dati di training, per produrre una previsione che è la media delle previsioni dei singoli alberi. Questo approccio riduce il rischio di overfitting e migliora la generalizzazione del modello.
Il Random Forest Regressor è particolarmente utile per gestire dati con relazioni non lineari e interazioni complesse tra le feature.

Gli iperparametri principali del Random Forest Regressor includono:

- **n_estimators**: Numero di alberi nel forest.
- **max_depth**: Profondità massima di ciascun albero.
- **min_samples_split**: Numero minimo di campioni richiesti per dividere un nodo.
- **min_samples_leaf**: Numero minimo di campioni richiesti in una foglia.
- **max_features**: Numero di feature considerate per la divisione di un nodo.

Nel nostro caso, gli iperparametri sono stati:

```
Ricerca degli iperparametri ottimali...
Migliori iperparametri trovati: {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 100, 'random_state': 0}
Valutazione del modello con i migliori iperparametri...
MSE: 1.12 (+/- 0.06)
MAE: 0.77 (+/- 0.02)
R2: 0.40 (+/- 0.05)
Migliori iperparametri per Gradient Boosting Regressor: {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 100, 'random_state': 0}
Iperparametri salvati in ../models/iperparametri5/gradient_boosting_params.pkl
--------------------------------------------------
```

5.2 Gradient Boosting Regressor

Il **Gradient Boosting Regressor** è un modello di ensemble che costruisce sequenzialmente alberi decisionali, dove ogni albero corregge gli errori residui del precedente. Questo approccio iterativo permette al modello di apprendere relazioni complesse e non lineari nei dati.
Il Gradient Boosting è noto per la sua alta accuratezza, ma può essere computazionalmente costoso e soggetto a overfitting se non regolarizzato correttamente.
Gli iperparametri principali del Gradient Boosting Regressor includono:

- **n_estimators**: Numero di alberi.
- **learning_rate**: Tasso di apprendimento.
- **max_depth**: Profondità massima di un albero.
- **min_samples_split**: Numero minimo di campioni richiesti per dividere un nodo.
- **min_samples_leaf**: Numero minimo di campioni richiesti in una foglia.

Nel nostro caso, gli iperparametri sono:

```
Ricerca degli iperparametri ottimali...
Migliori iperparametri trovati: {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 100, 'random_state': 0}
Valutazione del modello con i migliori iperparametri...
MSE: 1.12 (+/- 0.06)
MAE: 0.77 (+/- 0.02)
R2: 0.40 (+/- 0.05)
Migliori iperparametri per Gradient Boosting Regressor: {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 100, 'random_state': 0}
Iperparametri salvati in ../models/iperparametri5/gradient_boosting_params.pkl
--------------------------------------------------
```

5.3 Ridge Regressor

Il **Ridge Regressor** è una variante della regressione lineare che introduce una penalizzazione L2 sui coefficienti del modello. Questa penalizzazione riduce l'overfitting, specialmente quando ci sono molti feature correlate.
Il Ridge Regressor è particolarmente utile quando i dati presentano multicollinearità o quando il numero di feature è elevato rispetto al numero di osservazioni.

Gli iperparametri principali del Ridge Regressor includono:

- **alpha**: Parametro di regolarizzazione che controlla l'intensità della penalizzazione L2.
- **fit_intercept**: Se includere o meno un termine di intercetta nel modello.

Nel nostro caso, gli iperparametri sono:

```
Ricerca degli iperparametri ottimali...
Migliori iperparametri trovati: {'alpha': 0.1, 'random_state': 0}
Valutazione del modello con i migliori iperparametri...
MSE: 1.50 (+/- 0.08)
MAE: 0.92 (+/- 0.03)
R2: 0.20 (+/- 0.04)
Migliori iperparametri per Ridge Regressor: {'alpha': 0.1, 'random_state': 0}
Iperparametri salvati in ../models/iperparametri5/ridge_params.pkl
--------------------------------------------------
```

5.4 Support Vector Regression (SVR)

La **Support Vector Regression (SVR)** è una variante delle Support Vector Machines (SVM) utilizzata per problemi di regressione. A differenza della regressione lineare, la SVR cerca di trovare una funzione che approssimi i dati di training mantenendo un margine di tolleranza ϵ entro il quale gli errori non vengono penalizzati. Questo approccio è particolarmente utile quando i dati presentano relazioni non lineari.

La SVR utilizza una funzione kernel per mappare i dati in uno spazio di dimensione superiore, dove è possibile trovare un iperpiano di regressione. I kernel più comuni includono:

- **Lineare**: (K(x, x') = x^T x')
- **Polinomiale**: (K(x, x') = (x^T x' + r)^d)
- **Radial Basis Function** (RBF): (K(x, x') = (-|x - x'|^2))

Gli iperparametri principali della SVR sono:

- **kernel**: Tipo di kernel da utilizzare (es. 'rbf', 'poly', 'linear').
- **C**: Parametro di regolarizzazione che controlla il trade-off tra la minimizzazione dell'errore e la massimizzazione del margine.
- **gamma**: Coefficiente del kernel (soprattutto per kernel RBF e polinomiale).
- **degree**: Grado del kernel polinomiale. - epsilon: Margine di tolleranza entro il quale gli errori non vengono penalizzati.

Nel nostro caso, gli iperparametri scelti sono:

```
Ricerca degli iperparametri ottimali...
Migliori iperparametri trovati: {'C': 10, 'degree': 2, 'gamma': 'scale', 'kernel': 'rbf'}
Valutazione del modello con i migliori iperparametri...
MSE: 1.30 (+/- 0.08)
MAE: 0.82 (+/- 0.03)
R2: 0.30 (+/- 0.04)
Migliori iperparametri per Support Vector Regressor: {'C': 10, 'degree': 2, 'gamma': 'scale', 'kernel': 'rbf'}
Iperparametri salvati in ../models/iperparametri5/svr_params.pkl
--------------------------------------------------
```

6. (Meta Learners) Regressori

6.5 Linear Regressor

La **Regressione Lineare** è un modello statistico che cerca di modellare la relazione tra una variabile dipendente (target) e una o più variabili indipendenti (features) assumendo una relazione lineare tra di esse. L'obiettivo è trovare la linea retta (o iperpiano in più dimensioni) che minimizza la somma dei quadrati delle differenze tra i valori osservati e quelli predetti.

La regressione lineare è espressa dalla formula:

$$y=\beta 0+\beta 1x1+\beta 2x2+\cdots+\beta nxn+\epsilon$$

dove:

- y è la variabile dipendente,
- β0β0 è l'intercetta,
- β1,β2,...,βnβ1,β2,...,βn sono i coefficienti delle variabili indipendenti,
- x1,x2,...,xnx1,x2,...,xn sono le variabili indipendenti,
- ϵϵ è l'errore residuo.

La regressione lineare non richiede la specifica di iperparametri, poiché il modello è determinato direttamente dai dati di training attraverso la minimizzazione della funzione di costo (solitamente l'errore quadratico medio).

6.6 Multilayer Perceptron Regressor (MLPR)

Il **Multilayer Perceptron Regressor (MLPR)** è un modello di rete neurale artificiale utilizzato per problemi di regressione. È composto da più strati di neuroni, ciascuno dei quali è completamente connesso al successivo. Ogni neurone applica una funzione di attivazione non lineare ai dati in ingresso, permettendo al modello di apprendere relazioni complesse e non lineari tra le variabili.

Gli iperparametri principali del MLPR sono:

- **hidden_layer_sizes**: Numero di neuroni in ciascuno strato nascosto.
- **activation**: Funzione di attivazione (es. 'relu', 'tanh', 'logistic').
- **solver**: Algoritmo di ottimizzazione (es. 'adam', 'lbfgs', 'sgd').
- **random_state**: Seed per la riproducibilità dei risultati.

