CASO DI STUDIO
INGEGNERIA DELLA CONOSCENZA

OlympiX

Componenti del gruppo:

Leonardo Montemurro
matricola: 758401
l.montemurro7@studenti.uniba.it

Vincenzo Sarra
matricola: 758455
v.sarra@studenti.uniba.it

Link Github: https://github.com/Montesano22/progettoIcon

AA 2023-2024

Indice

1. Introduzione 3
1.1 Elenco argomenti di interesse 3
1.2 Dataset e tecnologie utilizzate 4
2. Apprendimento Supervisionato 5
2.1 Caricamento e pulizia dei dati 5
2.2 Preprocessing dei dati 5
2.3 Divisione Training set e Test set 6
2.4 Scelta degli iper-parametri 7
2.5 Addestramento e Valutazione dei Modelli 11
3.Apprendimento non Supervisionato 17
3.1 Selezione delle caratteristiche 17
3.2 Processing dei dati 18
3.3 K-Means 18
3.4 Interpretazione del grafico 21
3.5 Identificazione delle anomalie 23
4.Ontologie 24
4.1 Protégé 25
4.2 Classi e proprietà dell'ontologia 25
4.3 Esempi di query 27
5.Knowledge Base 29
5.1 Popolamento della KB 29
5.2 Fatti 30
5.3 Regole 30
6.Conclusioni Finali e Sviluppi Futuri 31

1.**Introduzione**

Il nostro progetto nasce dall'interesse nello sport che riponiamo in quanto crediamo che ogni attività sportiva abbia le sue peculiarità e punti di forza, migliorando la salute fisica e mentale dell'atleta. Ci siamo posti l'obbiettivo di trovare i dati delle edizioni delle olimpiadi passate applicando i vari argomenti e concetti spiegati durante il corso. Utilizzando le moderne tecniche di machine learning è possibile produrre nuova conoscenza individuando pattern inaspettati ed effettuando predizioni. Il progetto è possibile sezionarlo in diverse parti, fra cui:

- Apprendimento Supervisionato
- Apprendimento non Supervisionato
- Ontologie: esprimono relazioni, concetti e proprietà includendo informazioni quali atleti, nazioni, sport, eventi e medaglie ecc.
- Knowledge Base in cui l'utente potrà acquisire diverse informazioni sulle edizioni passate e altre curiosità.

1.1 **Elenco argomenti di interesse**

Gli argomenti presi in considerazione sono stati studiati durante il corso di ingegneria della conoscenza, nel progetto sono stati presi in considerazione alcuni argomenti e applicati in OlympiX che verranno approfonditi in seguito:

Apprendimenti Supervisionato:

- Random Forest
- Support Vector Machine
- Gradient Boosting

Apprendimento non Supervisionato

- K-means

1.2 **Dataset e tecnologie utilizzate**

Per OlimpyX abbiamo utilizzato un dataset trovato sul sito *Kaggle* al seguente link: https://www.kaggle.com/datasets/bhanupratapbiswas/olympic-data . Per il funzionamento di OlimpyX abbiamo utilizzato numerose librerie python che ci hanno permesso di realizzare le funzionalità richieste, ovvero:

- Pandas: Per l'elaborazione e manipolazione dei dati presenti dataset
- Scikit-learn: Fornisce un'ampia gamma di algoritmi per apprendimento supervisionato e non supervisionato
- pyswip: Permette di utilizzare SWI-Prolog direttamente da Python
- matplotlib: Per la visualizzazione e creazione di grafici
- owlready2: Per la gestione di ontologie basate su OWL
- numpy: Per calcoli scientifici avanzati.

Per una migliore visione e qualità visiva abbiamo optato per utilizzare delle librerie opzionali che non sono strettamente necessarie al corretto funzionamento del progetto ma ne semplificano la comprensione delle computazioni:

- Tabulate: Stampa i record usando una interfaccia sql-like.
- Seaborn: Per la creazione di tabelle e grafici in modo più semplice e con una migliore estetica rispetto a matplotlib tradizionale.

Oltre alle librerie citate abbiamo utilizzato due software esterni quali SWI-Prolog che ci permette di interrogare la KB e Protégé per la costruzione e gestione dell'ontologia.

2.Apprendimento Supervisionato

Nel nostro caso di studio, l'apprendimento supervisionato è stato utilizzato per predire la possibilità che un atleta possa vincere una medaglia alle Olimpiadi. Il dataset contiene informazioni dettagliate sugli atleti, come età, altezza, peso, sport praticato e paese di appartenenza. Questi dati sono stati utilizzati per costruire un modello in grado di distinguere tra atleti che hanno vinto una medaglia e atleti che non ne hanno vinta.
Per questa parte di progetto, sono state utilizzate le seguenti librerie.

- **Pandas e NumPy** per la manipolazione e gestione dei dati.
- **Scikit-learn(sklearn)** per gli algoritmi di machine learning e la valutazione dei modelli.
- **XGBoost** per implementare Gradient Boosting in modo ottimale.

2.1Caricamento e pulizia dei Dati

Prima dell'addestramento dei modelli di apprendimento supervisionato è stato necessario caricare il file CSV contenente le informazioni sugli atleti (peso, età, altezza, ecc.) ed effettuare delle operazioni preliminari tra cui la gestione dei valori mancanti. La colonna Medal degli atleti che non hanno vinto una medaglia è stata aggiornata sostituendo i valori mancanti con None, mentre le righe con valori mancanti nelle colonne più importanti sono state rimosse. Inoltre per semplificare la gestione delle medaglie abbiamo trasformato quest'ultime in una variabile binaria dove lo 0 rappresenta 'nessuna medaglia' mentre 1 'medaglia'. Questi passaggi ci hanno permesso di ridurre il rischio di bias o valori mancanti che avrebbe potuto inficiare sulle successive fasi.

2.2Preprocessing dei Dati

Dopo la fase di pulizia, il dataset è stato ulteriormente elaborato per prepararlo all'addestramento dei modelli di apprendimento supervisionato. Le variabili categoriali, come il sesso dell'atleta, la squadra, e lo sport praticato, sono state convertite in formato numerico attraverso un processo di codifica, utilizzando tecniche come il **One-Hot Encoding** (per colonne a bassa cardinalità come Sex e Age_Group) e il **Target Encoding** (per colonne ad alta cardinalità come Sport, Event e Team). Queste trasformazioni hanno permesso di rappresentare le variabili categoriali in un formato numerico compatibile con i modelli di machine learning, preservando allo stesso tempo le informazioni rilevanti.

Le feature numeriche (es Age,Height,Weight) sono state normalizzate utilizzando lo StandardScaler garantendo che valori con scale più grandi (esempio Height) potessero dominare il calcolo influenzando negativamente il processo di ottimizzazione dei modelli.

Future Engineering

I dati a nostra disposizione non erano sufficienti a produrre predizioni di qualità, per questo motivo ci siamo abbiamo creato ulteriori feature che permettessero di migliorare la rappresentazione del dataset e aumentare la capacità predittiva dei modelli.

Un esempio di alcune delle feature implementate:

- Experience_Year: anni di esperienza.
- Team_medal_probability: Probabilità storica di successo per ciascun team.
- Height_to_Weight_Ratio: rapporto tra peso e altezza.
- Medals_Per_Participation: rapporto tra il numero totale di medaglie vinte da un atleta e il numero di eventi a cui ha partecipato.

Per ogni atleta abbiamo creato un ranking realistico che è una combinazione ponderata di diverse metriche derivate dai dati, esse rappresentano informazioni importanti sulle performance degli atleti sia a livello individuale che di team, normalizzando il valore finale per garantire che tutti i punteggi si trovino nello stesso range, rendendo la variabile facilmente interpretabile dai modelli.

```
df['Ranking_Proxy'] = (
    0.3 * df['Team_Medals'].rank(ascending=False, method='dense').fillna(0) +
    0.2 * df['Sport_Medal_Probability'].rank(ascending=False, method='dense').fillna(0) +
    0.2 * df['Team_Medal_Probability'].rank(ascending=False, method='dense').fillna(0) +
    0.2 * (1 / (df['Weight'] + 1)).rank(ascending=False, method='dense') +
    0.1 * df['Age'].rank(ascending=True, method='dense')
)

# Normalizziamo il ranking
if 'Ranking_Proxy' in df.columns:
    df['Ranking_Proxy'] = (df['Ranking_Proxy'] - df['Ranking_Proxy'].min()) / (df['Ranking_Proxy'].max() - df['Ranking_Proxy'].min())
```

2.3Divisione Training set e Test set

Per garantire un'adeguata validazione del modello, il dataset è stato diviso in due parti: **il training set, contenente il 70% dei dati, utilizzato per addestrare i modelli il test set, costituito del restante 30%**, utilizzato per valutare le prestazioni finali. Durante questa fase, sono stati applicati accorgimenti per preservare la distribuzione delle classi, specialmente considerando il possibile squilibrio tra atleti vincitori e non vincitori di medaglie. Questo approccio ha garantito un dataset ben preparato per la successiva fase di addestramento e validazione dei modelli.

