



GoalGuru

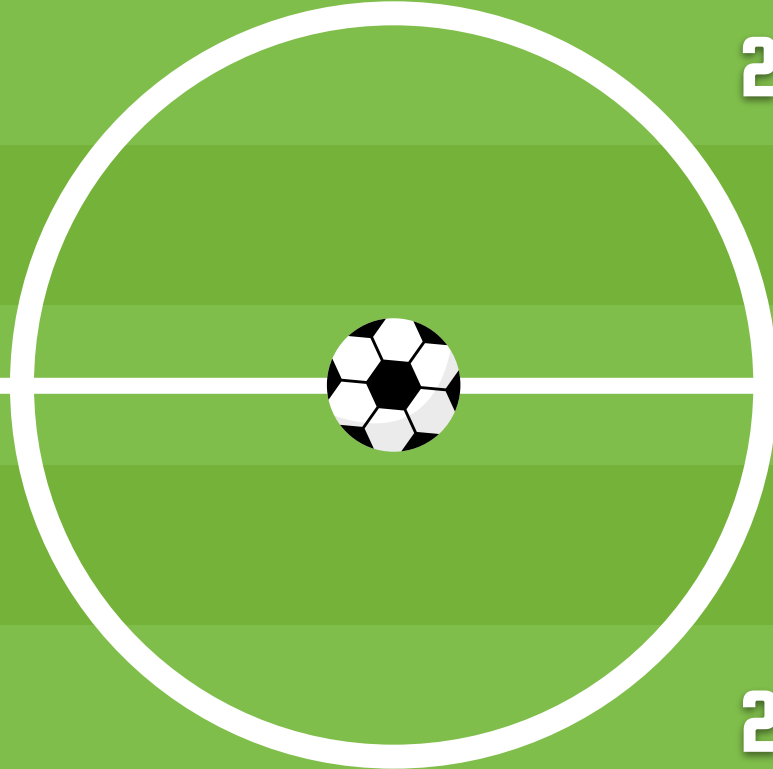
Choaib Goumri - 0512118390





1 INTRODUZIONE

2 MODELLO CRISP - DM



2.1 Business Understanding

2.2 Data Understanding

2.3 Data Preparation

2.4 Data Modeling

2.5 Evaluation

2.6 Deployment

01

INTRODUZIONE



INTRODUZIONE

Negli ultimi anni, l'applicazione **dell'intelligenza artificiale** nel mondo dello sport ha subito **un'evoluzione significativa**, rivoluzionando il modo in cui analisti, appassionati e professionisti interpretano il gioco. **L'analisi predittiva** delle partite di calcio, un campo che fino a poco tempo fa era dominato da **intuizioni soggettive** e **modelli statistici tradizionali**, sta ora beneficiando dei progressi dell'apprendimento automatico. Il **progetto GoalGuru** nasce con l'obiettivo di sfruttare tecniche di **Machine Learning** per fornire previsioni affidabili sui risultati delle partite di calcio, analizzando dati storici e individuando schemi nascosti che possano influenzare gli esiti degli incontri.

Per progettare il nostro modello di AI, abbiamo seguito la metodologia **CRISP-DM**, strutturato e affidabile per progetti di data mining. Questo framework ci ha guidato attraverso fasi chiare: dalla **comprensione del business** e dei **dati alla preparazione, modellazione, valutazione e implementazione**. Grazie a CRISP-DM, abbiamo ottimizzato il processo, allineato obiettivi e soluzioni tecniche, e garantito un modello efficace e scalabile.

2.1 Business Understanding



Business Understanding

Il progetto mira a creare un modello per prevedere i risultati delle partite di calcio, sfruttando il **machine learning** e dati storici per ottenere previsioni accurate.

Gli obiettivi?

- Migliorare l'**accuratezza** delle previsioni
- Offrire agli utenti **informazioni per scommesse** o analisi sportive
- Aumentare l'engagement spingendoli a seguire più partite e campionati
- **Fidelizzarli** con un servizio di alta qualità che superi le aspettative.



P.E.A.S.

- **Performance:** precisione delle previsioni dei risultati delle partite di calcio.
- **Environment:** contesto delle partite di calcio.
- **Actuators:** l'agente agisce sull'ambiente fornendo previsioni sui risultati delle partite di calcio.
- **Sensors:** l'agente percepisce le informazioni attraverso l'accesso diretto ai dati storici delle partite.