Nel nostro caso, gli iperparametri scelti sono:

{'activation': 'tanh',

'alpha': 0.0001,

'early_stopping': False,

'hidden_layer_sizes': (100,),

'learning_rate_init': 0.001,

'solver': 'adam'}

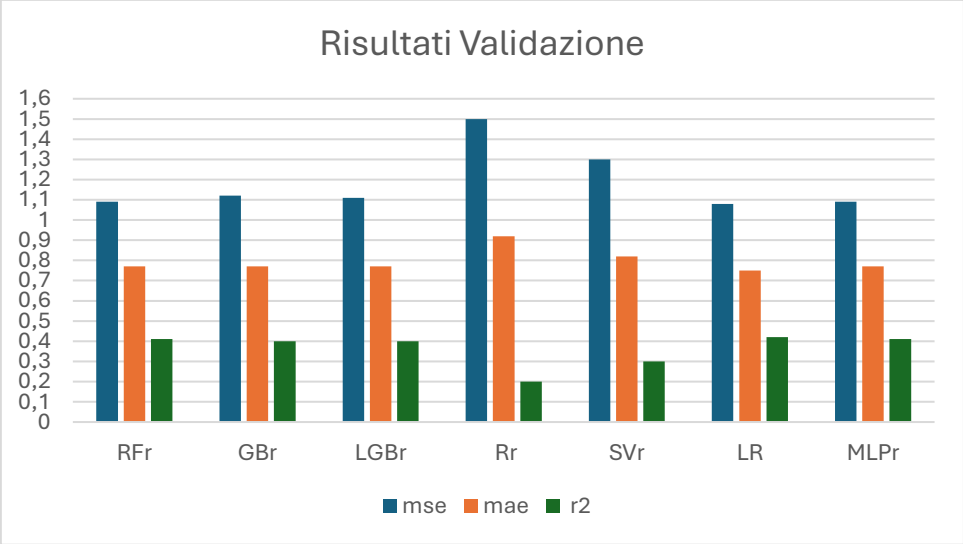
7. Risultati Validazione & Test

La scelta di quali iperparametri utilizzare è stata guidata dai risultati ottenuti in fase di validazione che è stata effettuata per ogni base learners e per il meta-learner.
Dopo aver scelto i risultati migliori sì é eseguito il test finale.
Nell validazione é stata effettuata con il 10-fold dove si sono usati una parte dei dati di training come set di valutazione del modello (validazion set) ottenendo risulati nel seguente formato :

Metrica principale:
Mean Squared Error +/- deviazione standard
Metrica secondaria:
Mean Absolute Error +/- deviazione standard
Coefficiente di determinazione +/- deviazione standard

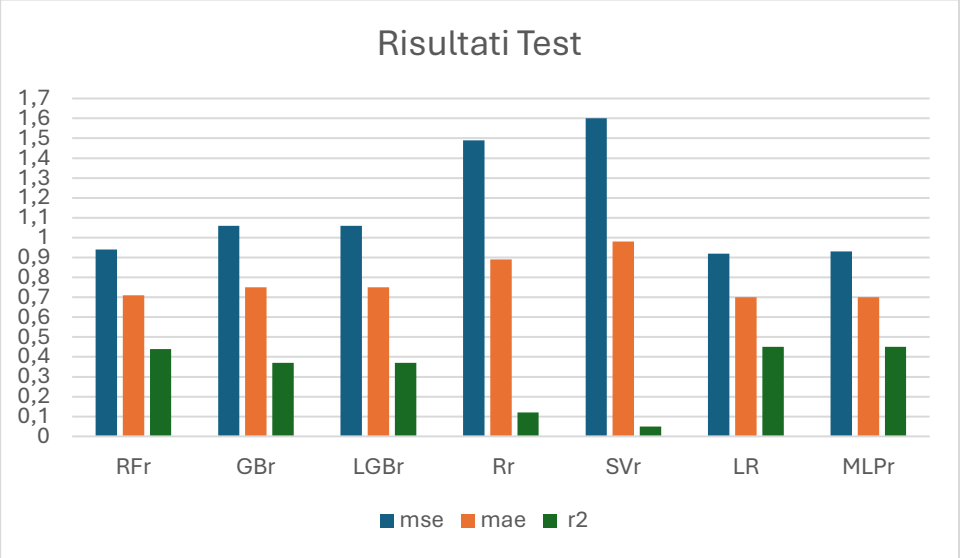
La convalida incorciata con il 10-fold mira a una scelta dei parametri / rappresentazione del modello per cui l'errore sia minimo sul set di validazione, ipotizzando che questo possa valere anche per il test set
I risultati ottenuti nella fase di validazione sono:

Model	Mean Squared Error	Standard Deviation
RFr	1,09	0.06
GBr	1,12	0.06
Rr	1,50	0.08
SVr	1,30	0,08
Stacking_LP	1.08	0.06
Stacking_MLPR	1.09	0.06



Soddisfatti dei risultati ottenuti con il validation set si è eseguito il test finale addestrando ogni modello sull'intero training set e si è eseguita la predizione sul test set mai utilizzato prima in fase di validazione, si sono ottenuti risultati migliori rispetto a quelli ottenuti in fase

di validazione come mostrato nella seguente figura che mostra i valori delle metriche ottenuti per ogni modello e lo stacking



Di seguito sono riportati per ogni modello i valori di Mean Squared Error, Mean Absolute Error e Coefficiente di determinazione ottenuti nel test

Model	MSE	MAE	R2
RFr	90,6%	92,9%	44%
GBr	89,4%	92,5%	37%
LGBMr	89,4%	92,5%	37%
Rr	85,1%	91,1%	12%
SVr	84%	90,2%	5%
Stacking_LR	90,8%	93%	95,5%
Stacking_MLP	90,7%	93%	95,5%

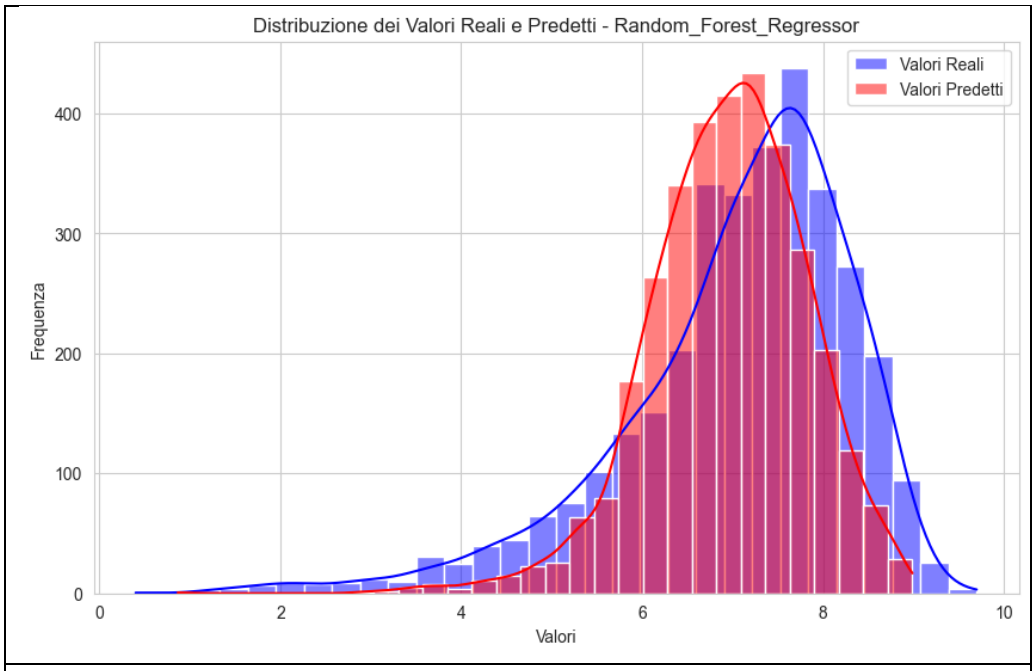
Questi valori rappresentano la performance dei modelli in percentuale, dove un valore più alto indica una migliore performance.