Il nostro dataset presenta uno squilibrio significativo tra la classe maggioritaria (nessuna medaglia) sulla minoritaria (medaglia), per questo

abbiamo implementato il metodo **SMOTE-Tomek**, che combina il sovracampionamento della classe minoritaria e la rimozione di rumore della classe maggioritaria bilanciando la distribuzione delle classi nel training set migliorando le capacità predittive dei modelli.

Scelta dei modelli: Nel nostro caso di studio, l'apprendimento supervisionato è stato utilizzato con scopo di classificazione. Per effettuare ciò, abbiamo deciso di utilizzare tre modelli di apprendimento:

- Random Forest (balanced)
- Support Vector Machine
- Gradient Boosting (XGBoost)

2.4Scelta degli Iper-parametri

Per ottimizzare le prestazione dei modelli di machine learning applicati al dataset olimpico, abbiamo utilizzato il metodo della Grid Search combinando con la Cross-Validation. Questa combinazione ci ha permesso di identificare i migliori iper-parametri per ogni modello,garantendo una valutazione solida e robusta delle loro prestazioni. La Grid Search esplora in modo sistematico una griglia predefinita di valori per ciascun Iper-parametro, testando tutte le combinazioni possibili. Ogni combinazione è valutata mediante la procedura di cross-validation per determinare quella configurazione massimizza la metrica di valutazione, in questo caso **l'accuratezza.**

In particolare, è stato implementato il metodo StratifiedKFold, che offre un approccio migliorato rispetto alla semplice Cross-Validation tradizionale. Questo metodo è stato scelto per preservare la distribuzione delle classi della variabile target,'Medal', all'interno di ciascun fold. Considerando che nel dataset olimpico la distribuzione delle classi è sbilanciata, con molti atleti senza medaglia rispetto a pochi vincitori, l'uso di StratifiedKFold ha permesso di garantire che ogni fold contenesse le stesse proporzioni delle diverse classi presenti nel dataset.

La procedura di **StratifiedKFold** prevede la suddivisione del dataset in cinque fold di dimensioni simili, dove ciascun fold conserva le proporzioni originali delle classi. A ogni iterazione, un fold viene utilizzato come set di validzione, mentre i restanti quattro fold vengono usati per addestrare il modello. Questo processo viene ripetuto per ciascun fold, assicurando che ogni campione del dataset venga utilizzato sia per l'addestramento sia per la validazione. La metrica di valutazione utilizzata è stata l'accuratezza, e la

performance complessiva del modello è stata determinata calcolando la media delle accuratezze ottenute su tutti i fold.

Iperparametri ricercati e utilizzati

Per gestire un dataset di grandi dimensioni e sbilanciato come il nostro abbiamo deciso di implementare una variante del Random Forest progettata per gestire dataset grandi e sbilanciati, il Balanced Random Forest, infatti, esegue un undersampling casuale della classe maggioritaria ("nessuna medaglia) creando un sottocampione bilanciato rispetto alla classe minoritaria ("medaglia") aumentando sensibilmente le prestazioni.

Random Forest:

- **n.estimators(Numero di Alberi):** sono stati testati i valori 50,100 questi valori aumentano la stabilità e la precisione, specialmente su dataset complessi come il nostro, ma con un costo computazionale maggiore.

- **Max_depth(Massima Profondità degli Alberi):**Sono stati testati i valori 10,20 e None. Il valore None permette agli alberi di crescere fino a quando tutte le foglie contengono un singolo campione o nessun ulteriore split è possibile. I valori 10 e 20 aiutano a controllare la complessità e ridurre il rischio di overfitting, questi valori sono appropriata per un dataset di moderata complessità come il nostro.
- **Min_samples_split(Campioni Minimi per Suddividere un Nodo):** Sono stati testati i valori 2,5.Un valore di 2 consente una maggiore flessibilità, ma potrebbe portare a modelli troppo complessi. Valori alti come **5**, forza ogni split a basarsi su un numero maggiore di campioni, rendendo il modello più robusto e adatto a dataset di grandi dimensioni come il nostro.
- **Min_samples_leaf(Campioni Minimi in una Foglia):** Sono stati testati i valori 1,2 Il valore 1 consente una grande granularità, i valori come 2 forza le foglie a contenere un numero minimo di campioni, il che migliora la generalizzazione.

```
brf_param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
}
```

SUPPORT VECTOR MACHINE(SVM)

Il Support Vector Machine (SVM) è un modello di apprendimento supervisionato che serve a classificare dati in due categorie. L'obiettivo principale è trovare una "linea di separazione" che divida in modo ottimale i dati di classi diverse. Nel nostro progetto, l'SVM aiuterà a prevedere se un atleta ha vinto una medaglia o meno.

- **PARAMETRO C:** è uno degli iper-parametri più importanti in un modello SVM, poiché controlla il trade-off tra la capacità del modello di classificare correttamente i dati di training e la sua capacità di generalizzare bene su nuovi dati è un parametro di regolarizzazione che determina quanto il modello penalizza gli errori di classificazione nei dati di training. Per i **valori alti di C** indicano che il modello è meno tollerante agli errori nel training, anche a costo di margini decisionali più stretti tra le classi (medaglia e nessuna medaglia). Questa strategia è utile quando i dati di training sono puliti. Per i **valori bassi di C** rendono il modello più tollerante agli errori nel set di training. In questo caso, il modello accetta margini più ampi tra le classi,milgiorando la generalizzazione sui dati mai visti.

Per il nostro progetto, abbiamo testato i valori di **C = [0.1,1,10]** che coprono un ampio intervallo di configurazioni:

- **C = 0.1:** Il modello tollera maggiormente gli errori nei dati di training, favorendo la generalizzazione su nuovi dati.
- **C=1:** Rappresenta un buon compromesso tra tolleranza agli errori e adattamento ai dati di training.
- **C=10:** Il modello si concentra sulla classificazione corretta dei dati di training, con margini più stretti.

- **PARAMETRO CLASS WEIGHT:**Il parametro class_weight assegna un peso diverso alle classi nel processo di ottimizzazione del modello. Questo è particolarmente utile per dataset sbilanciati, dove nel caso nostro una classe('nessuna medaglia') ha molti più campioni rispetto all'altra ('medaglia'). Il valore testabile è 'balanced' per calcolare i pesi automaticamente.
- **PARAMETRO MAX_ITER:** specifica il numero massimo di iterazioni dell'algoritmo per garantire la convergenza. Abbiamo testato il valore 2000 che rappresenta un'opzione bilanciata tra tempi di esecuzione e convergenza.

```
# Griglia di parametri per LinearSVC
svc_param_grid = {
    'C': [0.1, 1, 10],  # Parametro di regolarizzazione
    'class_weight': ['balanced'],  # Gestione dello sbilanciamento delle classi
    'max_iter': [2000]  # Iterazioni massime
}
```

Gradient Boosting Classifier:

Per il nostro progetto, è stato scelto una variante del Gradient Boosting ovvero XGBoost (eXtreme Gradient Boosting) che combina alberi di decisione deboli in unsemble (combina le previsioni di più modelli) migliorando la sua efficienza computazionale. A differenza del modello 'classico' le prestazioni sono state sensibilmente migliorate soprattutto in tempo di elaborazione. I parametri del modello utilizzati sono:

- **n_estimators:**Rappresenta il numero totale di alberi che compongono il modello. Un numero maggiore di alberi può migliorare la capacità del modello di catturare le relazioni nei dati, ma aumenta proporzionalmente il tempo di addestramento. Per trovare il giusto compromesso, abbiamo scelto i seguenti valori: [50, 100].
- **learning_rate:** Rappresenta la velocità con cui il modello apprende dai dati. Valori più bassi implicano un apprendimento più lento ma più preciso, riducendo ul rischio di overfitting. Sono stati per questo scelti i seguenti valori: [0.01, 0.1].
- **Max_depth:** Specifica la profondità massima di ciascun albero. Limitare la profondità aiuta a controllare la complessità del modello e migliorata la sua capacità di generalizzazione valori assegnati sono: [3, 5, 7].

```
xgb_param_grid = {
    'n_estimators': [50, 100],
    'learning_rate': [0.01, 0.1],
    'max_depth': [3, 5, 7],
}
```

2.5Addestramento e Valutazione dei Modelli

Per addestrare e valutare i modelli nel nostro progetto, è stata utilizzata la funzione **train_and_evaluate_model_grid_search**, che svolge un ruolo cruciale nella selezione degli iperparametri e nella valutazione delle performance. La funzione prende in input un modello specifico e utilizza **Cross-Validation** con **StratifiedKFold** per suddividere il dataset in 5 fold mantenendo la distribuzione delle classi della variabile target. Esegue una ricerca esaustiva sugli iperparametri specificati nella griglia, utilizzando **GridSearchCV** e trova la configurazione ottimale degli iperparametri per il modello, addestrandolo sul set di addestramento (**X_train** e **y_train**). In seguito, la funzione train_and_evaluate_model_grid_search stampa le seguenti metriche e informazioni:

- **Accuracy**: l'accuratezza del modello sul set di test.
- **Classification Report**: include precision, recall, F1-score e support per ciascuna classe, con relative medie.
- **Training e Test Accuracy**: l'accuratezza del modello sui dati di addestramento e test.
- **Cross-Validation Scores**: i punteggi di accuratezza ottenuti su ciascun fold della cross-validation.
- **Mean CV Accuracy**: la media dei punteggi di accuratezza ottenuti durante la cross-validation.
- **Standard Deviation CV Accuracy**: la deviazione standard dei punteggi di accuratezza tra i diversi fold.
- **Geometric Mean Average Precision (GMAP)**: Calcola la media geometrica delle precisioni medie per ciascuna classe, fornendo una misura equilibrata delle performance del modello su tutte le classi. La GMAP è accompagnata dalla deviazione standard, che rappresenta la variazione di precisione media tra i fold della cross-validation.