Environment

- **Completamente osservabile:** il dataset fornisce tutte le informazioni necessarie per prendere decisioni
- **Stocastico:** i risultati sono influenzati da molteplici fattori imprevedibili
- **Episodico:** la predizione per una partita non influenza le altre
- **Statico:** l'ambiente non cambia mentre l'agente opera
- **Discreto:** Le variabili sono misurate in valori distinti e ben definiti
- **Ad Agente Singolo:** vi sarà un unico agente ad operare



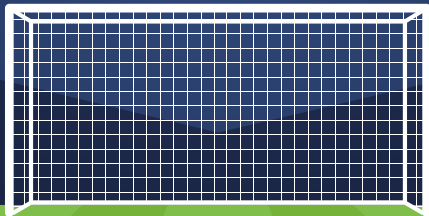
2.2 Data Understanding



DATASET

Questo dataset contiene **4788** partite di calcio, con dettagli su **statistiche di gioco**, risultati e contesto delle competizioni. Ogni partita è rappresentata con informazioni sulla data, l'orario, la competizione, la sede (casa o trasferta) e il risultato. Le statistiche includono gol fatti e subiti, expected goals (xG), possesso palla, tiri totali, distanza media dei tiri, calci piazzati e rigori. Sono presenti anche dati aggiuntivi come il capitano della squadra e l'arbitro.

Ogni riga collega una squadra al proprio avversario, con **informazioni utili per analisi comparative** o studi sulle performance stagionali. Sebbene la maggior parte delle colonne sia completa, **alcune**, come gli spettatori, **presentano valori mancanti**, e una colonna (notes) è completamente vuota. **Nel complesso**, il dataset offre una **visione dettagliata e versatile**, adatta per analisi statistiche, esplorazioni tattiche o modelli predittivi legati al calcio.



2.3 Data Preparation

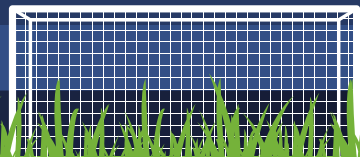


PREPROCESSING

Il preprocessing di un dataset complesso implica una serie di operazioni per **trasformare i dati grezzi in un formato utilizzabile** per l'analisi. Dopo un'esplorazione iniziale per comprendere la struttura e individuare anomalie, sono state **rimosse informazioni superflue e ottimizzati i tipi di dato** per ridurre l'occupazione di memoria. **Nuove colonne** sono state aggiunte per estrarre informazioni derivate, mentre dati mancanti o duplicati sono stati gestiti per garantire coerenza. Operazioni come normalizzazioni, calcoli di medie mobili e categorizzazioni hanno ulteriormente arricchito il dataset, rendendolo pronto per analisi più avanzate o applicazioni di machine learning.



#Colonne prima	#Colonne dopo
28	24



2.4 Data Modeling



KNN

Il K-Nearest Neighbors (KNN) è un algoritmo di machine learning **supervisionato** utilizzato per problemi di classificazione e regressione. L'idea alla base del KNN è semplice: dato un nuovo punto dati, l'algoritmo **cerca i K punti più vicini** nel dataset di training e assegna al nuovo punto la classe più frequente tra i vicini (per la classificazione) o la media dei valori dei vicini (per la regressione).

Ogni modello viene creato con una **combinazione** specifica di **iperparametri** (n_neighbors, weights, metric) dalla griglia definita.

- **Vicini:** Il parametro n_neighbors controlla quanti vicini considerare per classificare un'istanza.
- **Pesatura:** L'opzione weights determina se tutti i vicini contribuiscono ugualmente alla classificazione (uniform) o se i vicini più vicini hanno più peso (distance).
- **Distanza:** Il parametro metric specifica la metrica di distanza utilizzata (euclidea o manhattan).



Limiti del KNN

Nonostante la semplicità e l'efficacia in problemi di piccole dimensioni, il KNN presenta alcuni limiti:

Ogni modello viene creato con una **combinazione** specifica **di iperparametri** (`n_neighbors`, `weights`, `metric`) dalla griglia definita.