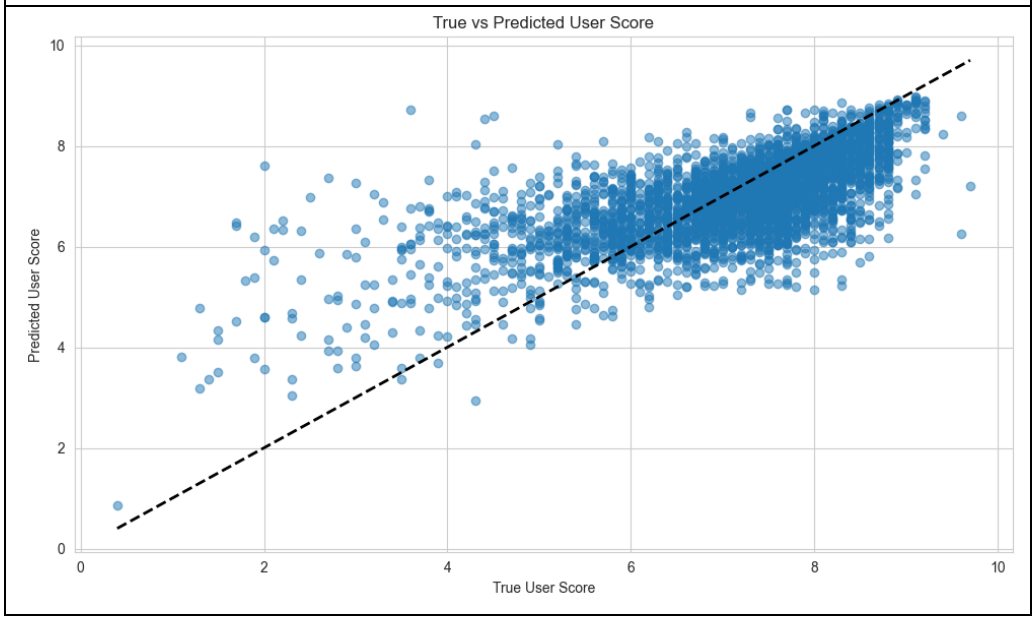
Formula di conversione in percentuale:

$$Valore\ percentuale = (1 - \frac{Valore}{Massimo\ valore\ possibile}) \times 100$$

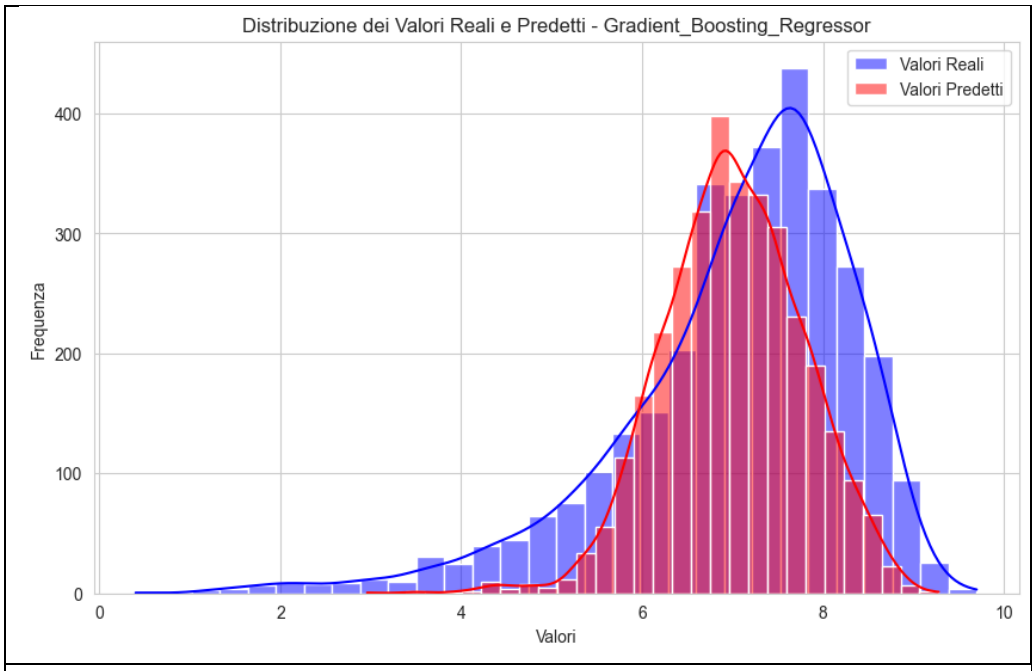
Conclusione della sperimentazione: il modello attualmente utilizzato è quello che ha performato meglio secondo i valori di valutazione considerati (MSE, MAE, R2), ovvero il modello **Stacking_LR**

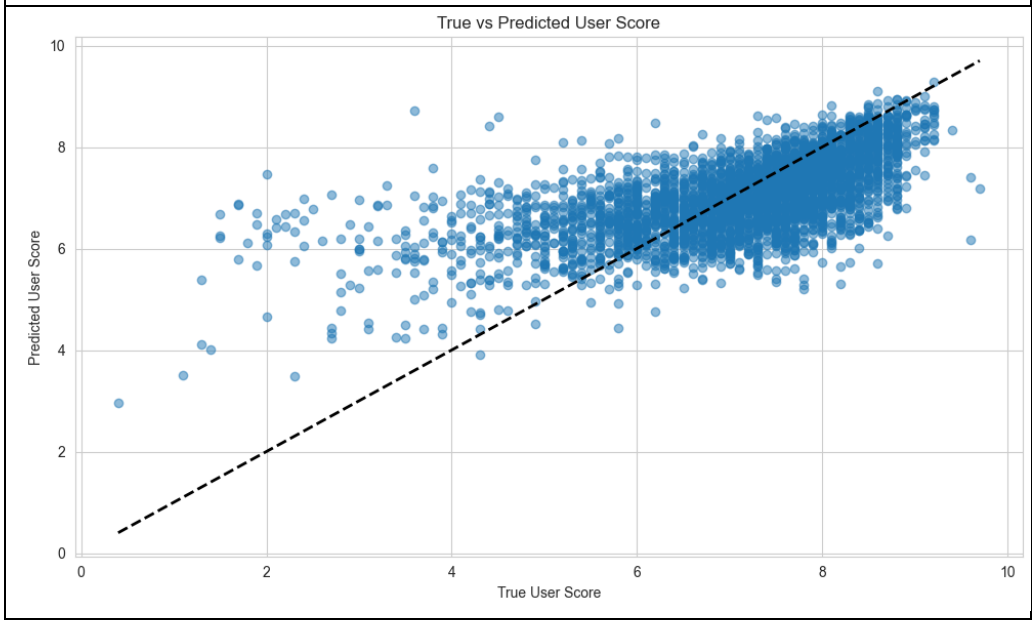
RFr


Distribuzione dei Valori Reali e Predetti - Random_Forest_Regressor
Valori Reali
Valori Predetti
Frequenza
Valori
0
100
200
300
400
0
2
4
6
8
10