Esecuzione e Visualizzazione dei Risultati

Il menù guida l'utente attraverso la selezione del modello e l'esecuzione e valutazione del modello scelto. Inoltre, i report di classificazione e altre metriche vengono stampati direttamente nella console. Si valutano i diversi modelli di apprendimento automatico e si riportano le considerazioni su quelle metriche che si ritengono importanti per valutare il modello non sulla singola iterazione ma sui **5 fold** stabili in fase di sviluppo:

RANDOM FOREST

```
--- Random Forest ---
Best params: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}
              precision    recall  f1-score   support

    No Medal       0.94      0.95      0.94     17980
       Medal       0.66      0.60      0.62      2906

    accuracy                           0.90     20886
   macro avg       0.80      0.77      0.78     20886
weighted avg       0.90      0.90      0.90     20886


Risultati della Cross-Validation:
Cross-Validation Accuracy (mean ± std): 0.9496 ± 0.0011

Training Accuracy: 0.9999
Test Accuracy: 0.9002
Geometric Mean Average Precision (GMAP): 0.9496 (± 0.0011)
```

Cross-Validation-Accuracy (mean ± std): 0.9496 ± 0.0011.

- Questo valore rappresenta la performance media del modello su diverse suddivisioni del dataset, il valore 0.9496 indica che il modello ha ottime prestazioni, classificando correttamente quasi il 95% dei dati in modo consistente, inoltre la deviazione standard molto basse (**± 0.0011**) suggerisce che il modello è estremamente stabile su diverse suddivisioni dei dati.

Accuracy:

- Training Accuracy: 0.999
  L'accuratezza sui dati di training è praticamente perfetta. Questo può indicare che il modello ha imparato estremamente bene i dati di addestramento.
- Test Accuracy: 0.9002.
  L'accuratezza sul test set è buona (90%), leggermente inferiore rispetto al training. Questo suggerisce che il modello riesce a generalizzare bene.

Geometric Mean Average Precision (GMAP) 0.9496 (± 0.0011)

- Il GMAP riflette il bilanciamento delle performance tra le classi. Un valore alto come 0.9496 indica che il modello riesce a trattare bene entrambe le classi, anche in un dataset sbilanciato. In più, la coerenza tra GMAP e Cross-Validation Accuracy mostra che il modello è equilibrato nel riconoscere entrambe le classi senza penalizzare la classe minoritari('Medal').

VALUTAZIONE GENERALE DEL MODELLO

- Il modello Random Forest dimostra prestazioni eccellenti con una **Cross-Validation Accuracy** del 94,96% e un GMAP altrettanto alto, che indica un buon equilibrio tra le classi, anche in presenza di uno

sbilanciamento evidente nei dati. La classe 'No Medal' è classificata con precisione e recall molto elevati (entrambi intorno al 94%-95%), mentre per la classe minoritaria 'Medal', sebbene le prestazioni siano migliorate rispetto a modelli precedenti, il recall (60%) e l'F1-score (62%) rimangono più bassi. C'è una leggera discrepanza tra il **Training Accuracy** (99.99%) e il **Test Accuracy** (90.02%), che potrebbe suggerire un lieve overfitting. Tuttavia, il modello generalizza bene grazie alla robustezza del Random Forest e all'ottimizzazione dei parametri.

SUPPORT VECTOR MACHINE

```
--- Support Vector Machine (LinearSVC) ---
Best params: {'C': 0.1, 'class_weight': 'balanced', 'max_iter': 2000}
              precision    recall  f1-score   support

    No Medal       0.95      0.78      0.86     17980
       Medal       0.36      0.77      0.49      2906

    accuracy                           0.78     20886
   macro avg       0.66      0.77      0.67     20886
weighted avg       0.87      0.78      0.81     20886


Risultati della Cross-Validation:
Cross-Validation Accuracy (mean ± std): 0.7790 ± 0.0033

Training Accuracy: 0.7787
Test Accuracy: 0.7767
Geometric Mean Average Precision (GMAP): 0.7790 (± 0.0033)
```

Cross-Validation-Accuracy(mean ± std): 0.7790 ± 0.0033

- La Cross-Validation Accuracy mostra una buona performance del modello, con una deviazione standard molto bassa (0.0033), il che indica che il modello è abbastanza stabile quando viene testato su diverse suddivisioni di dati.

Accuracy:

- Training Accuracy: 0.7787.
  Questo valore misura quanto bene il modello riesce a fare previsioni corrette sui dati su cui è stato addestrato, un valore alto indica che il modello ha imparato a riconoscere i pattern nei dati di addestramento.
- Test Accuracy: 0.7767.
  L'accuratezza sul test set è leggermente inferiore rispetto a quella sul training set, suggerendo che il modello ha generalizzato bene senza overfitting.

Geometric Mean Avarage Precision (GMAP): 0.7790 (± 0.0033)

- Il GMAP è molto vicino al valore di accuratezza generale e altre metriche (circa 77.9%). Questo suggerisce che il modello non è particolarmente sbilanciato nelle sue predizioni e riesce a catturare un buon equilibrio tra le classi, evitando di ignorare la classe minoritaria('Medal'). La bassa deviazione standard ± 0.0033 indica che il GMAP è consistente quando il modello viene valutato su diversa suddivisione dei dati, confermando la stabilità del modello.

VALUTAZIONE GENERALE DEL MODELLO

- Il modello presenta buone prestazioni generali, con un'accuratezza vicina al 78% e una buona stabilità sia durante la Cross-Validation che nei test. È in grado di gestire abbastanza bene lo sbilanciamento delle classi, ma fatica un pò a prevedere correttamente la classe minoritaria ('Medal'). Nel complesso, il modello è valido, ma potrebbe essere migliorato per aumentare la precisione e il bilanciamento nella classificazione delle classi meno rappresentate.

GRADIENT BOOSTING CLASSIFIER

```
--- Gradient Boosting Classifier ---
Best params: {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 100}
              precision    recall  f1-score   support

    No Medal       0.93      0.92      0.92     17980
       Medal       0.53      0.58      0.55      2906

    accuracy                           0.87     20886
   macro avg       0.73      0.75      0.74     20886
weighted avg       0.88      0.87      0.87     20886


Risultati della Cross-Validation:
Cross-Validation Accuracy (mean ± std): 0.9172 ± 0.0020

Training Accuracy: 0.9271
Test Accuracy: 0.8708
Geometric Mean Average Precision (GMAP): 0.9172 (± 0.0020)
```

Cross-Validation Accuracy (mean ± std): **0.9172 ± 0.0020**

- Il modello ha un'ottima performance media durante la Cross-Validation, con una deviazione standard molto bassa (0.0020). Questo indica stabilità e consistenza tra diverse suddivisioni del dataset.

Accuracy:

- Training Accuracy:0.9271
  Il modello raggiunge una buona accuratezza sui dati di training, indicando che ha appreso bene i pattern presenti nei dati utilizzati per l'addestramento. Questo valore riflette la capacità del modello di adattarsi e apprendere correttamente le informazioni dai dati forniti.
- Test Accuracy:0.8708
  L'accuratezza sul test set è leggermente inferiore, ma comunque alta, indicando una buona capacità di generalizzare e fare previsioni corrette su dati nuovi non visti durante l'addestramento. Questa differenza evidenzia che il modello riesce comunque a mantenere prestazioni elevate anche su dati sconosciuti.

Geometric Mean Average Precision (GMAP): 0.9172 (± 0.0020).

- Il GMAP conferma, in linea con la Cross-Validation Accuracy, conferma che il modello riesce a gestire entrambe le classi in modo equilibrato, nonostante lo sbilanciamento presente nei dati. Questo significa che il modello non si limita a favorire la classe dominante('No Medal') ma cerca di bilanciare le sue predizioni anche per la classe

minoritaria ('Medal'), che ha una rappresentazione significativa inferiore.

