

# OLYMPIC MEDALIST PREDICTION MODEL

A.A. 2025-2026

SILVIA SOLAZZO [779231]

S.SOLAZZO9@STUDENTI.UNIBA.IT

LINK GITHUB:

[HTTPS://GITHUB.COM/SILSOLA/OLYMPIC-MEDALIST-PREDICTION-MODEL.GIT](https://github.com/SILSOLA/OLYMPIC-MEDALIST-PREDICTION-MODEL.GIT)

Prof. Nicola Fanizzi

## SOMMARIO

### 1. **INTRODUZIONE**

- 1.1. Argomenti di interesse
- 1.2. Librerie utilizzate

### 2. **CREAZIONE DEL DATASET**

- 2.1. Pulizia e Preprocessing dei dati
  - 2.1.1. Normalizzazione delle colonne e definizione del target
  - 2.1.2. Conversioni variabili categoriali
  - 2.1.3. Standardizzazione dei dati

### 3. **APPRENDIMENTO SUPERVISIONATO**

- 3.1. Preparazione delle feature e del target
- 3.2. Addestramento dei modelli
  - 3.2.1. Regressione logistica (classificazione)
    - 3.2.1.1. Scelte progettuali
    - 3.2.1.2. Valutazione del modello
    - 3.2.1.3. Analisi dell'overfitting
    - 3.2.1.4. Risultati ottenuti
  - 3.2.2. Random Forest
    - 3.2.2.1. Iperparametri di Random Forest
    - 3.2.2.2. Ottimizzazione degli iperparametri con Grid Search e Cross Validation
    - 3.2.2.3. Gestione dei risultati e file di output
    - 3.2.2.4. Analisi dell'overfitting e risultati ottenuti
- 3.3. Conclusioni
- 3.4. Grafici

### 4. **APPRENDIMENTO PROBABILISTICO**

- 4.1. Scelte progettuali e Funzionamento
- 4.2. Iperparametri
  - 4.2.1. Creazione della tabella dei risultati
- 4.3. Gestione dell'incertezza e probabilità
- 4.4. Valutazione del modello
- 4.5. Grafici

### 5. **RAPPRESENTAZIONE DELLA CONOSCENZA – KB**

- 5.1. Struttura della KB
  - 5.1.1. Fatti
  - 5.1.2. Regole
- 5.2. Ragionamento inferenziale nella KB
- 5.3. Conclusioni
- 5.4. Esempi

### 6. **ANALISI DEL MEDAGLIERE**

### 7. **DIPENDENZE**

- 7.1. Configurazione dell'ambiente

### 8. **DOCUMENTAZIONE**

### 9. **RIFERIMENTI BIBLIOGRAFICI**

## 1. INTRODUZIONE

Questo progetto si propone di sviluppare un sistema predittivo intelligente in grado di stimare la probabilità che un atleta conquisti una medaglia olimpica, basandosi sull'elaborazione dei dati storici delle edizioni precedenti. Il sistema non si limita ad un'analisi puramente statistica, ma integra un agente decisionale ibrido che combina il Machine Learning con una base di conoscenza logica per fornire consigli motivati e contestualizzati.

Il progetto è stato sviluppato in linguaggio **Python** (versione 3.9.6.) e per la componente di ragionamento logico, è stato integrato il linguaggio **Prolog**.

Per l'esecuzione del progetto, è sufficiente avviare il file principale `main.py` posizionato nella cartella radice del progetto (*assicurarsi di aver installato le dipendenze elencate nel file `requirements.txt` e di aver un interprete Prolog (es. SWI-Prolog) configurato correttamente nel sistema*).

### 1.1. Argomenti di interesse

- Apprendimento supervisionato
- Apprendimento probabilistico
- Knowledge Base per supportare il processo decisionale e l'inferenza basata su regole

### 1.2. Librerie utilizzate

Di seguito sono elencate le principali librerie utilizzate nel progetto, classificate in base alla loro funzionalità:

#### GESTIONE E ANALISI DEI DATI

- **PANDAS**: libreria utilizzata per il caricamento del dataset degli atleti, la gestione dei valori nulli e la manipolazione dei dati stessi.
- **NUMPY**: libreria fondamentale per il calcolo numerico. Viene utilizzata dai modelli di Machine Learning per la gestione di matrici di dati e calcolo di metriche.

#### MACHINE LEARNING E MODELLISTICA

- **SCIKIT-LEARN**: libreria di Machine Learning che fornisce l'implementazione degli algoritmi (Logistic Regression, Random Forest e Naive Bayes), gli strumenti per la Standardizzazione delle feature e le classi per l'ottimizzazione degli iperparametri (GridSearchCV).
- **JOBLIB**: libreria utilizzata per il salvataggio dei modelli in formato binario (`.pkl`) e il caricamento di oggetti Python.

#### VISUALIZZAZIONE STATISTICA

- **MATPLOTLIB**: libreria utilizzata per la creazione di grafici, usati per il confronto delle prestazioni tra i modelli.
- **SEABORN**: libreria basata su `matplotlib`, usata per la generazione di grafici avanzati

#### INTEGRAZIONE LOGICA E UTILITY

- **PYSWIP (Prolog)**: libreria che permette l'interazione tra Python e il linguaggio Prolog. Consente la gestione di una Knowledge Base (KB) e l'esecuzione di inferenze logiche basate su regole.
- **THEFUZZ/PYTHON-LE LEVENSHTTEIN**: Implementano algoritmi di fuzzy string matching. Sono essenziali per rendere il sistema robusto, consentendo di riconoscere correttamente i codici nazione (NOC) e gli sport anche in presenza di piccoli errori di battitura da parte dell'utente.
- **JSON**: libreria standard di Python per la gestione dei file in formato JSON. Quindi utilizzata per salvare configurazioni e risultati in formato strutturato e leggibile.

- **OS:** modulo per la gestione dei percorsi (path) e delle cartelle del progetto. Fornisce un modo per interagire con il sistema operativo, garantendo che i file vengano salvati correttamente nelle rispettive directory.
- **SYS:** fornisce funzioni e variabili per interagire con l'interprete, come la gestione degli argomenti da riga di comando, il controllo del flusso di output e l'accesso al sistema operativo.

## 2. CREAZIONE DEL DATASET

Il **dataset** utilizzato in questo progetto è stato scaricato dalla piattaforma <https://www.kaggle.com/datasets> e si trova all'interno della cartella dataset con il nome `olympics_dataset.csv`.