- **Sensibilità alla scelta di k:** Un valore di `k` troppo piccolo rende il modello sensibile al rumore, mentre un valore troppo grande riduce la capacità di catturare le variazioni locali.
- **Scalabilità:** Il KNN diventa inefficiente con dataset di grandi dimensioni, poiché ogni nuova predizione richiede il calcolo della distanza con tutti i punti di training.
- **Difficoltà con dati complessi:** In problemi con distribuzioni più intricate, il KNN può avere difficoltà a generalizzare



Superamento dei limiti con una Rete Neurale

- Abbiamo eseguito **un'analisi dettagliata delle performance** dei modelli nella fase di valutazione, concentrandoci su **metriche chiave come accuratezza, tempo di esecuzione e capacità di generalizzazione**.
- Dai risultati ottenuti, **abbiamo osservato che il KNN**, pur essendo **efficace in scenari con dataset di piccole dimensioni** e distribuzioni relativamente semplici, **non riusciva a catturare adeguatamente pattern complessi nei dati**, risultando meno performante rispetto ad approcci più avanzati.
- Questo ci ha portato a **considerare metodi in grado di superare queste limitazioni**, valutando le loro prestazioni su diversi set di dati e individuando una soluzione che garantisse un migliore equilibrio tra accuratezza, efficienza computazionale e capacità di adattamento alle caratteristiche del problema.



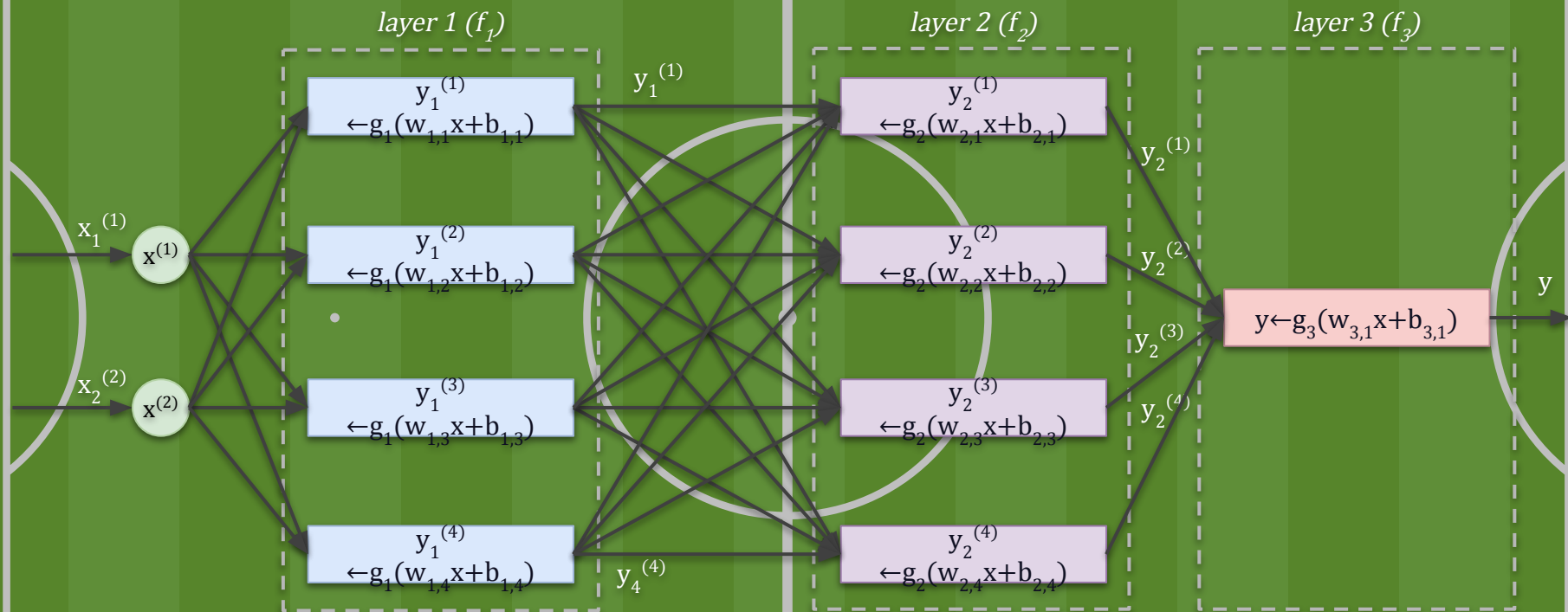
RETE NEURALE

La rete neurale è composta da due strati densi (dense layers) e da un layer di dropout. Ogni strato denso è completamente connesso.