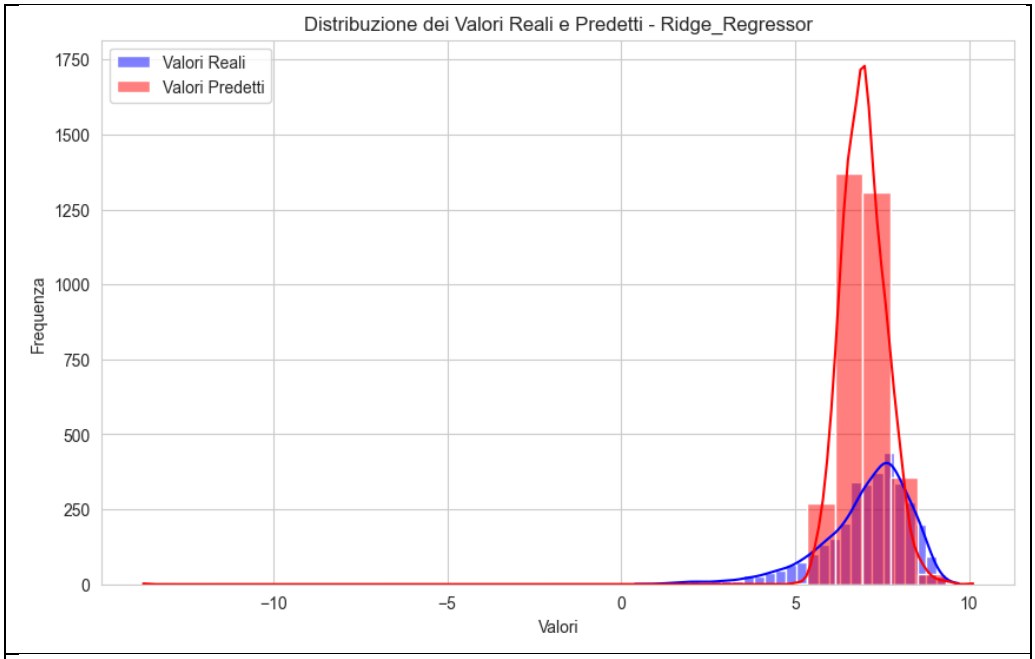

True vs Predicted User Score
Predicted User Score
True User Score
0
2
4
6
8
10

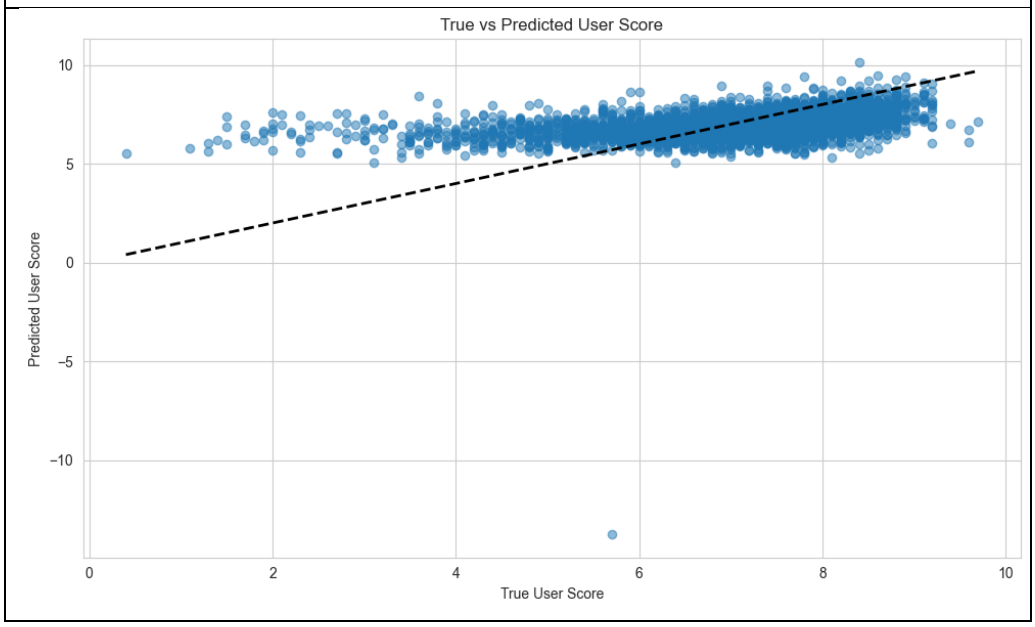
GBr


Distribuzione dei Valori Reali e Predetti - Gradient_Boosting_Regressor
Valori Reali
Valori Predetti
Frequenza
Valori
0
100
200
300
400
0
2
4
6
8
10

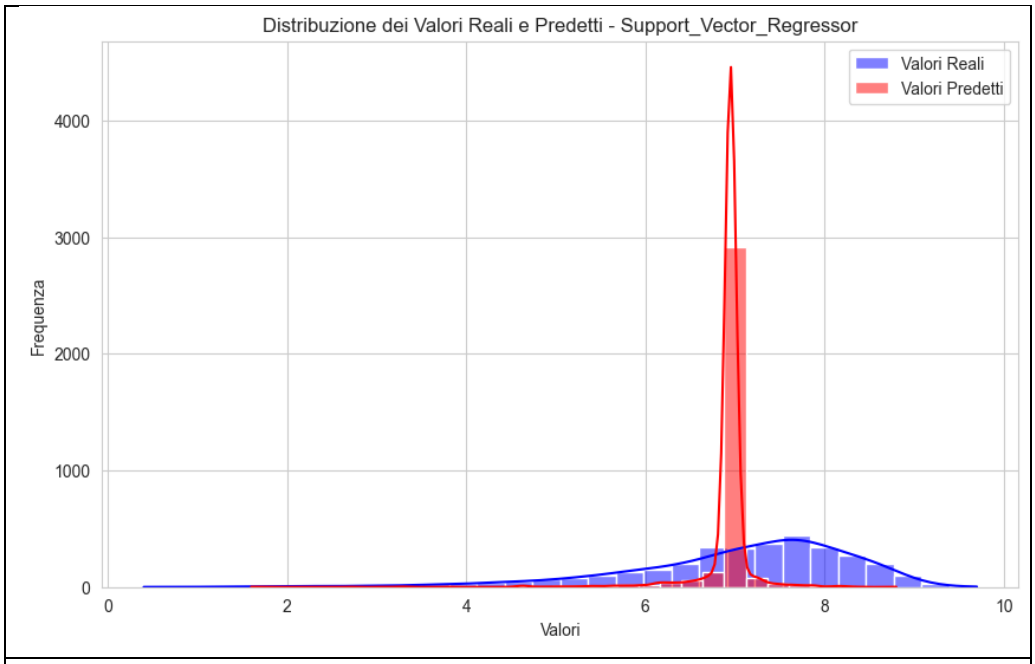

True vs Predicted User Score
Predicted User Score
True User Score
0
2
4
6
8
10

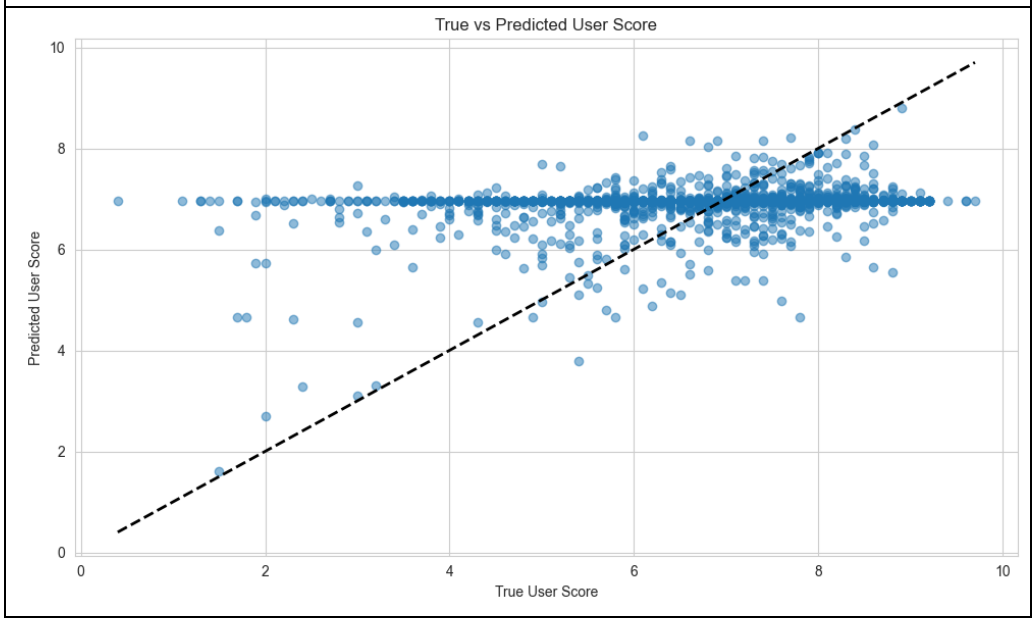
Rr


Distribuzione dei Valori Reali e Predetti - Ridge_Regressor
Valori Reali
Valori Predetti
Frequenza
Valori
1750
1500
1250
1000
750
500
250
0
−10
−5
0
5
10