Il file raccoglie informazioni relative ai Giochi Olimpici estivi, coprendo l'arco temporale che va da Atene 1896 a Parigi 2024. Questa base di dati permette di osservare le caratteristiche fisiche e i risultati degli atleti nel corso del tempo.

Nello specifico, il dataset è composto dalle seguenti colonne:

- **Player\_id**: Identificatore univoco per ogni atleta
- **Nome**: Nome dell'atleta
- **Sesso**: Genere dell'atleta (indicato con *M* per maschio o *F* per femmina)
- **Squadra**: Nome della squadra o della delegazione nazionale di appartenenza
- **NOC**: Codice del Comitato Olimpico Nazionale per il Paese
- **Anno**: Anno dei giochi olimpici
- **Stagione**: Stagione dei Giochi Olimpici (Estate).
- **Città**: Città ospitante dei Giochi Olimpici
- **Sport**: Categoria sportiva in cui l'atleta ha gareggiato
- **Evento**: Evento specifico all'interno dello sport
- **Medaglia**: Tipo di medaglia assegnata (Oro, Argento, Bronzo) o "Nessuna medaglia" se non assegnata

Questo dataset offre l'opportunità di svolgere un'analisi completa delle tendenze delle prestazioni olimpiche, dei risultati specifici del paese e della progressione di varie discipline sportive nel tempo. Attraverso lo studio di queste variabili, il progetto mira a implementare un modello di Machine Learning capace di prevedere la probabilità vittoria di una medaglia per un determinato atleta.

### 2.1. Pulizia e Preprocessing dei dati

La **pulizia dei dati** è una fase fondamentale del progetto, poiché garantisce che il dataset sia coerente e privo di anomalie prima del suo utilizzo nel modello. Questo processo trasforma i dati grezzi in un formato numerico e normalizzato, adatto all'elaborazione tramite algoritmi di Machine Learning.

Tali operazioni si trovano nel file `dataset_utils.py`, all'interno della cartella `dataset`. Di seguito vengono descritti analiticamente i passaggi eseguiti:

#### 2.1.1. Normalizzazione delle Colonne e Definizione del Target

Per prevenire errori di sintassi e assicurare che ogni colonna abbia un nome uniforme, i nomi delle colonne sono stati ripuliti da eventuali spazi bianchi andando a sostituire questi con il carattere `_`.

```
df.columns = [col.strip().replace(' ', '_') for col in df.columns]
```

#### 2.1.2. Conversioni Variabili Categoricali

- La colonna **Medal** contiene valori testuali riguardante la tipologia di medaglia (Gold, Silver o Bronze). Per questo motivo viene trasformata nella variabile binaria **Won\_Medal**: se l'atleta ha ottenuto una medaglia (Gold, Silver o Bronze) il valore assegnato è **1**, altrimenti è **0**.

```
df['Won_Medal'] = df['Medal'].apply(lambda x: 1 if str(x) in ['Gold', 'Silver', 'Bronze'] else 0)
```

- Anche la colonna **Sex** (che indica il genere dell'atleta) contiene valori testuali. Poiché i modelli di Machine Learning lavorano meglio con dati numerici, questa variabile è stata convertita in numeri: Sex=0 significa che l'atleta è maschio (M), Sex=1 significa che l'atleta è femmina (F).

```
df['Sex'] = df['Sex'].map({'M':0, 'F':1})
```

- Infine, sono state gestite anche le colonne relative alla disciplina sportiva praticata dall'atleta (**Sport**) e quella sul Comitato Olimpico Nazionale (**NOC**). Per avere una coerenza del sistema, sono state convertite in una variabile categorica: ad ogni tipo di Sport/NOC viene associato un numero e viene creata una mapping tra i codici e la corrispondenza originale per riconvertire i dati; questo mapping inoltre viene salvato su disco (sport\_mapping.pkl / noc\_mapping.pkl) ed è situato nella cartella modelli dei due addestramenti.

```
df['Sport'] = df['Sport'].astype('category')
sport_mapping = dict(enumerate(df['Sport'].cat.categories))
joblib.dump(sport_mapping, 'modelli/sport_mapping.pkl')
os.makedirs('modelli', exist_ok=True)
df['Sport'] = df['Sport'].cat.codes

df['NOC'] = df['NOC'].astype('category')
noc_mapping = dict(enumerate(df['NOC'].cat.categories))
joblib.dump(noc_mapping, 'modelli/noc_mapping.pkl')
df['NOC_ID'] = df['NOC'].cat.codes
```

Durante questa fase ho effettuato una **Feature Selection mirata**: variabili come Name o Season sono state escluse perché fungono solo da identificativi univoci. Pertanto, sono state elaborate e decodificate solo le variabili categoriche che l'utente è chiamato a inserire in fase di test.

---

### 2.1.3. Standardizzazione dei Dati

Sebbene il dataset finale si concentri su variabili categoriche, la **Standardizzazione** è stata utilizzata nel preprocessing per uniformare i vettori di input. Questa operazione trasforma i codici numerici derivanti dall'encoding (NOC e Sport) affinché abbiano media nulla e varianza unitaria.

Tale procedura previene che i pesi del modello di Regressione Logistica vengano influenzati dalla magnitudo arbitraria degli identificativi numerici (es. uno Sport con ID 200 rispetto a uno con ID 5), garantendo una convergenza più stabile dell'ottimizzatore e una corretta integrazione dei segnali probabilistici nel sistema ibrido.

### 3. APPRENDIMENTO SUPERVISIONATO

In questa fase di progettazione, l'obiettivo è addestrare algoritmi capaci di memorizzare una relazione tra gli input e gli output, affinché siano in grado di fare previsioni accurate su dati mai inseriti prima.

Questa fase si scompone in:

- Scelta degli iperparametri
- Fase di addestramento
- Fase di test
- Valutazione delle prestazioni

#### 3.1. Preparazione delle Feature e del Target

In questa fase vengono scelte le **feature**, cioè le variabili indipendenti e la **variabile target** da predire.

Le feature che ho scelto sono: `Sex`, `NOC`, `Sport` perché sono le variabili che influiscono maggiormente nella predizione di vittoria della medaglia olimpica da parte di un atleta.

- `Sex`: genere dell'atleta (indicato con *M* per maschio o *F* per femmina)
- `NOC`: Comitato Olimpico Nazionale codice a 3 lettere (es. ITA per Italia)
- `Sport`: disciplina sportiva praticata dall'atleta

Mentre la variabile target è `Won_Medal` cioè se l'atleta potrebbe vincere o meno una medaglia.