- **Input:** Il modello riceve in ingresso variabili numeriche e categoriche, con una dimensione determinata dal numero di variabili nel dataset.
- **Strato nascosto 1:** Un primo strato denso con un numero di neuroni definito da iperparametri.
- **Strato nascosto 2:** Un secondo strato denso, anch'esso con neuroni regolati da iperparametri.
- **Dropout:** Un livello di dropout (10%) applicato dopo il secondo strato per evitare overfitting, disattivando casualmente alcuni neuroni durante l'addestramento.
- **Output:** Uno strato denso con 3 neuroni (vittoria, pareggio, sconfitta) e attivazione softmax, che restituisce le probabilità per ciascuna classe, sommate a 1.



ESEMPIO RETE NEURALE



IPERPARAMETRI

Sono parametri che **controllano il processo di apprendimento** del modello, ma non vengono appresi dai dati. Servono per **ottimizzare l'apprendimento**, fornendo al modello gli strumenti migliori per imparare.

- **learning_rate**: Determina quanto velocemente i pesi vengono aggiornati. Un valore troppo alto può causare instabilità, uno troppo basso può rallentare l'apprendimento.
- **epochs**: Il numero di volte che l'intero dataset viene passato nella rete. Troppo poche epoche possono portare a underfitting, troppe a overfitting.
- **neurons_1layer e neurons_2layer**: Il numero di neuroni nei rispettivi strati nascosti. Più neuroni possono aumentare la capacità di apprendimento, ma aumentano il rischio di overfitting.
- **activation_functions**: Funzioni che determinano come i neuroni attivano i segnali. Le più comuni sono:
 - ReLU: Attiva valori positivi, azzera quelli negativi. Veloce e comune.
 - Sigmoid: Mappa valori tra 0 e 1. Utile per classificazione binaria.
 - Tanh: Mappa valori tra -1 e 1.
- **batch_size**: Il numero di esempi elaborati prima di aggiornare i pesi. Batch più grandi rendono l'apprendimento più stabile, ma richiedono più memoria.

Grid Search

Viene utilizzato il grid search per **testare diverse combinazioni di iperparametri**, come il numero di neuroni negli strati nascosti, la funzione di attivazione, il learning rate, il numero di epoche e la dimensione del batch. **Ogni combinazione viene valutata** e il modello con la minore perdita di validazione viene scelto come migliore.

Random Search

Viene utilizzato il grid search per **testare diverse combinazioni di iperparametri**, come il numero di neuroni negli strati nascosti, la funzione di attivazione, il learning rate, il numero di epoche e la dimensione del batch. **Ogni combinazione viene valutata** e il modello con la minore perdita di validazione viene scelto come migliore.



2.5

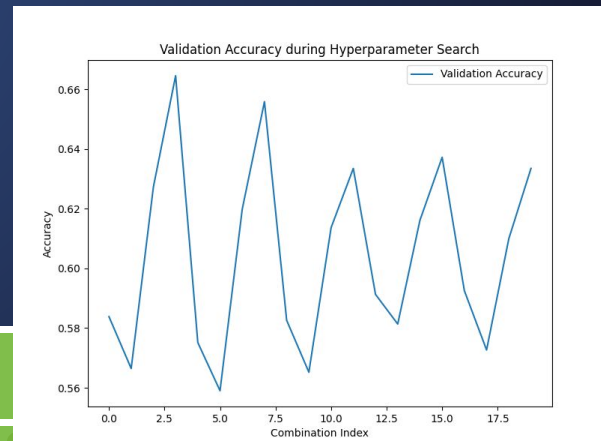
Evaluation



Analisi dell'Accuratezza di validazione KNN

Intervallo di accuratezza: 0.56–0.60, con un massimo di **0.60** alla combinazione **15.0 (indice 17.5)**.

- ◆ **Distribuzione non uniforme:** Solo **4 su 18 combinazioni** superano 0.59 → pochi iperparametri sfruttano il pieno potenziale del modello.
- ◆ **Sensibilità ai parametri:**
 - Learning rate > 0.01 → calo di accuratezza (0.56), segno di instabilità.
 - Regularizzazione moderata ($L2 = 0.001$) → migliora la generalizzazione.
 - ◆ **Limiti intrinseci:** Accuratezza max 60% potrebbe dipendere da **rumore nei dati, feature poco informative o squilibrio di classe**.