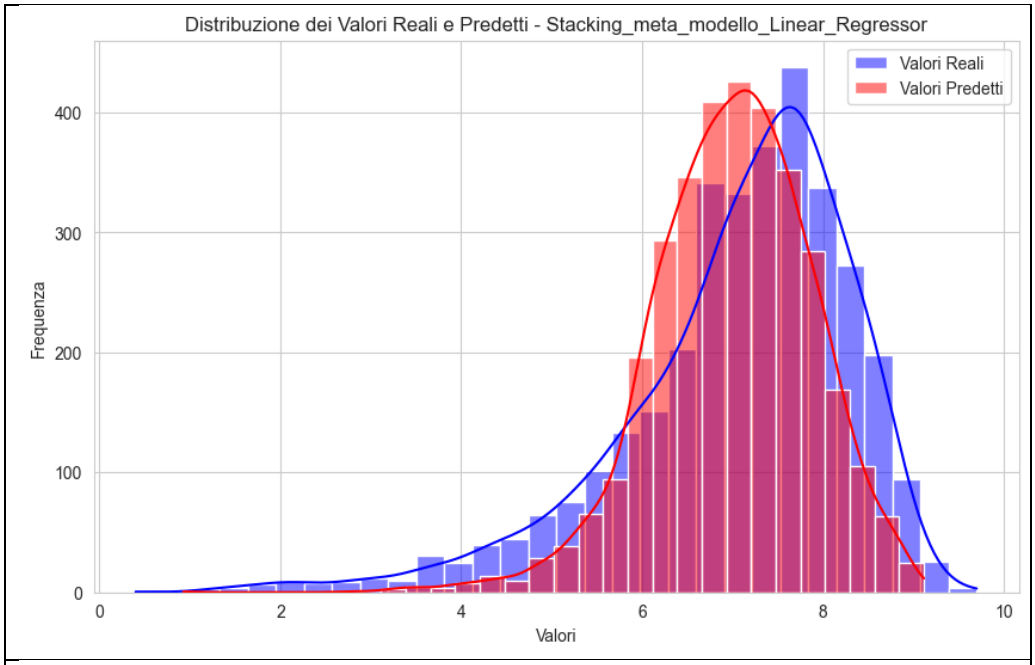

True vs Predicted User Score
Predicted User Score
True User Score
10
5
0
−5
−10
0
2
4
6
8
10

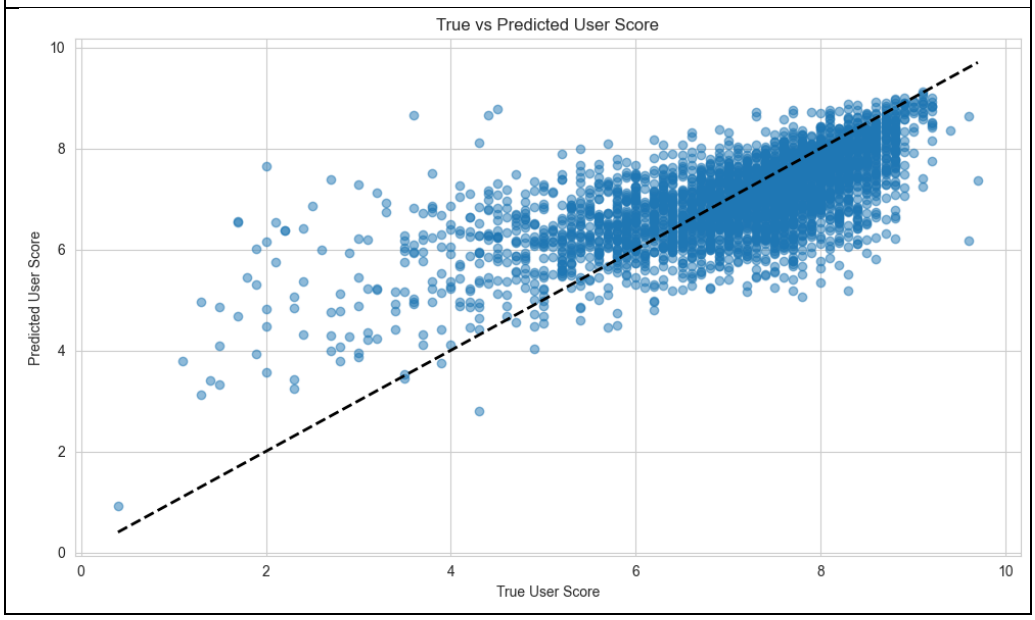
SVr


Distribuzione dei Valori Reali e Predetti - Support_Vector_Regressor
Valori Reali
Valori Predetti
Frequenza
Valori
0
1000
2000
3000
4000
0
2
4
6
8
10