VALUTAZIONE GENERALE DEL MODELLO

- Il modello **Gradient Boosting Classifier** mostra buone prestazioni complessive, con un'accuratezza elevata e una capacità affidabile di distinguere le diverse classi. È particolarmente efficace nel gestire la classe dominante, mentre offre risultati accettabili per la classe meno rappresentata. La coerenza tra i risultati di validazione e test evidenzia che il modello è stabile e generalizza bene su dati non visti. Nel complesso, è un modello solido e adatto per applicazioni generiche, con margini di miglioramento nella gestione delle classi minoritarie.

CONCLUSIONI

Nel corso dell'analisi sono stati testati diversi modelli di machine learning su un dataset con un evidente sbilanciamento tra le classi. Ogni modello ha mostrato punti di forza e debolezze, ma alcuni hanno dimostrato una maggiore efficacia nel gestire il problema dello sbilanciamento.

In generale, i modelli Random forest e Gradient Boosting Classifier hanno ottenuto le migliori prestazioni complessive, con un'elevata accuratezza e una buona capacità di generalizzare su dati non visti. Entrambi i modelli sono stati in grado di gestire il dataset in modo stabile e consistente, come evidenziato dai risultati della cross-validation e delle metriche globali.

In conclusione, i risultati ottenuti indicano che i modelli utilizzati sono validi e affidabili per questo tipo di problema. Con ulteriori ottimizzazioni e tecniche di bilanciamento, potrebbero essere ulteriormente migliorati per gestire meglio le situazioni in cui le classi sono fortemente sbilanciate.

3.**Apprendimento non Supervisionato**

L'apprendimento non supervisionato è una tecnica di machine learning utilizzata per esplorare e analizzare dati privi di etichette o categorie predefinite. L'obiettivo principale è scoprire pattern nascosti o gruppi simili nei dati.

Applicazioni:

- Segmentazione dei clienti.
- Analisi esplorativa dei dati.
- Rilevamento di anomalie.
- Riduzione della dimensionalità per la visualizzazione.

Pulizia Dati

Prima di effettuare qualsiasi operazione è buona norma "pulire" il dataset. Abbiamo effettuato alcune operazioni preliminari atte a rimuovere valori duplicati o mancanti o parziali che potessero inficiare sulla qualità delle computazioni.

3.1**Selezione delle caratteristiche**

Per la selezione delle caratteristiche su cui fare l'analisi è stata naturale la scelta dei parametri numerici Peso, Altezza ed Età. Esse, infatti, rappresentano caratteristiche fondamentali e misurabili degli atleti, influenzando significativamente il loro rendimento nei vari sport.

Età: Determina il livello di maturità fisica e prestazioni atletiche. Alcuni sport richiedono atleti giovani (es. ginnastica), altri prediligono maggiore esperienza (es. tiro).

Peso: Influenzano direttamente la tipologia di sport a cui un atleta è più adatto.

Altezza: Determinanti in alcuni sport, tra cui basket, pallavolo o rugby.

Le variazioni in età, peso e altezza creano una buona separazione tra gli atleti, utile per individuare gruppi distinti e caratteristiche uniche per ciascun cluster.

3.2**Processing dei dati**

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

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

scaler = StandardScaler()
X = scaler.fit_ transform(numeric_ df)

Abbiamo ridotto la dimensione del dataset attraverso il PCA (Principal Component Analysis), trasformandolo in uno spazio bidimensionale, riducendo ridondanze e aiutandoci a farci meglio visualizzare i cluster nello scatterplot.

pca = PCA(n_ components=2)
X_ reduced = pca.fit_ transform(X)

3.**3KMeans**

L'algoritmo **K-*means*** è un algoritmo di analisi dei gruppi partizionale che permette di suddividere un insieme di oggetti in *k* gruppi sulla base dei loro attributi.

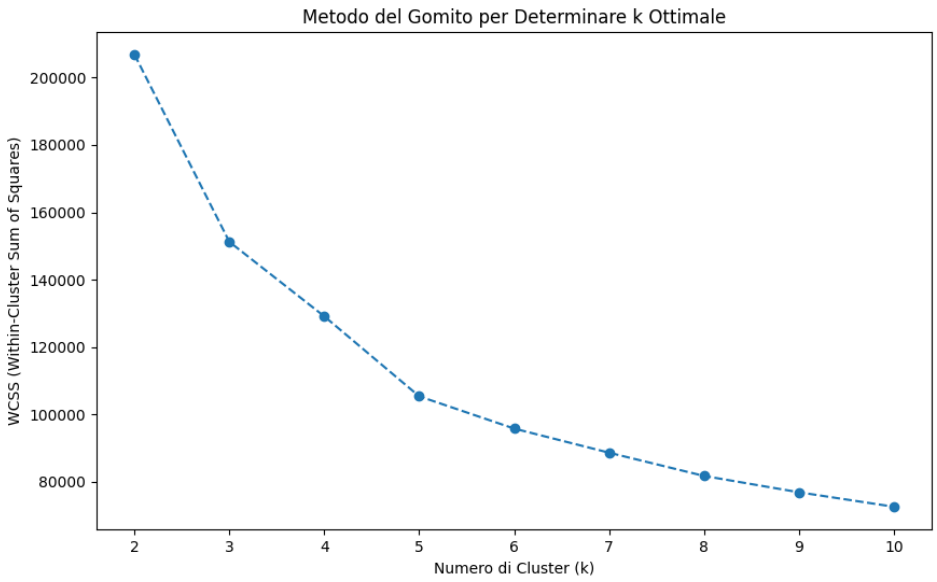
L'algoritmo KMeans segue i seguenti passaggi principali:

1. **Inizializzazione**: Si selezionano casualmente k centroidi iniziali dai dati. Nell'implementazione realizzata, si è utilizzata l'opzione random_state=42 per garantire la riproducibilità dei risultati. Questo approccio è fondamentale per evitare che differenti esecuzioni producano risultati diversi.
2. **Assegnazione**: Ogni punto viene assegnato al cluster il cui centroide è più vicino. La distanza calcolata è generalmente la distanza euclidea nello spazio multidimensionale delle feature.
3. **Aggiornamento**: I centroidi vengono aggiornati calcolando la media delle posizioni di tutti i punti assegnati a ciascun cluster. Questo permette ai centroidi di spostarsi verso le regioni con densità maggiore di dati.
4. **Iterazione**: I passi di assegnazione e aggiornamento vengono ripetuti fino a che:
    a. I centroidi smettono di muoversi (convergenza).
    b. Si raggiunge un numero massimo di iterazioni

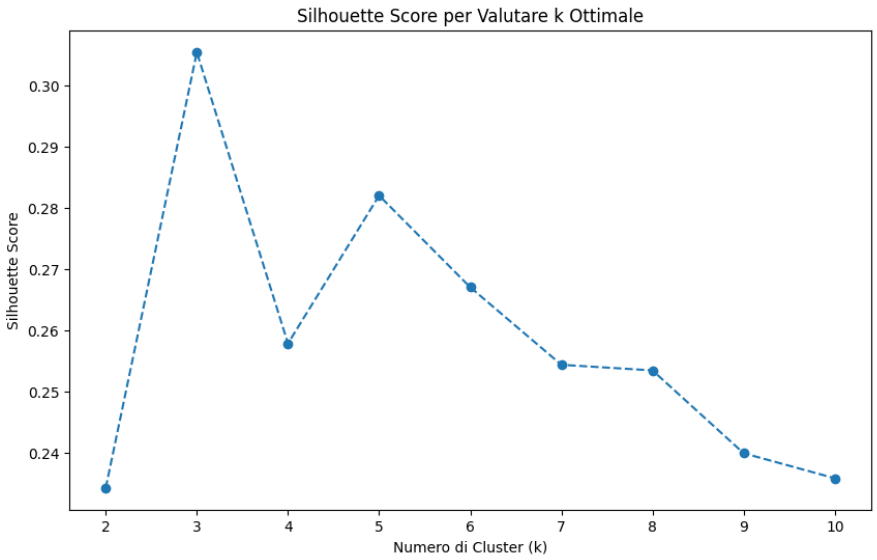
Determinazione del Numero Ottimale di Cluster (K)

Il metodo del gomito è una tecnica visiva per determinare il numero ottimale di cluster in un dataset. Si basa sul calcolo della metrica **WCSS (Within-Cluster Sum of Squares)**, che misura la somma delle distanze quadratiche tra i punti dati e il centroide del loro cluster. Valori più bassi di WCSS indicano cluster più compatti e ben definiti.

