SAPIENZA UNIVERSITÀ DI ROMA

Facoltà di Lettere e Filosofia

Corso di Laurea in Filosofia e Intelligenza Artificiale

Riconoscimento di Manovre Complesse in Ambienti Videoludici Competitivi

Un approccio basato su Random Forest e Parsing Vettoriale

Gruppo di Lavoro:

Brando Cappucci	Matricola: 2139484
Matteo Maria Martino Remondini	Matricola: 2136808
Lorenzo Stranieri	Matricola: 2153723

Corso di Intelligenza Artificiale 2 - A.A. 2025/2026

Prof.ssa Flavia Monti

2 Febbraio 2026

Indice

Sommario

Il presente progetto affronta la sfida della classificazione supervisionata di manovre acrobatiche (*skillshot*) nel videogioco basato sulla fisica *Rocket League*. A differenza di approcci tradizionali basati su statistiche descrittive semplici, proponiamo una metodologia innovativa che combina un'ingestione dati ottimizzata vettorialmente con un Feature Engineering "comportamentale".

Utilizzando il dataset ufficiale UCI Machine Learning Repository, abbiamo trasformato serie temporali a lunghezza variabile in vettori informativi mediante l'uso di quartili (per la robustezza fisica) e contatori di transizione (*Toggle Count*) per catturare la frenesia degli input. Il modello selezionato, un **Random Forest Classifier** ottimizzato, ha raggiunto un'accuratezza del **86.67%** sul test set, dimostrando una superiorità netta rispetto alla baseline e a modelli alternativi come le Support Vector Machines (SVM).

1 Introduzione e Definizione del Problema

1.1 Il Contesto: Rocket League e l'E-Sport

L'industria degli sport elettronici (e-sport) genera enormi quantità di dati di telemetria che, se analizzati correttamente, possono rivoluzionare il coaching automatizzato e l'analisi delle performance. *Rocket League*, un gioco che ibrida il calcio con veicoli a razzo, rappresenta un caso di studio eccellente per l'Intelligenza Artificiale a causa della sua natura puramente basata sulla fisica.

A livelli competitivi alti, i giocatori non si limitano a guidare, ma eseguono manovre aeree complesse chiamate *skillshot* (es. *Ceiling Shot*, *Air Dribble*). Queste manovre sono definite da sequenze precise di input e stati fisici del veicolo (posizione, velocità, rotazione).

1.2 La Domanda di Ricerca

La sfida principale risiede nella natura dei dati: ogni manovra ha una durata variabile. Un *Air Dribble* può durare 50 frame, mentre un *Power Shot* solo 10. La domanda di ricerca che guida questo lavoro è:

> *"È possibile addestrare un modello di Machine Learning per classificare correttamente una manovra osservando solo i dati grezzi dei sensori, indipendentemente dalla durata dell'esecuzione e dallo stile del giocatore?"*

1.3 Obiettivi del Progetto

- Acquisire e pulire il dataset in modo riproducibile e computazionalmente efficiente.
- Superare il problema della lunghezza variabile delle serie temporali tramite un Feature Engineering mirato.
- Confrontare modelli non lineari (Random Forest vs SVM) per identificare l'architettura più robusta.
- Interpretare i risultati per capire quali "segnali" fisici distinguono una manovra dall'altra.

2 Dataset e Pipeline di Ingestione Dati

2.1 Origine dei Dati

Abbiamo utilizzato il *Rocket League Skillshots Data Set*, disponibile presso l'UCI Machine Learning Repository [1]. Per garantire la riproducibilità scientifica, il nostro codice scarica ed estrae automaticamente i dati dall'endpoint statico ufficiale.

Il dataset contiene 298 sequenze temporali multivariate, suddivise in 7 classi:

1. **Noise** (Rumore/Nessuna manovra)
2. **Ceiling Shot**
3. **Power Shot**
4. **Waving Dash**
5. **Air Dribble**
6. **Front Flick**
7. **Musty Flick**

2.2 Parsing Vettoriale Ottimizzato

Uno degli aspetti tecnici più rilevanti del nostro lavoro è la metodologia di parsing. Il file grezzo `.data` presenta una struttura ibrida (righe di etichetta intervallate a righe di dati) che tradizionalmente richiederebbe cicli iterativi lenti.