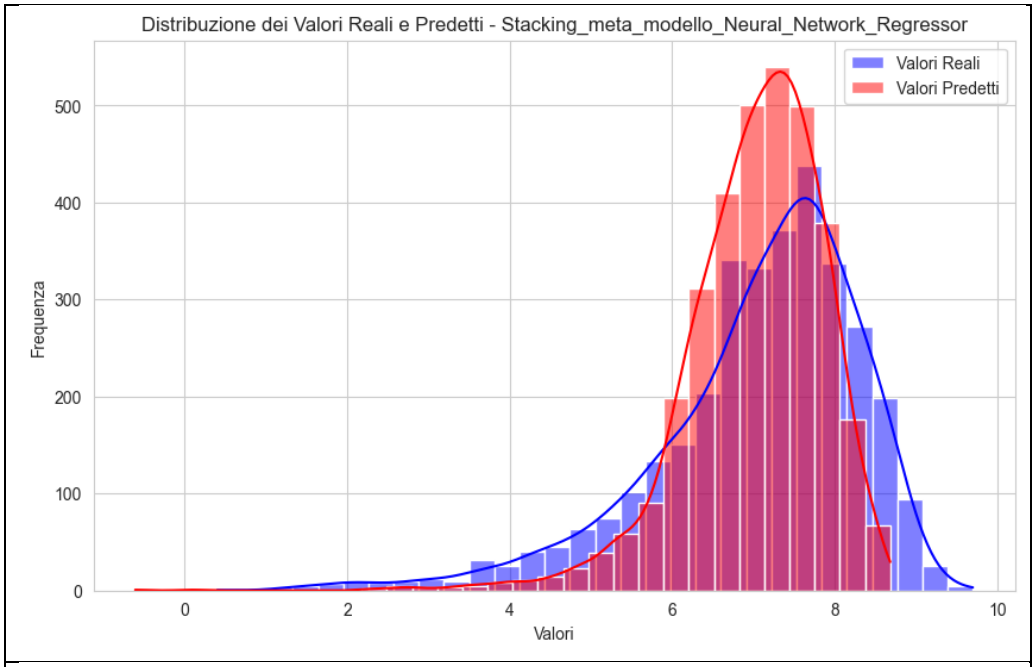

True vs Predicted User Score
Predicted User Score
True User Score
0
2
4
6
8
10

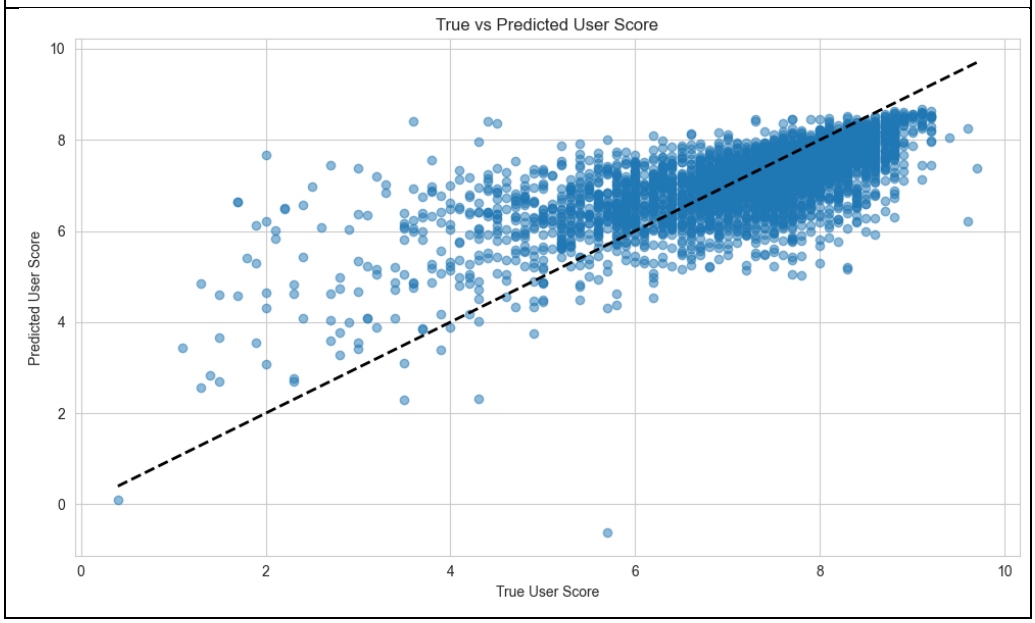
Stacking_LR


Distribuzione dei Valori Reali e Predetti - Stacking_meta_modello_Linear_Regressor
Valori Reali
Valori Predetti
Frequenza
Valori
0
100
200
300
400
0
2
4
6
8
10