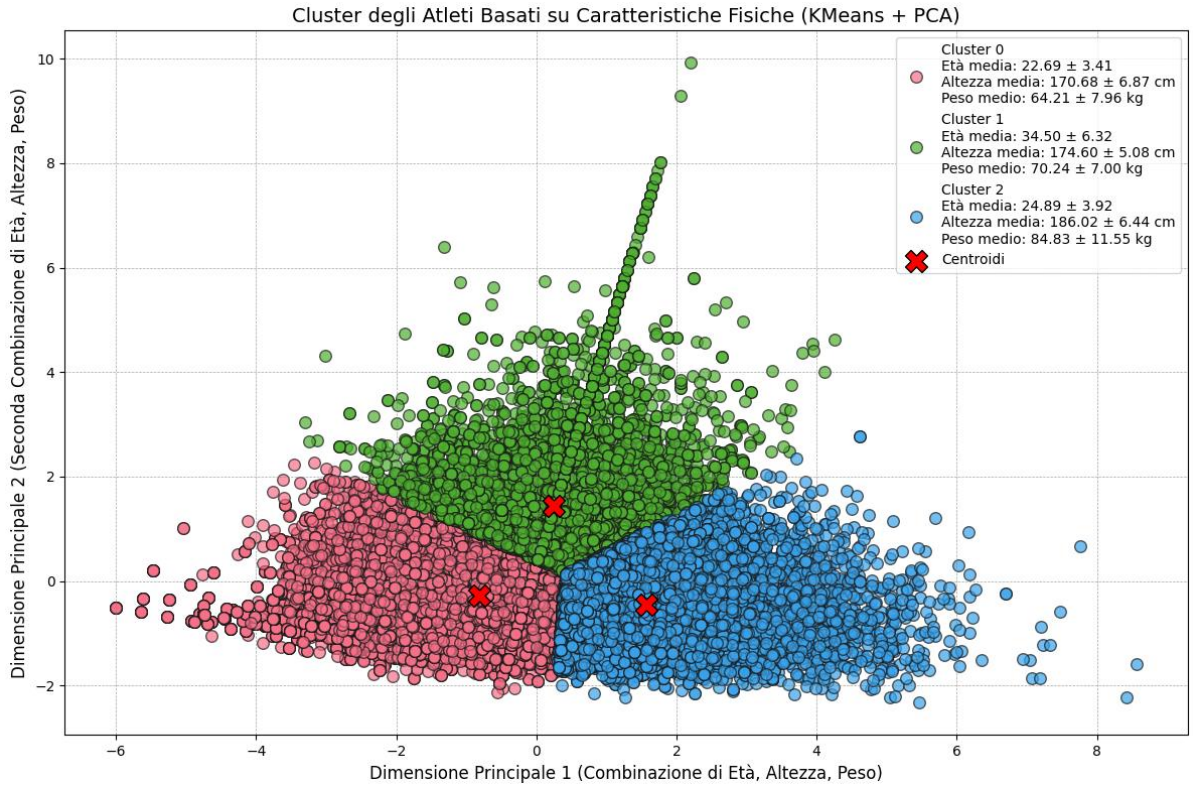
Nel nostro caso, il grafico mostra un chiaro punto di gomito a **k=3**, indicando che tre cluster rappresentano una buona struttura sottostante dei dati. Scegliere un valore di k più alto comporterebbe una maggiore frammentazione, senza apportare significativi miglioramenti alla coesione intra-cluster.



Inoltre, abbiamo deciso di utilizzare il **Silhouette Score,** esso viene calcolato per confermare la qualità dei cluster, i quali, nonostante non presentino valori ottimali, hanno mostrando una separazione discreta tra essi. Il Silhouette Score valuta la separazione e la coesione dei cluster, con valori che vanno da -1 (sovrapposizione dei cluster) a +1 (buona separazione). Nel nostro caso abbiamo ottenuto un punteggio di 0,38 che indica una distanza discreta.



3.4 **Interpretazione del grafico**



Assi:

- **Asse delle ascisse (X)**:
  - L'asse X rappresenta la **prima componente principale**, che cattura la **massima varianza** nei dati combinando le caratteristiche di età, altezza e peso.
  - Gli atleti con valori estremi lungo questo asse differiscono notevolmente dagli altri in termini di una combinazione di età, altezza e peso.
- **Asse delle ordinate (Y)**:
  - L'asse Y rappresenta la **seconda componente principale**, che spiega la seconda maggiore fonte di varianza non catturata dalla prima.
  - Questo asse evidenzia differenze che potrebbero non essere visibili direttamente dalle singole variabili originali.

Colori dei punti:

- I punti nel grafico sono colorati in base al **cluster** a cui appartengono:
  - **Cluster 0**: Rosa
  - **Cluster 1**: Verde
  - **Cluster 2**: Blu

In dettaglio:

Cluster 0 (Rosa):

- **Età media**: 22.69 anni ± 3.41 anni.
- **Altezza media**: 170.68 cm ± 6.87 cm.
- **Peso medio**: 64.21 kg ± 7.96 kg.
- Questo cluster include principalmente atleti più giovani e più leggeri, probabilmente appartenenti a sport che richiedono agilità e una bassa massa corporea, come la ginnastica o il nuoto.

Cluster 1 (Verde):

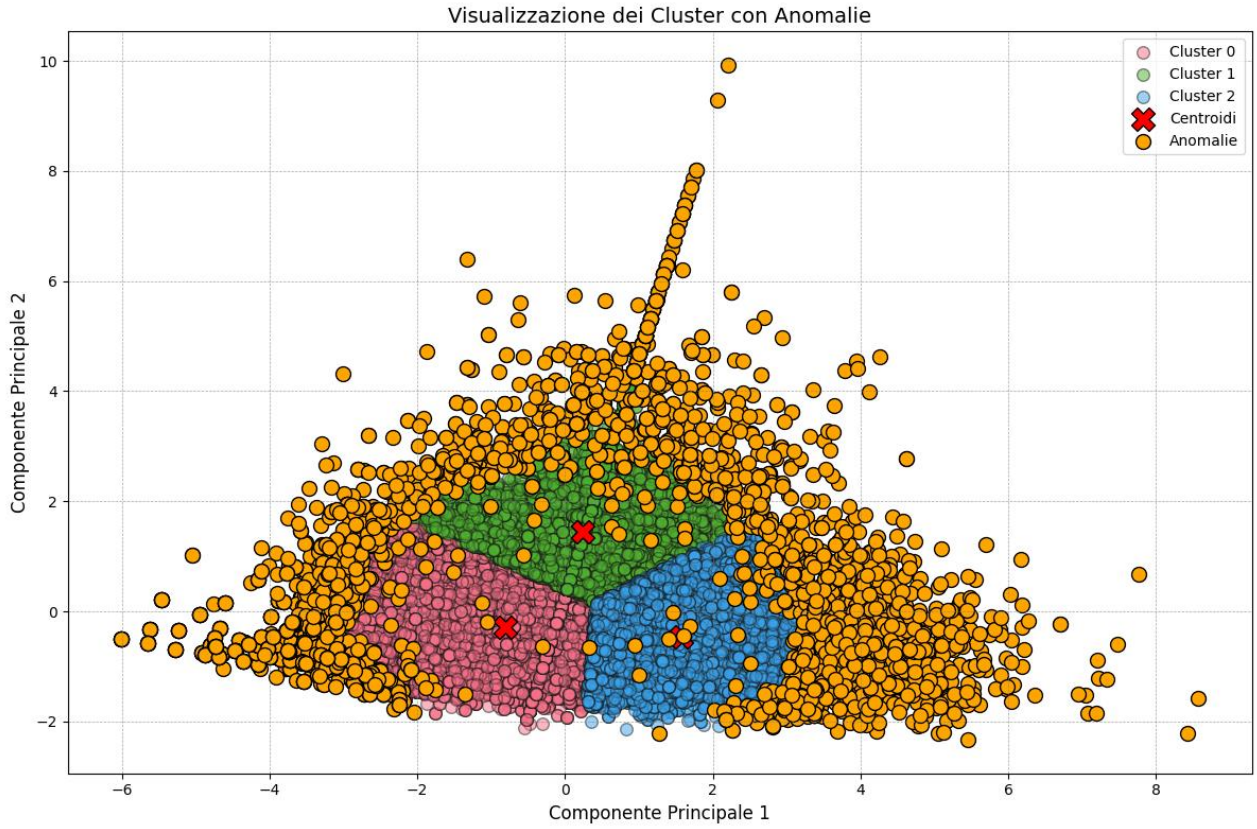
- **Età media**: 34.50 anni ± 6.32 anni.
- **Altezza media**: 174.60 cm ± 5.08 cm.
- **Peso medio**: 70.24 kg ± 7.00 kg.
- Gli atleti in questo cluster sono generalmente più maturi e con una corporatura media. Potrebbero appartenere a sport che richiedono maggiore esperienza e precisione, come il tiro con l'arco o il golf.

Cluster 2 (Blu):

- **Età media**: 24.89 anni ± 3.92 anni.
- **Altezza media**: 186.02 cm ± 6.44 cm.
- **Peso medio**: 84.83 kg ± 11.55 kg.
- Questo cluster rappresenta atleti con una corporatura più grande e alta, spesso legati a sport di forza e potenza, come il basket o il sollevamento pesi.

3.5**Identificazione Anomalie**

Nel grafico delle anomalie, i punti evidenziati in **arancione** rappresentano gli atleti considerati **atipici** rispetto ai cluster a cui appartengono. Questi atleti hanno caratteristiche fisiche (età, altezza, peso) che si discostano significativamente dalla maggior parte degli altri membri del loro cluster. Per identificare le anomalie, è stata utilizzata una soglia basata sul **95° percentile** delle distanze dai centroidi dei cluster. Questo approccio è un compromesso bilanciato per evitare sia troppe anomalie (soglia più bassa) sia troppo poche (soglia più alta).