Il dataset viene quindi suddiviso in due parti:

- Un **training set** per l'addestramento dei modelli
- Un **test set** per la valutazione delle prestazioni

#### 3.2. Addestramento dei Modelli

Vengono così addestrati due modelli di apprendimento supervisionato:

- Modello di **Regressione Logistica** (Classificazione Lineare)
- Modello di **Random Forest**

##### 3.2.1. Regressione Logistica (Classificazione)

La **Classificazione lineare** è stata adottata come *baseline* (modello di riferimento) per affrontare il problema della previsione del successo olimpico. Questo modello permette di stabilire una relazione logistica tra le variabili indipendenti (feature di input) e la variabile target (vittoria di una medaglia).

In questo progetto si è scelto di impiegare la **Regressione Logistica**, un classificatore basato su una funzione lineare "compressa" (*squashed*). Il modello, quindi, utilizza una funzione di attivazione **sigmoidea** per mappare i risultati in un intervallo reale compreso tra  $[0,1]$ , permettendo così di interpretare l'output come la probabilità di appartenenza dell'atleta alla classe positiva (vincitore di medaglia).

##### 3.2.1.1. Scelte Progettuali

Il modello è stato implementato tramite la libreria `scikit-learn`. Per questo classificatore non è stata applicata una ricerca estensiva degli iperparametri tramite *Grid Search*, in quanto il suo scopo principale è fornire una soglia minima di prestazione per modelli non lineari più complessi.

Per garantire la convergenza dell’algoritmo di ottimizzazione e una corretta interpretazione dei coefficienti, i dati sono stati preventivamente **standardizzati** (media zero e varianza unitaria). Questa operazione di feature scaling è fondamentale per evitare che variabili con scale numeriche differenti influenzino in modo sproporzionato il calcolo del gradiente.

### 3.2.1.2. Valutazione Del Modello

La valutazione del modello è stata eseguita mediante una **3-Fold Cross-Validation**. Questo approccio garantisce che ogni porzione del dataset venga utilizzata sia per l’addestramento che per la validazione, fornendo una stima della capacità di generalizzazione dell’agente più robusta e meno dipendente dal caso.

Per misurare il successo, sono state analizzate le seguenti metriche:

- **Accuracy Score:** rappresenta la percentuale media di predizioni corrette sull’intero dataset;
- **F1-Score:** adottata come metrica cardinale perché media armonica tra Precision e Recall, essenziale per valutare il modello su un dataset sbilanciato come quello olimpico;
- **Precision e Recall:** utili a discriminare rispettivamente la capacità del modello di non generare falsi positivi e di identificare correttamente tutti i reali vincitori.

### 3.2.1.3. Analisi dell’Overfitting

Il **sovradattamento** si manifesta quando il modello impara regolarità specifiche del training set che non si riflettono nel test set o nel mondo reale.

Per monitorare la tenuta del modello, si è analizzata la **varianza delle prestazioni** tra i diversi fold della validazione incrociata. Una deviazione standard contenuta nelle metriche finali indica che il modello ha trovato equilibrio ottimale nel **bias-variance tradeoff**, catturando pattern sistematici e generali del dominio olimpico invece di memorizzare record specifici.

### 3.2.1.4. Risultati Ottenuti

I risultati ottenuti dal modello di classificazione lineare (Logistic Regression) mostrano una notevole stabilità statistica, come indicato dalle deviazioni standard estremamente ridotte. Tuttavia, un’analisi critica delle metriche evidenzia i limiti strutturali di un approccio lineare per questo specifico dominio:

- **Accuratezza vs F1-Score:** Nonostante un'accuratezza media del **54.97%**, il valore dell'F1-Score (**27.15%**) e della Precision (**18.07%**) rivela che il modello fatica sensibilmente a distinguere i rari casi di successo olimpico dal rumore di fondo.
- **Gestione dello Sbilanciamento:** Il basso valore di Precision indica che il classificatore genera un numero elevato di "falsi allarmi", faticando a isolare i veri vincitori di medaglia all'interno di un dataset dove la classe "No Medaglia" è fortemente predominante.
- **Capacità Predittiva:** Sebbene il Recall (54.59%) indichi che il modello riesce a intercettare circa la metà dei vincitori reali, lo fa a costo di una precisione molto bassa, confermando che la frontiera di decisione lineare non è sufficientemente complessa per mappare correttamente le dinamiche del successo atletico.

MODELLO	METRICA	MEDIA (CV)	DEV. STANDARD ( $\pm$ )
Logistic Regression	Accuracy	54.97%	0.21%
	Precision	18.07%	0.11%
	Recall	54.59%	0.43%
	F1	27.15%	0.17%

In sintesi, la Regressione Logistica assolve al suo compito di baseline, dimostrando però la necessità di passare a modelli non lineari e più sofisticati per catturare le interdipendenze tra nazionalità, sesso e disciplina sportiva.

### 3.2.2. Random Forest

Il secondo modello analizzato è il **Random Forest**, selezionato per superare i limiti di separabilità lineare riscontrati nella baseline. A differenza della Regressione Logistica, questo approccio non-lineare permette di mappare le interazioni cross-feature (ad esempio, la correlazione specifica tra un determinato NOC e uno Sport specifico) che definiscono il successo olimpico.

L'uso di un *ensemble* di alberi indipendenti è stato preferito al singolo albero di decisione per mitigare la naturale tendenza di questi ultimi all'overfitting. Attraverso il **Bootstrap Aggregating**, ogni stimatore analizza una porzione differente dello spazio campionario, garantendo una previsione finale più robusta e meno sensibile ai valori anomali (outlier) presenti nei record storici.

Data la rarità dell'evento "vittoria" nel dataset, è stato fondamentale l'utilizzo del parametro `class_weight='balanced'`. Questa scelta tecnica impone al modello di penalizzare maggiormente gli errori sulla classe minoritaria (vincitori), riequilibrando artificialmente la funzione di costo senza dover ricorrere a tecniche di campionamento invasivo come SMOTE.