True vs Predicted User Score
Predicted User Score
True User Score
0
2
4
6
8
10

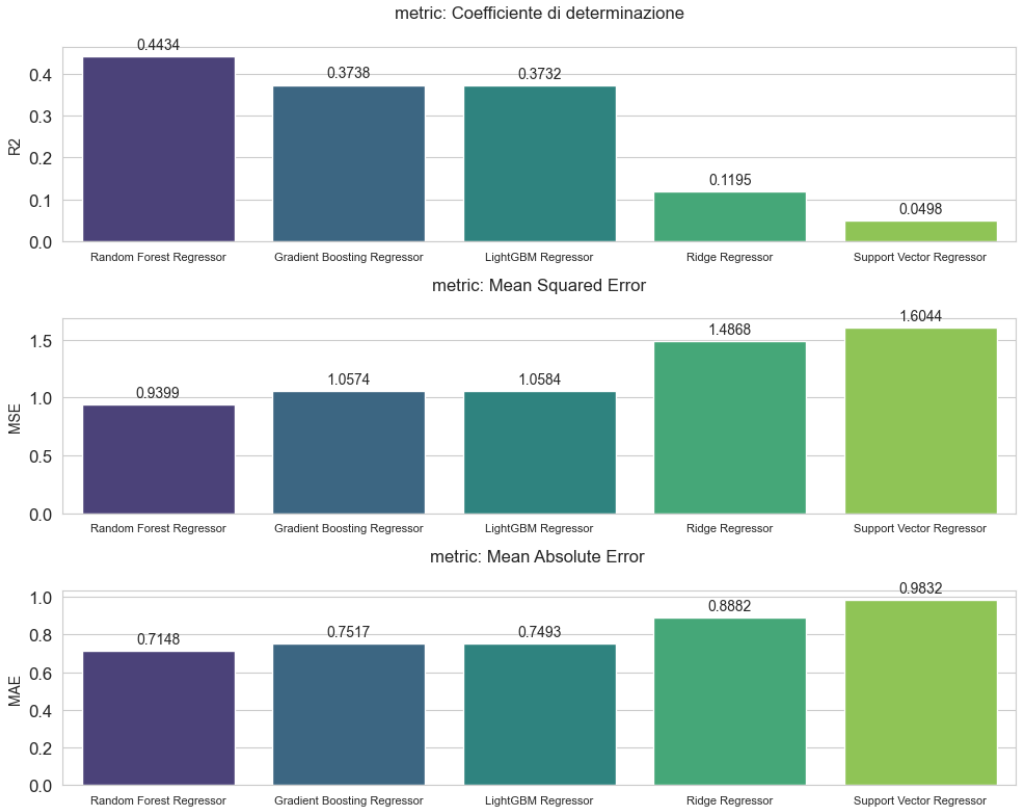
Stacking_MLPR



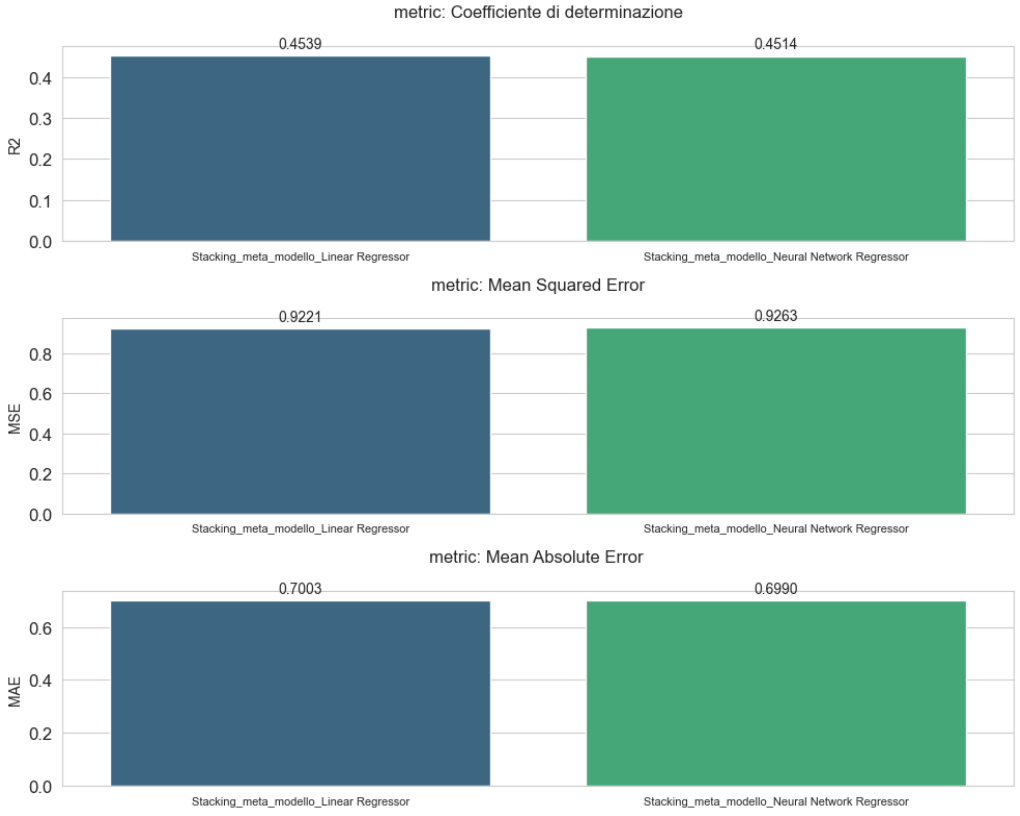


Confronto dei modelli

- Modelli base



- Modelli stacking

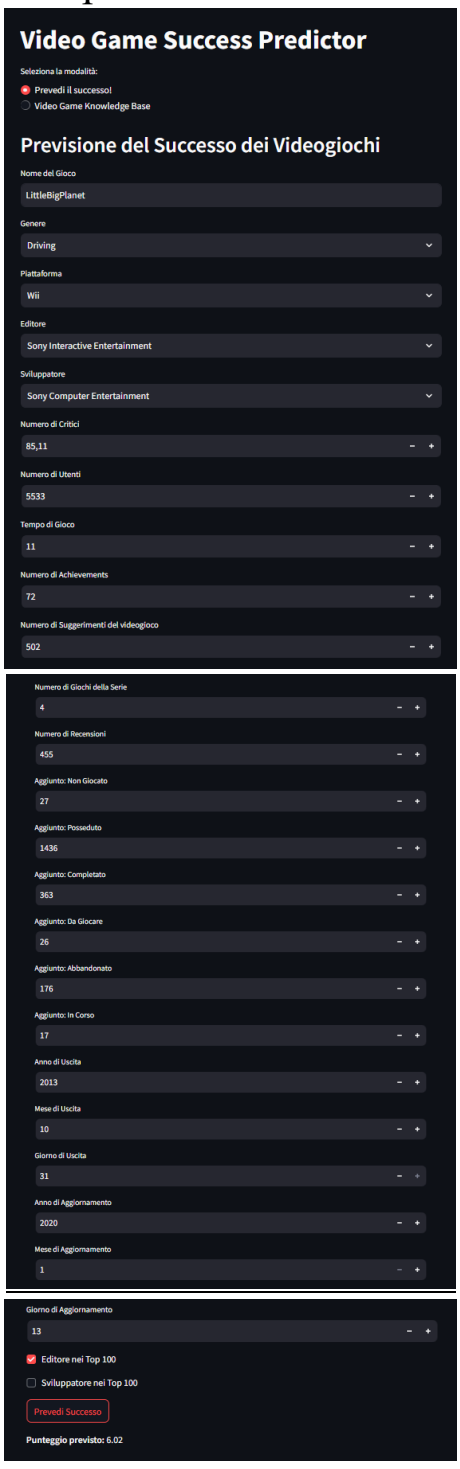


8. Interfaccia

Per aiutare l'utente a interagire al meglio con il nostro sistema è stato creato un prototipo di interfaccia in modo da interagire con il sistema. Una volta avviato il file VideoGameSucess.bat, si aprirà in pagina web sul proprio browser predefinito che visualizzare la home dove si potranno scegliere le due opzioni principali

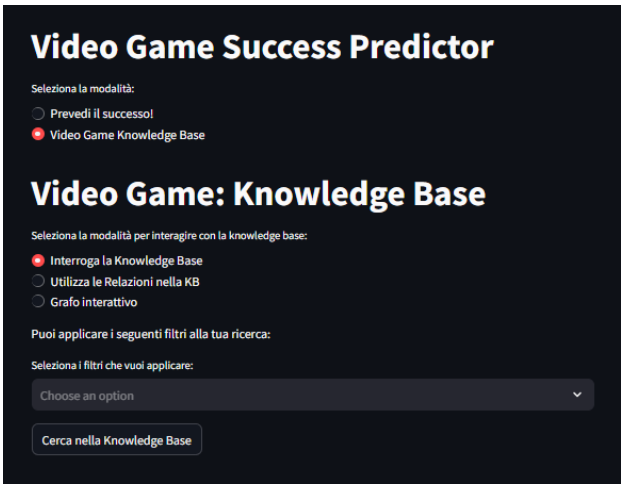
1. predire il successo di un videogioco
2. Interagire con la base di conoscenza utilizzata nel sistema

8.1. Schermata principale:



Nella schermata principale l'utente può da subito scegliere di passare alla seconda modalità o utilizzare la prima per interagire con il modello addestrato a predire il successo del videogioco. Il range del punteggio va da 0 a 10

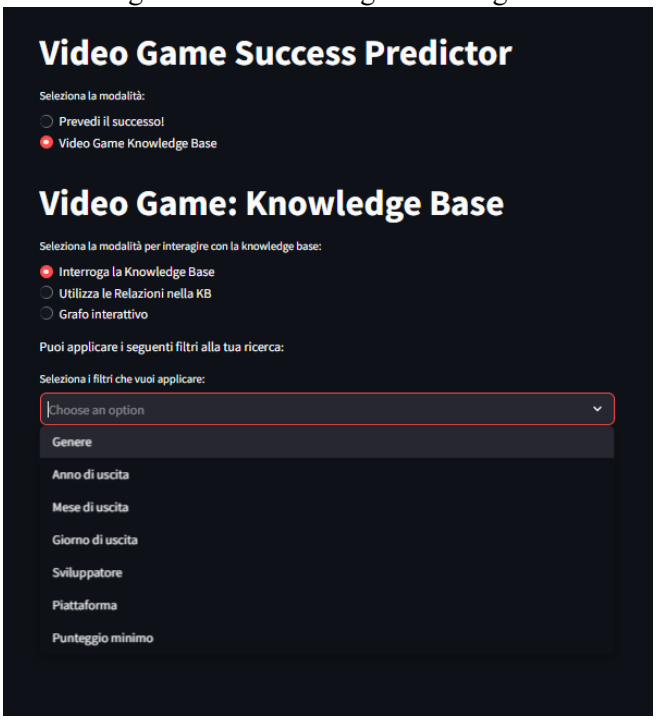
8.2. Schermata Knowledge base:



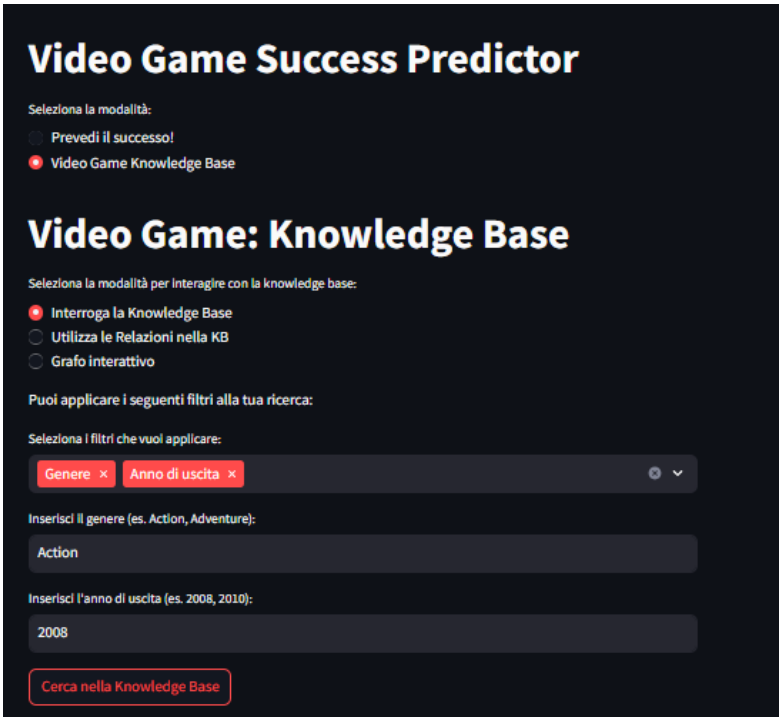
In questa schermata è possibile interagire con la Knowledge base in tre diversi modi

1. Utilizzare una query di base arricchita da filtri che l'utente più aggiungere a piacere;
2. Utilizzare delle query specifiche per scansionare la correlazione tra i dati presenti nella KB
3. Visualizzare tre diversi tipi di grafi che mostrano l'interazione tra l'entità game e i suoi attributi.