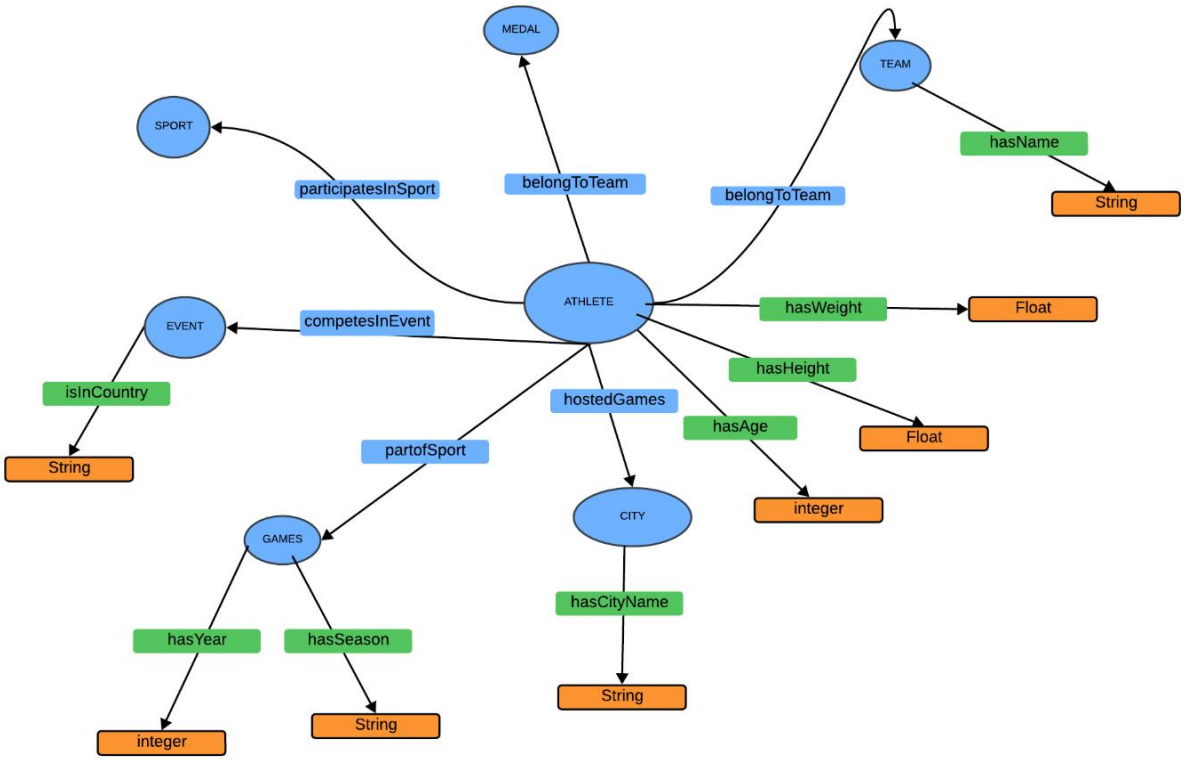
Conclusioni

Grazie all'analisi dei cluster siamo riusciti positivamente a produrre conoscenza attraverso gruppi di atleti che condividono caratteristiche simili, facendo intuire che per determinati sport, vi sono caratteristiche che sono comuni e possano essere indicative dello sport praticato. Inoltre, è possibile migliorare l'analisi delle anomalie cercando di capire se ciò è dovuto da prestazioni eccezionali dell'atleta dal cluster di appartenenza o ad altre cause.

4.Ontologie

Nell'informatica, le ontologie sono modelli formali che rappresentano la conoscenza di un dominio specifico, con la definizione di concetti (classi), relazioni(proprietà) e individui.

Abbiamo scelto di sviluppare un'ontologia che potesse rappresentare al meglio i Giochi Olimpici, con l'obiettivo di rappresentare in modo strutturato e completo le diverse componenti che caratterizzano questo evento globale. L'ontologia permette di modellare informazioni sugli atleti partecipanti, le discipline sportive, gli eventi specifici, le squadre rappresentate e le città ospitanti. Attraverso questa rappresentazione, gli utenti possono esplorare dettagli come le competizioni a cui hanno partecipato gli atleti, le medaglie assegnate, i collegamenti tra sport ed eventi, nonché le edizioni dei Giochi e le relative caratteristiche, come l'anno di svolgimento e la stagione (estiva o invernale). Inoltre, l'ontologia facilita l'analisi tra i diversi elementi, offrendo una visione chiara e approfondita delle associazioni tra gli atleti e i contesti olimpici.

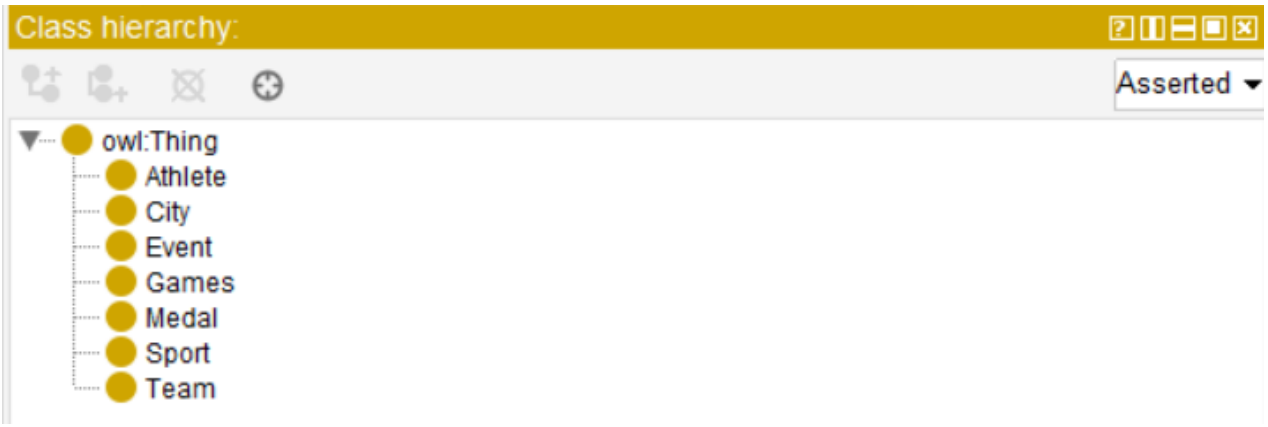


4.1 Protègè

Per la realizzazione dell'Ontologia abbiamo scelto di utilizzare l'editor di ontologie open source Protégé. L'ontologia prevede la presenza di diverse classi, la maggior parte delle quali rappresenta categorie legate ai Giochi Olimpici, descrivendo dettagliatamente elementi come gli sport, gli eventi, le squadre, le città ospitanti e le edizioni dei giochi(year,season,location..).

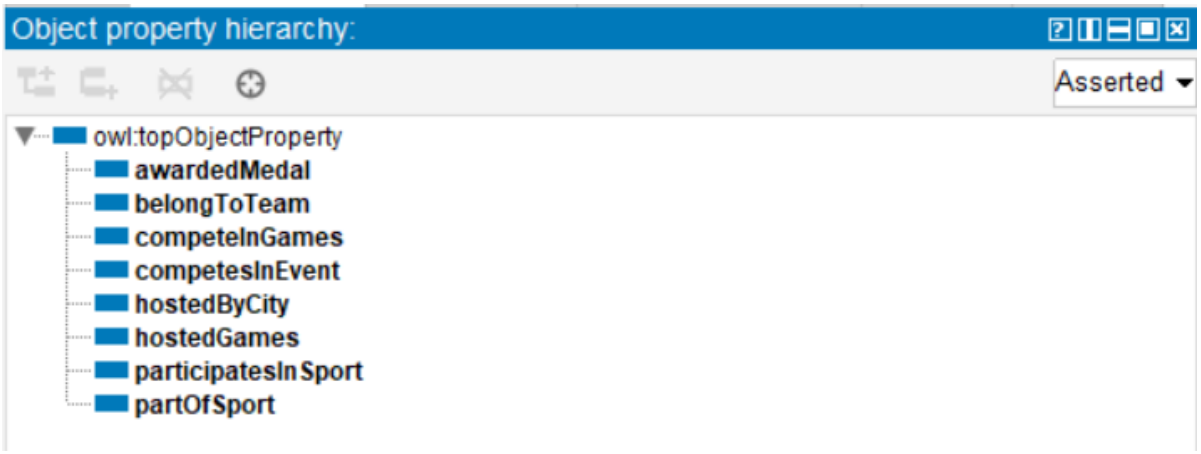
Sono inoltre presenti classi che rappresentano gli "attori" coinvolti nei Giochi Olimpici, come gli atleti, le squadre e i comitati organizzativi, permettendo di modellare in modo completo le relazioni e le informazioni associate a questo evento globale.