#### 3.2.2.1. Iperparametri del modello Random Forest

Gli **iperparametri** regolano la complessità e la capacità di generalizzazione del modello. Per il Random Forest, i parametri principali ottimizzati scelti sono:

L'architettura della foresta è stata definita attraverso una ricerca sistematica nello spazio degli stati degli iperparametri, motivata dalle seguenti necessità tecniche:

- **n\_estimators**: il numero di alberi nella foresta. La scelta di testare valori tra 50 e 100 è finalizzata a trovare il punto di convergenza oltre il quale l'errore di generalizzazione si stabilizza, minimizzando il tempo di ingerenza senza perdere precisione.
- **max\_depth**: la profondità massima di ogni albero. È il parametro critico per controllare l'overfitting. Limitare la profondità è stato necessario per impedire agli alberi di “memorizzare” il rumore del dataset e costringerli a estrarre regole decisionali generali.
- **min\_samples\_split**: il numero minimo di campioni richiesti per suddividere un nodo intero, utile per impedire una crescita eccessiva degli alberi. È stato impostato per evitare la creazione di foglie troppo specifiche che avrebbero descritto singoli atleti anziché nazionali.
- **class\_weight**: impostato su “*balanced*” per gestire eventuali sbilanciamenti nelle classi del dataset olimpico.

```
param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [10, 20],
    'min_samples_split': [5, 10],
    'class_weight': ['balanced']
}
```

#### 3.2.2.2. Ottimizzazione degli Iperparametri con Grid Search e Cross Validation

Per individuare la combinazione ottimale è stata utilizzata la **Grid Search**, una tecnica che esplora sistematicamente lo spazio degli iperparametri valutando ogni combinazione possibile.

La valutazione è stata condotta tramite **3-Fold Cross-Validation**: il dataset viene suddiviso in tre sottoinsiemi (folds); ciclicamente, il modello viene addestrato su due di essi e validato sul rimanente. Dato l'ampio dataset olimpico, una partizione in 3 fold garantisce che ogni ciclo di addestramento abbia una base statistica sufficientemente ampia per modellare interazioni non lineari, garantendo al contempo che la valutazione avvenga su dati non visti dal modello durante il training.

### 3.2.2.3. Gestione dei Risultati e File di Output

Per garantire la riproducibilità dell'esperimento, i risultati vengono salvati in file strutturati:

- **Tabella dei risultati:** il file CSV in `iperparametri/tabelle/grid_search_rf.csv` contiene l'elenco completo delle combinazioni testate con i relativi punteggi di `mean_test_score`. In questa fase, la metrica di riferimento per la selezione del modello "vincente" è stata l'**F1-Score**, più affidabile dell'accuratezza semplice in presenza di classi sbilanciate;
- **Migliore configurazione:** I parametri che hanno garantito la miglior capacità predittiva sono stati esportati nel file JSON `iperparametri/migliori/rf_best_params.json`, permettendo il ricaricamento istantaneo del modello ottimizzato.

### 3.2.2.4. Analisi dell'Overfitting e Risultati Ottenuti

L'analisi del Random Forest ha confermato prestazioni nettamente superiori alla Regressione Logistica, dimostrando come la struttura non lineare a foresta riesca a isolare meglio i pattern che conducono alla vittoria (ad esempio, l'interazione tra nazionalità storicamente dominanti e specifiche fasce d'età).

Per mitigare l'**overfitting**, tipico dei modelli basati su alberi, si è agito sulla regolarizzazione limitando la `max_depth`. Il monitoraggio della varianza tra i fold della Cross-Validation ha confermato che il modello non ha semplicemente "memorizzato" i campioni, ma ha estratto regole decisionali generalizzabili.

Dalla procedura di validazione sono emersi i seguenti valori medi per il Random Forest:

MODELLO	METRICA	MEDIA (CV)	DEV. STANDARD ( $\pm$ )
Random Forest	Accuracy	73.91%	0.12%
	Precision	34.36%	0.08%
	Recall	76.62%	0.32%
	F1	47.44%	0.03%

## 3.3. Conclusioni

Il confronto tra le metriche di performance evidenzia una netta superiorità del modello **Random Forest**, che raggiunge un'accuratezza media del **73.91%** e un F1-Score del **47.44%**, contro il **54.97%** di accuratezza e il debole **27.15%** di F1-Score della Regressione Logistica.

Questo divario (particolarmente evidente nell'F1-Score) conferma che la natura del problema predittivo olimpico è intrinsecamente non lineare: mentre il modello lineare fallisce nel separare le classi a causa dello sbilanciamento del dataset, l'algoritmo ensemble basato su alberi decisionali riesce a catturare le interazioni complesse tra le variabili (Nazione, Sport, Genere), riducendo drasticamente i falsi positivi.

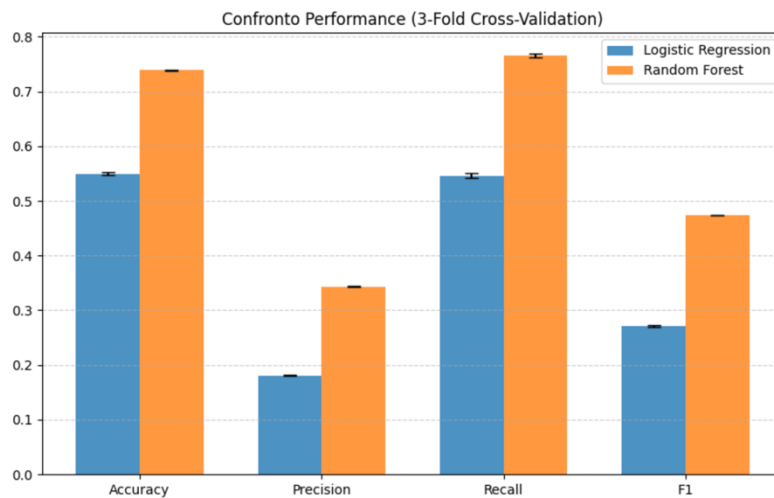
La stabilità dei risultati, certificata da una deviazione standard estremamente contenuta nei fold della Cross-Validation, fornisce una base statistica solida e affidabile per alimentare il successivo ragionamento logico del sistema esperto.

## 3.4. Grafici

Durante la fase di valutazione dei modelli, sono stati generati dei grafici per analizzare comparativamente le prestazioni e la capacità di generalizzazione degli algoritmi sviluppati.

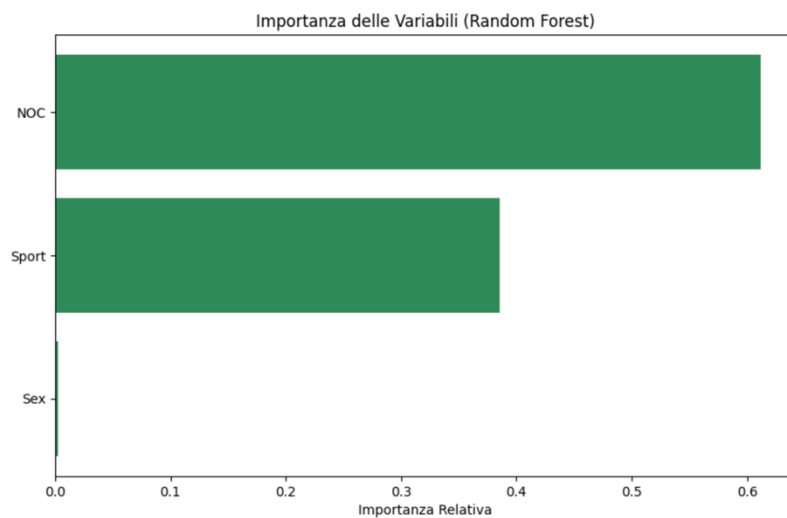
Il grafico principale mostra il confronto tra Logistic Regression e Random Forest basandosi sulle quattro metriche fondamentali della classificazione: Accuracy, Precision, Recall e F1-Score.