Confronto Grid Search vs. Random Search

- Grid Search (GS)
 - ✓ Andamento della loss: da 1.05 → 0.75 in 5 epoche ($\Delta 0.06$ per epoca).
 - ✓ Saturazione dopo 3 epoche ($\Delta < 0.02$ per epoca).
 - ⚠ Costo computazionale: 12h su CPU Intel i7, 16GB RAM.
- Random Search (RS)
 - ✓ Andamento della loss: da 0.75 → 0.68 in 4 epoche ($\Delta 0.12$ per epoca).
 - ✓ Esplora il 30% in meno di combinazioni → più efficiente.
 - ✓ Tempo di training: 6h → riduzione del costo computazionale.
- 🔑 Key Insights
 - ✓ RS converge più velocemente (loss inferiore: 0.68 vs. 0.75).
 - ✓ Maggiore stabilità (varianza inter-epoca ridotta).
 - ✓ Trade-off: GS esplora più a fondo, utile per domini ben definiti,

mentre RS ottimizza il tempo di ricerca.



MIGLIORI IPERPARAMETRI (Grid Search)



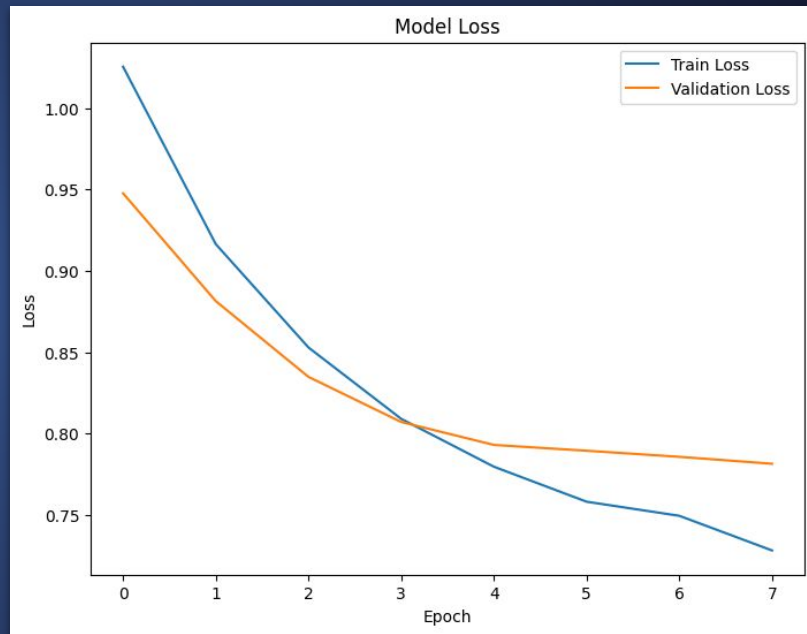
Grid Search

218:44

Random Search



Learning Rate	0.001
Epochs	8
Numero di Neuroni (Layer 1)	55
Numero di Neuroni (Layer 2)	50
Funzione di Attivazione	ReLu
Batch Size	200



2.5

Deployment



Deployment

- ◆ **Obiettivo:** Integrare il modello in un ambiente operativo per inferenze rapide, scalabili e affidabili.
- ◆ **Sfide principali:**
 - **Scalabilità:** Gestione efficiente di richieste crescenti.
 - **Affidabilità:** Disponibilità e resilienza del servizio.
 - **Manutenibilità:** Aggiornamenti senza interruzioni.



Attività del Deployment

- ✓ **Salvataggio dei risultati** → `test_results.txt` documenta **accuratezza, precision, recall, F1-score**.
- ✓ **Analisi errori** → **Matrice di confusione** (`matrice_di_confusione.png`) per ottimizzazione modello.
- ✓ **Esportazione modello** → `best_model.h5` pronto per:
 - **API REST** (Flask, FastAPI, Django).
 - **Cloud Deployment** (AWS, Google AI Platform, Azure).
 - **Edge Computing** (dispositivi locali per bassa latenza).



CONCLUSIONI

Il modello è stato testato, ottenendo una **precisione del 67%** e una **perdita di 78.11%**.

Mentre la **precisione è accettabile**, la **perdita** relativamente **alta indica che il modello commette ancora molti errori**.

Questo suggerisce che ci sia spazio per miglioramenti, ad esempio ottimizzando ulteriormente gli iperparametri o applicando tecniche di regolarizzazione per affinare le prestazioni.

Goal Guru ha sviluppato un **workflow completo di deployment ML**, con un'API operativa e scalabile, pronta per future integrazioni e ottimizzazioni.





GRAZIE!

Choaib Goumri - 0512118390