4.2 **Classi e proprietà dell'ontologia**

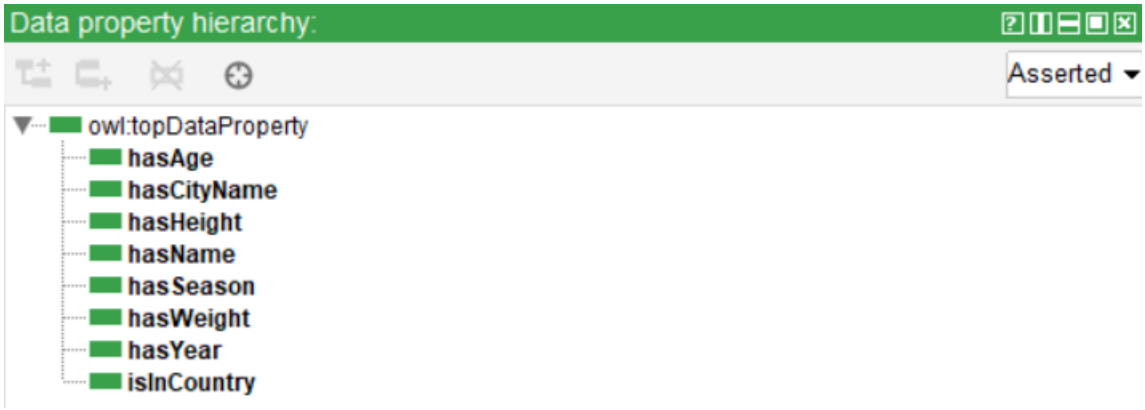


Classi dell'ontologie

Dopo aver creato le classi abbiamo creato, inoltre, una serie di proprietà:object-properties e data-properties.Queste proprietà servono a connettere due individui appartenenti a classi uguali o differenti(object properties)o collegare un individuo a un tipo di dato primitivo (data properties).

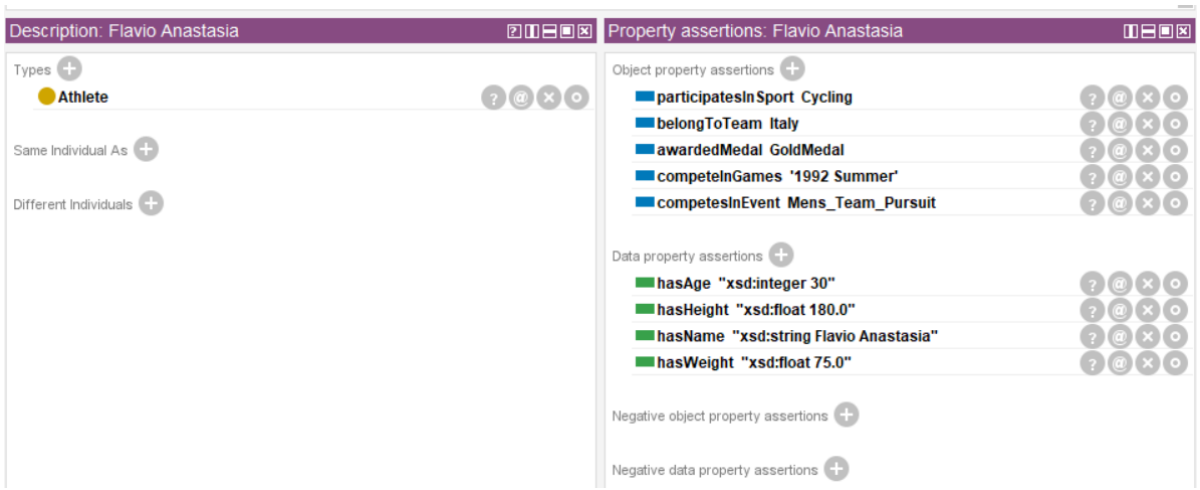


Object property

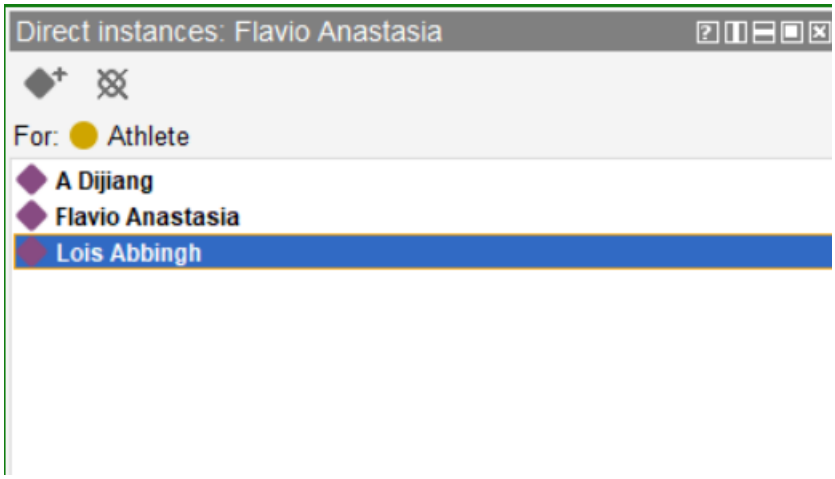


Data property

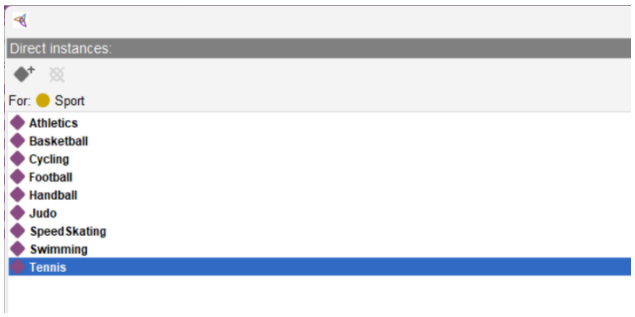
Successivamente siamo passati alla realizzazione degli individui stessi che rappresentano esempi concreti relativi ai Giochi Olimpici. Abbiamo creato individui per atleti, squadre, eventi, edizioni dei giochi e città ospitanti, utilizzando informazioni rilevanti per modellare relazioni e dettagli specifici. Questi individui consentono di illustrare, ad esempio, le competizioni a cui hanno partecipato determinati atleti, le medaglie vinte, i paesi rappresentati e le discipline praticate, offrendo una visione chiara e strutturata nel dataset.