L'uso di un grafico a barre con error bars (barre di errore) permette di visualizzare non solo il valore medio delle performance, ma anche la loro stabilità attraverso i diversi fold della validazione incrociata.



Per il modello Random Forest è stato generato un grafico specifico relativo alla **Feature Importance**. Questo strumento è cruciale per la "spiegabilità" dell'IA (Explainable AI), in quanto permette di identificare quali variabili (Sesso, NOC, Sport) abbiano avuto il peso maggiore nel processo decisionale della foresta.

Grazie a questa visualizzazione, è possibile verificare se il modello stia basando le sue predizioni su fattori coerenti con il dominio olimpico (ad esempio, l'influenza storica di una nazione in una determinata disciplina).



Tutti i grafici prodotti vengono generati automaticamente e salvati all'interno della cartella `grafici/` per garantirne la consultazione e la portabilità nella fase di documentazione. I file principali includono:

- `confronto_modelli_cv.png`: Visualizzazione comparativa delle metriche di validazione.
- `feature_importance_rf.png`: Analisi del peso delle variabili nel modello Random Forest

## 4. APPENDIMENTO PROBABILISTICO

L'integrazione dell'approccio probabilistico è finalizzata alla gestione dell'**incertezza** nella previsione del successo olimpico, fornendo un grado di confidenza numerico essenziale per il ragionamento logico della Knowledge Base.

### 4.1. Scelte progettuali e Funzionamento

Per la modellazione è stato adottato il classificatore **Multinomial Naive Bayes**. Sebbene la variante Gaussian sia comune, la natura del dataset – composto da variabili categoriche (NOC, Sport) trasformate in frequenze di occorrenza – ha reso l'approccio multinomiale tecnicamente superiore.

Questa scelta permette di calcolare la verosimiglianza basandosi sulla **densità storica degli eventi**, evitando assunzioni di distribuzione normale non congrue con i dati olimpici.

L'output finale non viene interpretato come un verdetto assoluto, ma come un **grado di confidenza statistica** che alimenta la logica sfocata della Knowledge Base Prolog, permettendo di distinguere tra "Leader storici" e "Outsider".

### 4.2. Iperparametri

La configurazione del modello è stata raffinata tramite **Grid Search** integrata con una **3-fold Cross-Validation**. Per garantire la stabilità statistica. Le decisioni tecniche derivanti dalla sperimentazione sono:

- **Parametro  $\alpha$** : la ricerca esaustiva ha evidenziato che un valore di Smoothing estremamente piccolo massimizza la capacità predittiva (Accuratezza CV ~18.5%). Questa scelta indica che il sistema performa meglio mantenendo una **forte aderenza alle frequenze** originali dei dati storici, pur garantendo una probabilità minima residua per gestire combinazioni di feature mai osservate.
- **fit\_prior**: si è scelto di utilizzare le frequenze reali delle medaglie nel dataset anziché una distribuzione uniforme. Questa decisione riflette la natura sbilanciata del dominio olimpico, dove il successo non è distribuito equamente tra tutte le nazioni.

La scelta di una **3-fold Cross-Validation** è stata dettata dalla necessità di mantenere una dimensione dei fold statisticamente significativa, dato il filtraggio applicato alle nazioni con bassa frequenza di medaglie. Suddividendo i dati in tre sottogruppi, il modello è stato addestrato e testato ciclicamente su porzioni diverse del dataset, assicurando che la combinazione di iperparametri selezionata fosse la più stabile e non dipendente da una specifica distribuzione casuale dei dati.

#### 4.2.1. Creazione della Tabella dei Risultati

Il sistema automatizza il salvataggio dei dati prodotti dalla Grid Search secondo due modalità:

- **Analisi Comparativa** (grid\_search\_nb.csv): Tutti i risultati dei test condotti durante la Cross-Validation sono esportati in formato tabellare. Questo file contiene il log dettagliato delle performance (mean test score) per ogni combinazione di  $\alpha$  e fit\_prior, permettendo un'analisi a posteriori sulla sensibilità del modello rispetto alle variazioni degli iperparametri.
- **Configurazione Ottimale** (nb\_best\_params.json): La combinazione di iperparametri che ha massimizzato l'accuratezza viene isolata e salvata in formato JSON. Questa scelta garantisce che il modulo di inferenza (il sistema finale) possa ricaricare istantaneamente la configurazione vincente senza dover ripetere la fase di addestramento, assicurando la persistenza della "conoscenza" appresa dal classificatore.

### 4.3. Gestione dell'Incertezza e Probabilità

L'adozione del modello Multinomial Naive Bayes permette di quantificare l'**incertezza** epistemologica derivante dalla conoscenza parziale del dominio olimpico, fornendo una stima probabilistica continua anziché un verdetto binario.

Il sistema interpreta la distribuzione di probabilità come un **grafo di confidenza**:

- **Alta confidenza** ( $P \geq 0.8$ ): indica scenari caratterizzati da una forte ricorrenza statistica e un'evidenza storica consolidata (es. nazioni leader in discipline specifiche). In questi casi, l'evidenza empirica domina il processo decisionale.
- **Incertezza elevata** ( $P \approx 0.5$ ): Segnala situazioni di ambiguità predittiva, spesso dovute alla scarsità di dati nel training set o all'elevata competitività (varianza) della disciplina sportiva.

Questa informazione alimenta la Knowledge Base Prolog per l'**arricchimento semantico**: il sistema ibrido non si limita al dato numerico, ma usa regole logiche per "promuovere" una probabilità media a verdetto positivo se l'atleta appartiene a un **élite storica** (es. scherma per l'Italia), passando così da una semplice previsione a un **verdetto argomentato**.