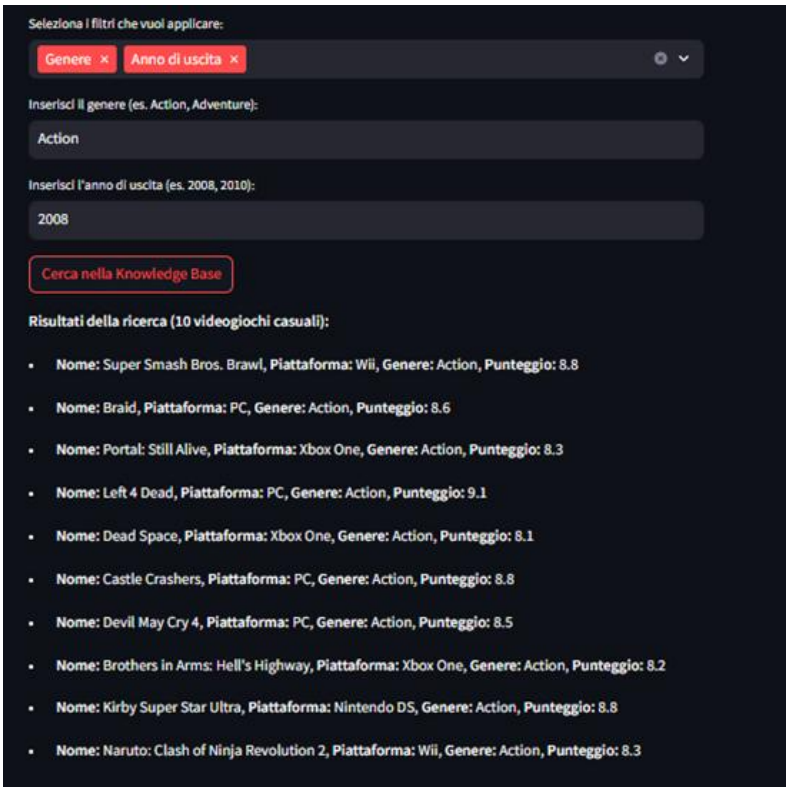
Opzione 1: interagire con la knowledge base scegliendo dei filtri



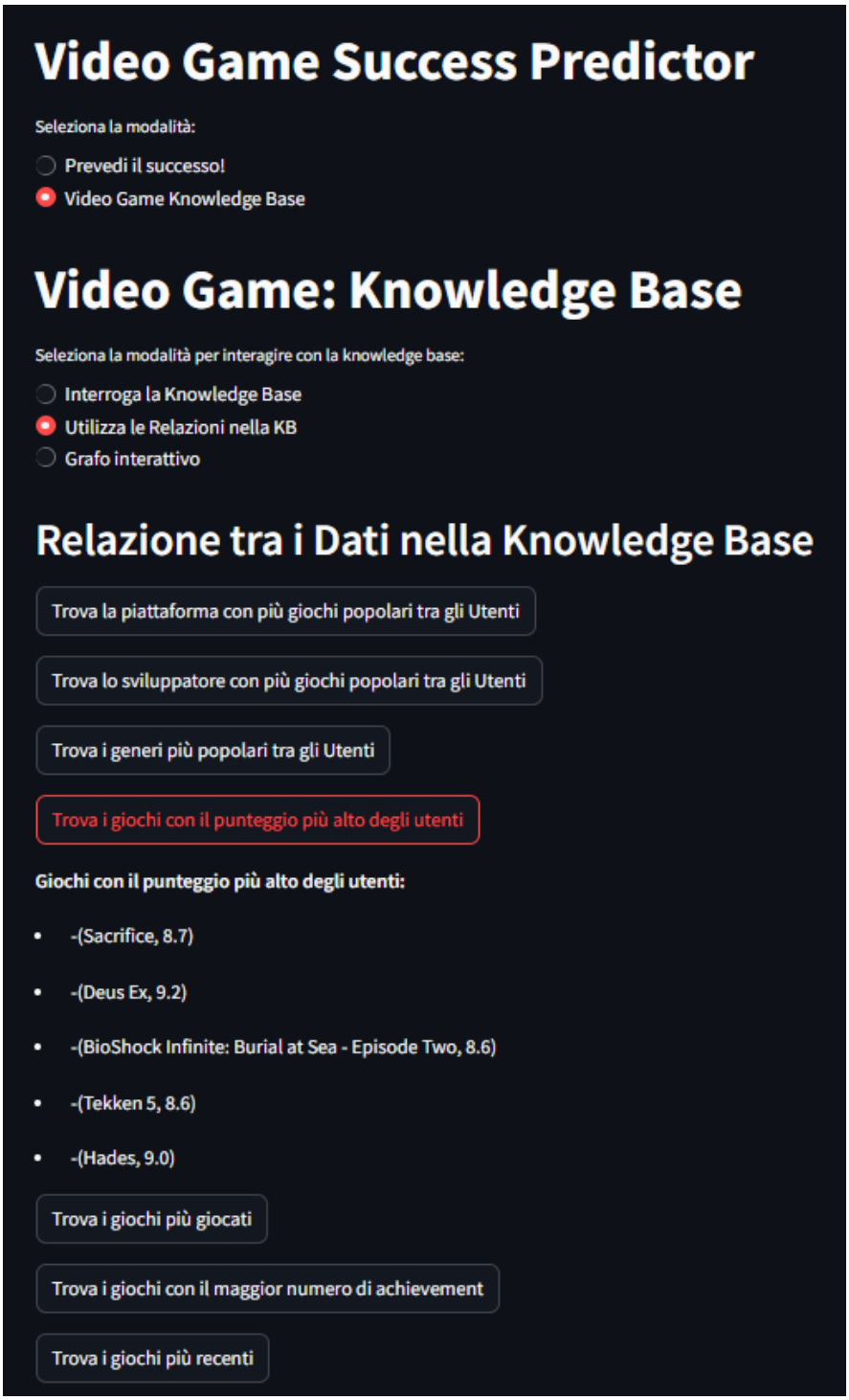
Inserimento dei filtri:



Risultati dei filtri:



opzione 2: utilizza delle regole derivate dalla KB per scansionare
la correlazione tra i dati presenti nella KB



Opzione 3: visualizzare un grafo interattivo per studiare i l comportamento dei valor nella KB

9.Guida dell'installazione

Questa guida ti aiuterà a configurare e avviare il progetto Python sul tuo sistema.

Prerequisiti

Assicurati di avere i seguenti strumenti installati sul tuo sistema:

- **Python 3.8 o superiore**: Scarica Python
- **pip** (gestore di pacchetti Python): Di solito è incluso con l'installazione di Python.
- **Git** (opzionale): Scarica Git

Installazione

1. **Clona il repository (se il progetto è su GitHub)**:

   git clone: Afasano32/Games-Succession-Prediction

2. **Avvio rapido:**
   esegui il file **VideoGameSuccess.bat** (NON eseguire come amministratore).
   Il file contiene uno script che verificherà la presenza di python nel sistema e in caso di mancanza lo installerà, installerà tutte le dipendenze e avvierà l'interfacci web di streamlit per avere accesso al prototipo.

3. **Crea un ambiente virtuale (consigliato)**:

   Creare un ambiente virtuale ti permette di isolare le dipendenze del progetto.

   Comando python: **python -m venv venv**

   Comando - Windows: **venv\Scripts\activate**

   Comando - Linux/macOS: **source venv/bin/activate**

4. **Installa le dipendenze**:

   Usa pip per installare le dipendenze necessarie elencate nel file requirements.txt.

   comando: **pip install -r requirements.txt**

5. **Esegui il progetto**:

Una volta configurato tutto, puoi avviare il progetto con il comando:

comando: **streamlit run app/main.py**

10. Sviluppi futuri

I possibili sviluppi dal punto di vista funzionale potrebbero essere migliorare le possibilità che il sistema offre all'utente in modo tale da esse usato in abito lavorativo come tool di sviluppo creativo per nuovi videogames.
Dal punto di vista progettuale si potrebbe rendere il sistema un eseguibile o una vera e propria app da utilizzare non solo per sviluppatori creativi ma per qualsiasi tipo di utente sia interessato all'ambito videoludico.

Riferimenti Bibliografici

-Dataset utilizzati
https://www.kaggle.com/datasets/brunovr/metacritic-videogames-data
https://www.kaggle.com/datasets/jummyegg/rawg-game-dataset

Ragionamento logico: D. Poole, A. Mackworth: Artificial Intelligence: Foundations of Computational Agents. 3e. Cambridge University Press [Ch.5]

Prolog: D. Poole, A. Mackworth: Artificial Intelligence: Foundations of Computational Agents. 3e, Cambridge University Press [Ch.15]

Apprendimento supervisionato: D. Poole, A. Mackworth: Artificial Intelligence: Foundations of Computational Agents. 3e. Cambridge University Press. [Ch.7]

Modelli Neurali: D. Poole, A. Mackworth: Artificial Intelligence: Foundations of Computational Agents. 3e. Cambridge University Press. [Ch.8]