Esempio di individuo con properties annesse



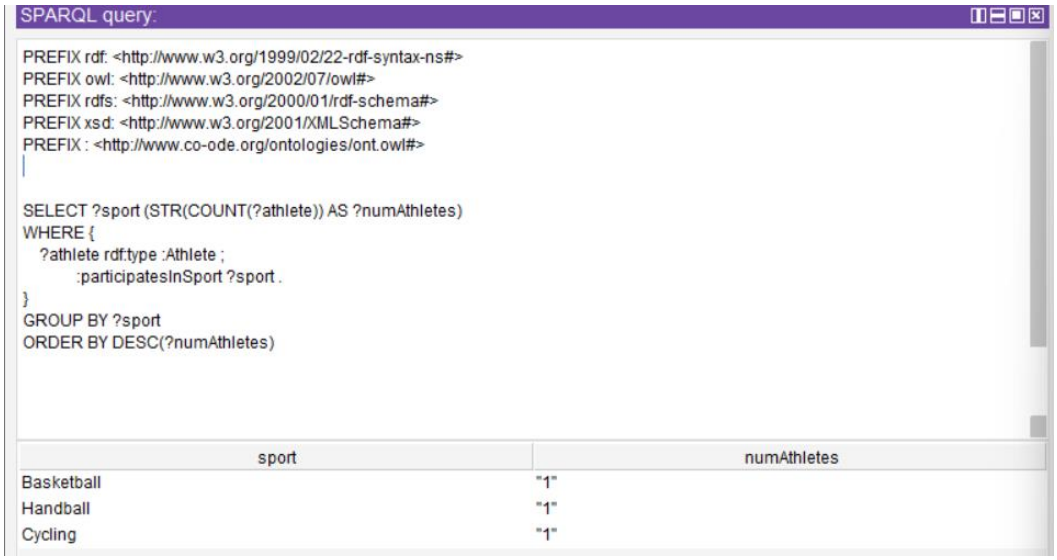
Individui della classe Athlete



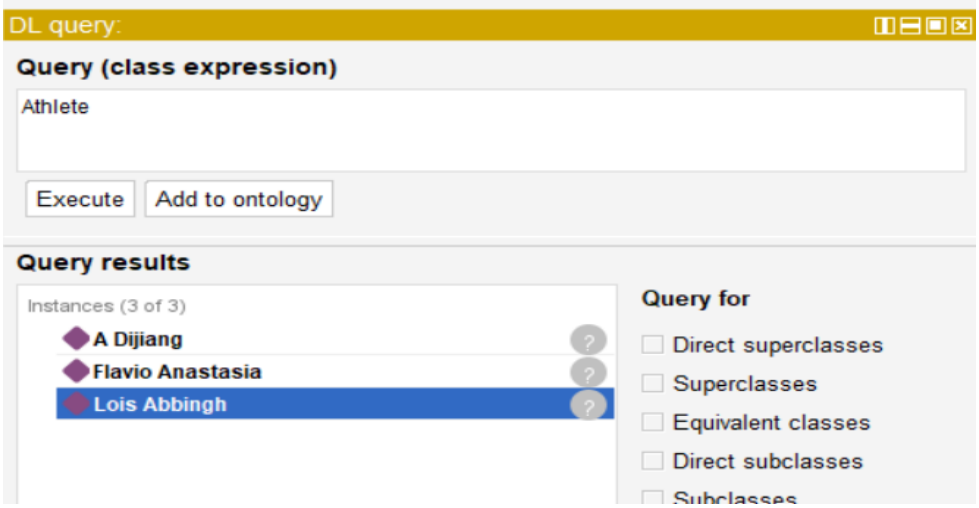
Individui della classe Sport

4.3**Esempi di query**

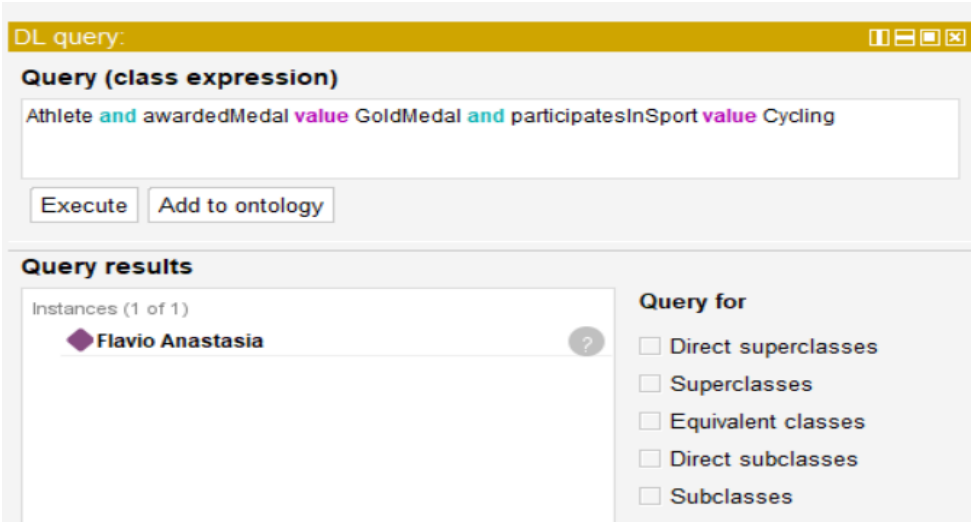
Possiamo interrogare un'ontologia utilizzando due tipologie principali di query:DL Query e SPARQL Query. Entrambe le modalità offrono modi potenti per esplorare e analizzare i dati strutturati in un'ontologia OWL, ma si differenziano per scopi e caratteristiche.



SPARQL query che mostra un conteggio degli atleti che partecipano a uno sport specifico.



DL query per trovare gli individui che appartengono alla classe Athlete



DL Query per atleti che hanno vinto una medaglia d'oro in uno specifico sport

CONCLUSIONI

L'ontologia sviluppata offre una rappresentazione strutturata e completa dei Giochi Olimpici, rappresentando atleti, discipline, eventi, edizioni e città ospitanti. Grazie a Protègè, sono state definite classi, relazioni (object e data proporites) e individui, permettendo un'organizzazione chiara e dettagliata dei dati. L'uso di query come DL Query e SPARQL Query consente di esplorare e analizzare il dataset in modo efficace, evidenziando collegamenti tra gli elementi principali.

5.**Knowledge Base**

Una Knowledge Base è una raccolta di informazioni (fatti) con il quale è possibile rispondere a quesiti, prendere decisioni e produrre conoscenza. Essa è composta dai fatti (assiomi sempre veri) e un insieme di regole, le quali ci permettono di interrogare la kb.

Definiamo formalmente una KB come un insieme di proposizioni, dette assiomi, che consentono di creare un ambiente capace di raccogliere, organizzare e diffondere la conoscenza.

Come definito in precedenza, ci siamo serviti della libreria python Pyswip la quale ci permette di interrogare la KB via codice.

5.1 **Popolamento KB**

Per creare i fatti che sono alla base della kb abbiamo caricato il dataset all'interno del codice in python utilizzando la libreria pandas come segue:

```python
import pandas as pd

cleaned_dataset_path = "cleaned_olympics_dataset.csv"
output_prolog_file = "olympicsKb.pl"

# Carica il dataset
data = pd.read_csv(cleaned_dataset_path)

# Funzione per pulire e formattare le stringhe per Prolog
def clean_string(value):
    if pd.isna(value):
        return "none"  # Rimpiazza valori NaN con 'none'
    return str(value).replace("'", "\\'")  # Escape per apostrofi singoli

# Genera il file Prolog
with open(output_prolog_file, "w", encoding="utf-8") as file:
    file.write("%% Knowledge Base degli atleti olimpici\n\n")
    for _, row in data.iterrows():
        try:
            fact = (
                f"atleta('{clean_string(row['Name'])}', '{clean_string(row['Sex'])}', "
                f"{int(row['Age']) if not pd.isna(row['Age']) else 'none'}, "
                f"{int(row['Height']) if not pd.isna(row['Height']) else 'none'}, "
                f"{int(row['Weight']) if not pd.isna(row['Weight']) else 'none'}, "
                f"'{clean_string(row['Team'])}', {int(row['Year'])}, "
                f"'{clean_string(row['Sport'])}', '{clean_string(row['Event'])}', "
                f"'{clean_string(row['Medal'])}').\n"
            )
            file.write(fact)
        except Exception as e:
            print(f"Errore con riga: {row}\nErrore: {e}")

print(f"File Prolog generato correttamente: {output_prolog_file}")
```

5.2 **Fatti**

Abbiamo caricato i dati del dataset in Python utilizzando la libreria Pandas per creare i fatti di base. Questi includono:

- `name(id, name)`: relazione tra l'identificativo di un atleta e il suo nome.

- `sex(id, sex)`: relazione tra identificativo e sesso.

- `age(id, age)`: età degli atleti.

- `team(id, team)`: squadra o nazione rappresentata.

- Altri, come altezza, peso, anno di partecipazione, sport, evento e medaglia.

5.3 **Regole**

Le regole definiscono la logica attraverso cui la KB risponde alle interrogazioni. Le regole sfruttano i fatti presenti nella KB per fornire all'utente sapere, nel nostro progetto abbiamo incluso diverse opzioni che l'utente può utilizzare per interrogare la kb. Alcune delle regole implementate sono:

- miglior_medaglia(Nome, Medaglia).
  Determina il miglior risultato (medaglia) ottenuto da un atleta.
- atleta_piu_giovane_o_anziano(Tipo, Nome, Eta).
  Determina l'atleta più giovane o anziano che ha vinto una medaglia.
- distribuzione_medaglie_per_edizione(Anno, Conteggio).
  Determina la distribuzione delle medaglie per ogni edizione. Abbiamo implementato un menu che dà la possibilità di scegliere diverse opzioni, ovvero:

```
--- Menu Opzioni ---
1. Cerca informazioni su un atleta
2. Trova tutti gli atleti di una nazionalità
3. Trova atleti che hanno vinto medaglie in uno sport
4. Trova il miglior risultato di un atleta in termini di medaglia
5. Trova gli sport più popolari in termini di medaglie vinte da una nazione
6. Trova l'atleta più giovane/anziano che ha vinto una medaglia
7. Trova il rapporto medaglie/partecipanti per una nazione
8. Trova la distribuzione delle medaglie per edizione
9. Esci
Scegli un'opzione (1-9):
```

CONCLUSIONI

L'implementazione della KB insieme alle regole in SWI-Prolog ci hanno permesso di estrarre informazioni utili riguardo al dominio. E' stato possibile analizzare prestazioni, tendenze demografiche e storiche, e altre informazioni utili. Questo sistema dimostra l'efficacia della logica dichiarativa nell'analisi di dati strutturati, offrendo una base solida per ulteriori applicazioni e analisi future.

6.CONCLUSIONI FINALI E SVILUPPI FUTURI

Il nostro progetto ha dimostrato come le moderne tecniche di apprendimento supervisionato e apprendimento non supervisionato possano essere utilizzate su un dominio complesso come quello che abbiamo utilizzato. Nonostante i risultati promettenti, il nostro progetto non è esente da problemi presentando diversi limiti.

Si può pensare di espandere il progetto implementano dashboard interattive che permettono agli utenti di esplorare i dati olimpici attraverso grafici dinamici e visualizzazione che mostrano trend, confronti storici e connessioni tra atleti, discipline e città ospitanti.

Potrebbe essere un'implementazione futura una funzionalità per aggiornare automaticamente il dataset e l'ontologia con i risultati delle edizioni olimpiche future, mantenendo il progetto aggiornato e rilevante.