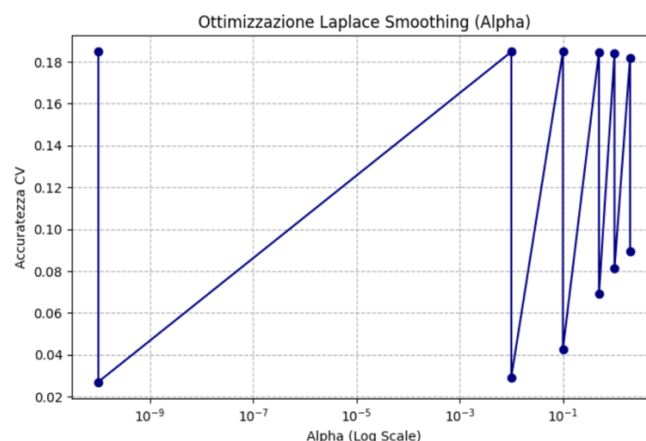
### 4.4. Valutazione del Modello

La valutazione del modello Multinomial Naive Bayes è stata condotta tramite 3-Fold Cross-Validation, analizzando la risposta del sistema alle diverse configurazioni di *smoothing* per garantire la generalizzazione su dati non visti.

- **Analisi delle Performance**: il modello si stabilizza su un'Accuratezza CV massima di **~0.185 (18.5%)**. nonostante il valore numerico possa apparire basso, esso è ritenuto congruo data la complessità del dominio e il forte sbilanciamento delle classi, dove i non-medagliati superano drasticamente i vincitori.
- **Ottimizzazione del parametro Alpha**: la Grid Search ha identificato come valore ottimale  $1e^{-10}$ . questa scelta tecnica indica che il modello massimizza la capacità predittiva mantenendo una forte **aderenza alle frequenze originali** dei dati storici, minimizzando l'intervento della regolarizzazione.
- **Stabilità del modello**: la tenuta dell'accuratezza per valori di Alpha estremamente piccoli conferma che la distribuzione delle medaglia segue logiche frequenziste ben definite che il classificatore riesce a mappare correttamente.

### 4.5. Grafici

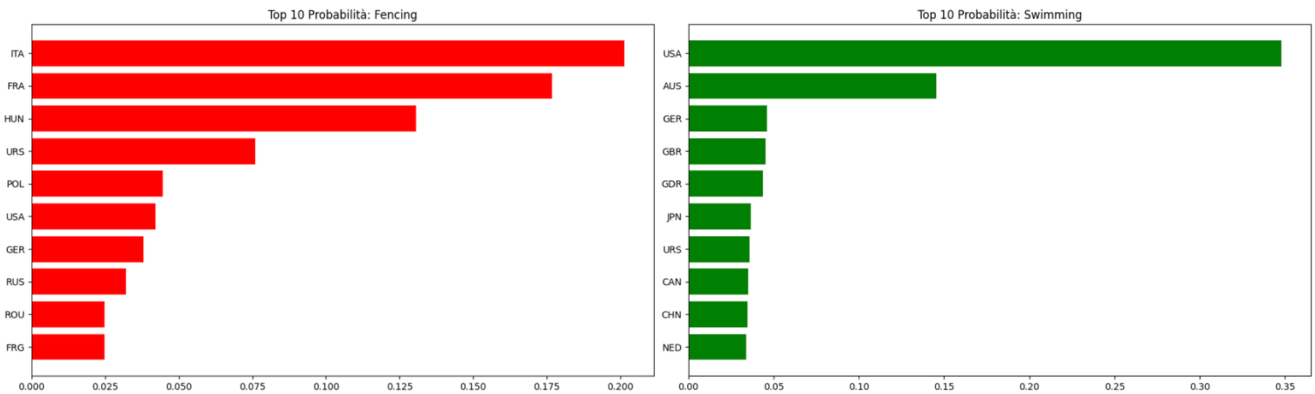
Il codice genera un grafico per avere una visione più chiara. Tale grafico è salvato nella cartella `grafici`.



Il grafico `ottimizzazione_nb.png` mostra dei "salti": questo succede perché il modello è molto sensibile ai piccoli cambiamenti dei dati tra un fold e l'altro della Cross-Validation.

All'interno della cartella possiamo trovare anche due grafici esemplificativi. I grafici `dominanza_swimming.png` e `dominanza_fencing.png` permettono di osservare visivamente la dominanza statistica calcolata dal modello su due specifiche discipline olimpiche.

Questi grafici mostrano la distribuzione delle probabilità di vittoria per le nazioni coinvolte, rendendo immediatamente percepibile il divario tra i vari competitor:



## 5. RAPPRESENTAZIONE DELLA CONOSCENZA – KB

La **Base di Conoscenza** (KB) costituisce il modulo di **ragionamento logico** del sistema, progettato per fornire una valutazione qualitativa e contestuale dei risultati prodotti dal modello di Machine Learning. Sviluppata in linguaggio Prolog, la KB non si limita a riflettere l'output statistico, ma lo integra con la conoscenza di dominio (tradizione storica e fluidità del medagliere), permettendo di generare un agente decisionale ibrido capace che emettere consigli strategici (*Olympic Advice*).

L'obiettivo della KB è agire come un **Sistema Esperto** per:

- **Gestire l'incertezza epistemologica:** raffinare le previsioni numeriche nei casi di probabilità intermedia, dove il solo dato statistico risulterebbe ambiguo;
- **Pondere la competitività:** valutare se una probabilità del ML è sufficiente per un exploit in base alla “chiusura” storica del settore sportivo;
- Fornire spiegabilità: superare l'approccio “black box” del ML generando motivazioni logiche che giustificano il verdetto finale.

### 5.1. Struttura della Kb

La KB è strutturata in modo modulare per superare il limite del semplice database statico, introducendo livelli di astrazione intermedia:

- **Fatti:** rappresentano informazioni di base predefinite, come le nazioni leader in specifiche discipline e le relative percentuali di dominanza storica;
- **Regole:** definiscono la logica inferenziale utilizzata per mediare l'incertezza del Machine Learning, classificare la competitività del settore sportivo fornire un verdetto finale argomentato.

#### 5.1.1. Fatti

I fatti nella KB rappresentano le “**Eccellenze Storiche**” (`historical_elite`), ovvero nazioni che vantano una tradizione di successo consolidata in determinati sport, arricchite dall'indice di Percentuale di Dominanza calcolato analiticamente. I fatti nel sistema sono:

<code>historical_elite('ITA', 'Fencing', 20.12).</code>	<code>%Italia nella Scherma</code>
<code>historical_elite('USA', 'Swimming', 34.80).</code>	<code>%USA nel Nuoto</code>
<code>historical_elite('USA', 'Golf', 70.69).33</code>	<code>%USA nel Golf</code>
<code>historical_elite('CHN', 'Table Tennis', 45.61).</code>	<code>%Cina nel Tennis Tavolo</code>
<code>historical_elite('JAM', 'Athletics', 30.00).</code>	<code>%Jamaica nell'Atletica</code>
<code>historical_elite('JPN', 'Judo', 38.00).</code>	<code>%Giappone nel Judo</code>
<code>historical_elite('BRA', 'Football', 25.00).</code>	<code>%Brasile nel Calcio</code>

#### 5.1.2. Regole

Le regole della KB definiscono il modo in cui i fatti vengono combinati con la probabilità continua (0.0 – 1.0) ricevuta dal modello Random Forest per generare raccomandazioni contestualizzate. Le regole implementate sono:

- **Determinazione del tipo di settore:** la fluidità di una disciplina viene classificata in base alla dominanza del leader storico

```
/** sector_type(+Sport, -Type)
    Definisce la fluidità di un settore in base alla dominanza del leader.
*/
sector_type(Sport, closed) :-
    historical_elite(_, Sport, Dominance), Dominance > 40.0, !.
sector_type(_, open).
```

- **Identificazione delle Superpotenze:** verifica se una nazione possiede una struttura d'eccellenza in più settori (almeno due), garantendo un supporto sistemico all'atleta

```
/** is_superpower(+NOC)
    Vero se la nazione possiede una struttura d'élite in più settori.
*/
is_superpower(NOC) :-
    findall(S, historical_elite(NOC, S, _), L),
    length(L, N), N >= 2.
```

- **Generazione del verdetto (Olympic Advice):** incrocia la probabilità statistica con la logica simbolica per distinguere tra successi attesi ed exploit

```
/** olympic_advice(+Probability, +NOC, +Sport, -Advice)
    Predicato principale che genera il verdetto finale.
*/

% Caso A: Eccellenza Storica Confermata (ML e KB concordano)
olympic_advice(Prob, NOC, Sport, Advice) :-
    Prob >= 0.70,
    historical_elite(NOC, Sport, Dom),
    atomic_list_concat(['SUCCESSO ATTESO: Dominio storico confermato (Dominanza
leader: ', Dom, '%).'], Advice), !.

% Caso B: Tradizione Resiliente (La storia compensa dati ML incerti)
% Caso C: Nuova Potenza (Exploit in ascesa in settori fluidi)
% Caso D: Exploit Difficile (Dato ML alto contrastato da egemonia storica altrui)
% Caso E: Scommessa (Atleta promettente senza radici sistemiche)
% Caso F: Fallback (Bassa probabilità e assenza di supporto storico)
```

- **Generazione delle motivazioni:** implementa la trasparenza del sistema fornendo le ragioni logiche dietro ogni consiglio

```
/** explain_verdict(+Probability, +NOC, +Sport, -Reasons)
    Genera una lista di stringhe che motivano logicamente la decisione.
*/
explain_verdict(Prob, NOC, Sport, Reasons) :-
    findall(M, (
        (Prob > 0.70 -> M = 'Solidità statistica del modello ML');
        (historical_elite(NOC, Sport, _) -> M = 'Forte tradizione storica
nazionale');
        (is_superpower(NOC) -> M = 'Appartenenza a nazione leader (Superpower)');
        (sector_type(Sport, open), Prob >= 0.45 -> M = 'Settore competitivo aperto a
nuovi talenti');
        (Prob < 0.45 -> M = 'Performance storiche e recenti insufficienti')
    ), Reasons),
    (Reasons = [] -> Reasons = ['Analisi basata su parametri standard'] ; true).
```

## 5.2. Ragionamento Inferenziale nella KB

Il processo inferenziale avviene tramite **unificazione e pattern matching**. Quando il modulo Python interroga la KB, non invia un'etichetta rigida, ma la probabilità esatta calcolata dal Random Forest. Il sistema Prolog:

- **Ponderazione delle soglie:** confronta la probabilità con le soglie logiche definite nelle clausole `olympic_advice` per determinare il grado di confidenza statistica.
- **Analisi contestuale del settore:** il sistema valuta la fluidità della disciplina (Settore Chiuso o Aperto) e l'eventuale supporto sistemico dell'atleta se appartenente a una "Superpotenza" olimpica.
- **Valutazione storica:** effettua una ricerca nella base dei fatti per verificare se la coppia (Nazione, Sport) appartenga all'élite storica del settore, permettendo di "promuovere" o "mitigare" il verdetto del Machine Learning.
- **Ottimizzazione tramite Cut (!):** utilizza l'operatore "cut" per bloccare il backtracking una volta individuata la categoria corretta, ottimizzando la velocità di risposta.

Questo trasforma una soft-label numerica in una **conoscenza esplicativa**, rendendo il sistema un vero supporto alle decisioni (DSS).

### 5.3. Conclusioni

Il sistema ibrido sviluppato per la previsione del successo olimpico e la generazione di consigli strategici ha dimostrato di essere un'implementazione robusta ed efficace. L'integrazione tra la potenza predittiva del Machine Learning (Random Forest e Multinomial Naive Bayes) e il rigore logico della Base di Conoscenza (KB) in Prolog permette di ottenere non solo una stima probabilistica, ma anche una validazione contestuale basata su criteri storici, tecnici e sulla fluidità dei settori sportivi.

Attualmente, il sistema elabora parametri quali il genere, la nazione di appartenenza (NOC) e la disciplina sportiva. L'originalità del modello risiede nella sua capacità di pesare l'incertezza statistica attraverso la Percentuale di Dominanza dei leader storici, distinguendo tra sport "chiusi" (egemonici) e "aperti" (fluidi) per fornire una spiegabilità (XAI) ai propri verdetti.

Tuttavia, il progetto presenta diverse direzioni evolutive per migliorarne l'accuratezza:

- **Espansione del modello predittivo:** In futuro, si potrebbe arricchire il dataset includendo variabili biometriche specifiche (come l'età dell'atleta o l'esperienza pregressa) e variabili macroeconomiche delle nazioni (come il PIL o gli investimenti nello sport), parametri che la letteratura scientifica indica come fortemente correlati al successo olimpico.
- **Ottimizzazione della Base di Conoscenza:** Si prospetta l'ampliamento della KB includendo lo storico dettagliato degli scontri diretti (head-to-head) o l'analisi delle prestazioni nelle competizioni mondiali preolimpiche.
- **Logica dell'Incertezza:** L'introduzione di logiche basate sull'incertezza, come la logica fuzzy, all'interno di Prolog permetterebbe di gestire con maggiore granularità le sfumature dei consigli forniti, rendendo il sistema un supporto decisionale (DSS) ancora più sofisticato per allenatori e federazioni sportive.

### 5.4. Esempio

SISTEMA DI PREDIZIONE OLIMPICA
Sesso (M/F): F Codice Nazione (es. ITA, USA): ITA Sport (es. Basketball): Fencing
RISULTATI PREVISIONE: 76.40% di probabilità podio.
ANALISI DI SETTORE (FENCING): Leader storico: ITA (Dominanza: 20.12%)
CONSIGLIO DELL'ESPERTO: SUCCESSO ATTESO: Dominio storico confermato (Dominanza leader: 20.12%)
MOTIVAZIONI LOGICHE: • Solidità statistica del modello ML

## 6. ANALISI MEDAGLIERE

Per arricchire la base di conoscenza del sistema e fornire un contesto storico ai modelli di Machine Learning, è stata implementata una procedura di analisi dei dati denominata `genera_report_leader()` e situata in `analisi_medagliere.py`. Questa funzione ha lo scopo di individuare la nazione "leader" per ogni disciplina sportiva, calcolando non solo il numero assoluto di medaglie vinte, ma anche l'indice di **Percentuale di Dominanza**.

Il processo di elaborazione si articola nelle seguenti fasi:

- **Filtro e Selezione:** Il sistema carica il dataset olimpico e filtra esclusivamente le riga corrispondenti a una medaglia vinta (Gold, Silver, Bronze), escludendo gli atleti che non sono saliti sul podio.
- **Aggregazione dei Dati:** Tramite la libreria `Pandas`, i dati vengono raggruppati per combinazione di Sport e NOC (nazione). Questo permette di contare quante medaglie ogni paese ha vinto storicamente in una specifica disciplina.
- **Individuazione del Leader:** Per ogni sport, il sistema identifica la nazione con il numero massimo di medaglie utilizzando la funzione `idxmax()`. Questo identifica chi, storicamente, detiene il primato nello sport in questione.
- **Calcolo della Dominanza:** Viene introdotta una metrica specifica, la Percentuale di Dominanza, calcolata come:

$$\text{Percentuale di Dominanza} = \left( \frac{\text{Totale medaglie assegnate nello Sport}}{\text{Medaglie vinte dalla Nazione Leader}} \right) \times 100$$

Questa metrica è fondamentale per distinguere tra sport in cui esiste un leader assoluto (alta dominanza) e sport caratterizzati da un maggiore equilibrio competitivo tra le nazioni.

I risultati di questa analisi vengono infine esportati in un file dedicato (`report_leader_tesi.csv`), che funge da riferimento per convalidare le previsioni probabilistiche ottenute tramite il modello Multinomial Naive Bayes.

```
=====
ANALISI MEDAGLIERE: LEADER STORICI PER SPORT
=====

Analisi completata su 71 discipline sportive.

SPORT | LEADER | MEDAGLIE | DOMINANZA
-----|-----|-----|-----
Roque | USA | 3 | 100.0%
Aeronautics | SUI | 1 | 100.0%
Basque Pelota | ESP | 2 | 100.0%
Croquet | FRA | 8 | 100.0%
Racquets | GBR | 10 | 100.0%
Cricket | GBR | 21 | 87.5%
Motorboating | GBR | 6 | 85.71%
Golf | USA | 41 | 70.69%
Jeu De Paume | GBR | 2 | 66.67%
Lacrosse | CAN | 36 | 60.0%
Alpinism | GER | 2 | 50.0%
Table Tennis | CHN | 104 | 45.61%
Polo | GBR | 30 | 44.78%
Trampoline Gymnastics | CHN | 5 | 41.67%
Ice Hockey | USA | 11 | 40.74%

[OK] File 'report_leader_tesi.csv' generato correttamente!
[INFO] Il report include i dati storici da Atene 1896 a Parigi 2024.
```

(Si precisa che, all'interno della presente documentazione, gli screenshot relativi all'output del terminale mostrano un'analisi limitata a 15 nazioni a scopo puramente illustrativo. Questa scelta è stata dettata da necessità di spazio e leggibilità editoriale, per permettere una visione chiara della struttura del report. Tuttavia, il sistema esegue l'elaborazione completa sull'intero dataset olimpico).

## 7. DIPENDENZE

Per garantire il corretto funzionamento del sistema di previsione e analisi delle medaglie olimpiche, è necessario installare le seguenti librerie Python:

- `pandas`: Fondamentale per la manipolazione del dataset olimpico, la gestione dei file CSV e la creazione del report dei leader storici.
- `scikit-learn`: Utilizzata per l'implementazione dei modelli di Machine Learning (Multinomial Naive Bayes e Random Forest), la gestione della Grid Search e il calcolo delle metriche di valutazione.
- `numpy`: Necessaria per le operazioni vettoriali sulle probabilità e la gestione delle matrici di dati durante l'addestramento.
- `matplotlib/seaborn`: Librerie dedicate alla generazione dei grafici di dominanza statistica e alla visualizzazione delle curve di apprendimento.
- `joblib`: Utilizzata per la serializzazione dei modelli addestrati (salvataggio in formato `.pkl`), permettendone il caricamento rapido senza dover rieseguire l'apprendimento.
- `pyswip`: Il bridge fondamentale che permette a Python di comunicare con SWI-Prolog, inviando i risultati del Machine Learning alla Base di Conoscenza sotto forma di fatti logici.

### 7.1. Configurazione dell'ambiente

1. Installazione di Python dal sito ufficiale <https://www.python.org/>
2. Installazione di SWI-Prolog da <https://www.swi-prolog.org/>
3. Installazione delle librerie necessarie con il comando: `pip install nomelibreria` oppure `pip install -r requirements.txt`

## 8. DOCUMENTAZIONE

All'interno del progetto ho inserito la documentazione di Python: tale documentazione è stata generata automaticamente tramite lo strumento `Sphinx`, che estrae i metadati e le descrizioni direttamente dalle `docstrings` presenti nel codice sorgente.

La documentazione è archiviata nella cartella `docs/`. Per visualizzare i manuali tecnici, è necessario navigare nella sottocartella `build/html/` e aprire il file `index.html`. Il file offre un'interfaccia ipertestuale navigabile che illustra le classi, i metodi e le funzioni implementate nei moduli `main.py` e `analisi_medagliere.py`, con i relativi parametri di input e output.

## 9. RIFERIMENTI BIBLIOGRAFICI

- <https://www.kaggle.com/>
- David L. Poole, Alan K. Mackworth - Artificial Intelligence Foundations of Computational Agents-Cambridge University Press (2010)