Abbiamo invece implementato un approccio **vettoriale** basato sulla libreria *Pandas*. Come mostrato nel seguente estratto del nostro codice, carichiamo l'intero file e utilizziamo maschere booleane e operazioni di somma cumulativa (`cumsum`) per assegnare l'ID della sequenza e la classe.

```
# Identificazione delle righe "Etichetta"
mask_is_label = split_data[1].isna()

# Generazione Sequence_ID tramite somma cumulativa
raw_df['Sequence_ID'] = mask_is_label.cumsum()

# Propagazione della Class_Label (Forward Fill)
raw_df['Class_Label'] = raw_df['temp_label'].ffill()

# Filtro finale: teniamo solo le righe che contengono dati
final_df = split_data[~mask_is_label].copy()
```

Listing 1: Logica di parsing vettoriale (Tratto dalla funzione `load_and_parse_vectorized`)

Questo approccio riduce drasticamente i tempi di elaborazione e prepara il dataset per l'analisi immediata, risultando in un totale di **6892 frame** analizzati.

3 Analisi Esplorativa dei Dati (EDA)

Prima di procedere alla modellazione, abbiamo condotto un'analisi esplorativa per identificare bias e criticità strutturali.

3.1 Distribuzione delle Classi

Come evidenziato nel grafico sottostante, il dataset soffre di un marcato sbilanciamento delle classi. La classe *Power Shot* è sovra-rappresentata (60 campioni), mentre classi complesse come *Ceiling Shot* sono rare (30 campioni).

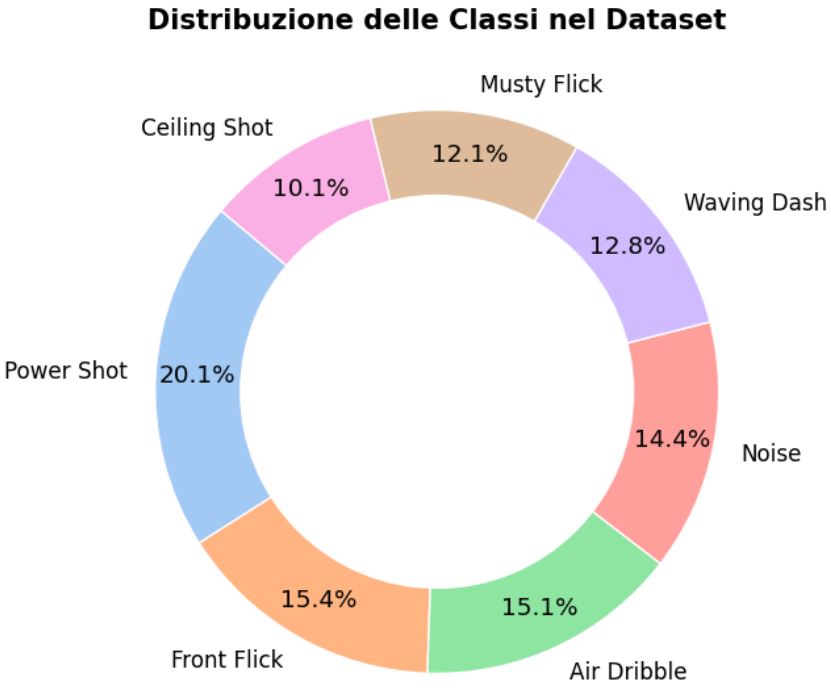

Figura 1: Distribuzione percentuale delle classi nel dataset.

Questa osservazione ha dettato due scelte metodologiche fondamentali:

- L'uso della **Stratified Cross-Validation** per garantire che ogni fold mantenga le stesse proporzioni.
- L'adozione dell'**F1-Score Weighted** come metrica principale, poiché l'Accuracy sarebbe fuorviante.

3.2 Variabilità Temporale

Il secondo aspetto critico è la durata delle manovre. Analizzando la lunghezza delle sequenze per ogni classe, emerge una variabilità significativa.

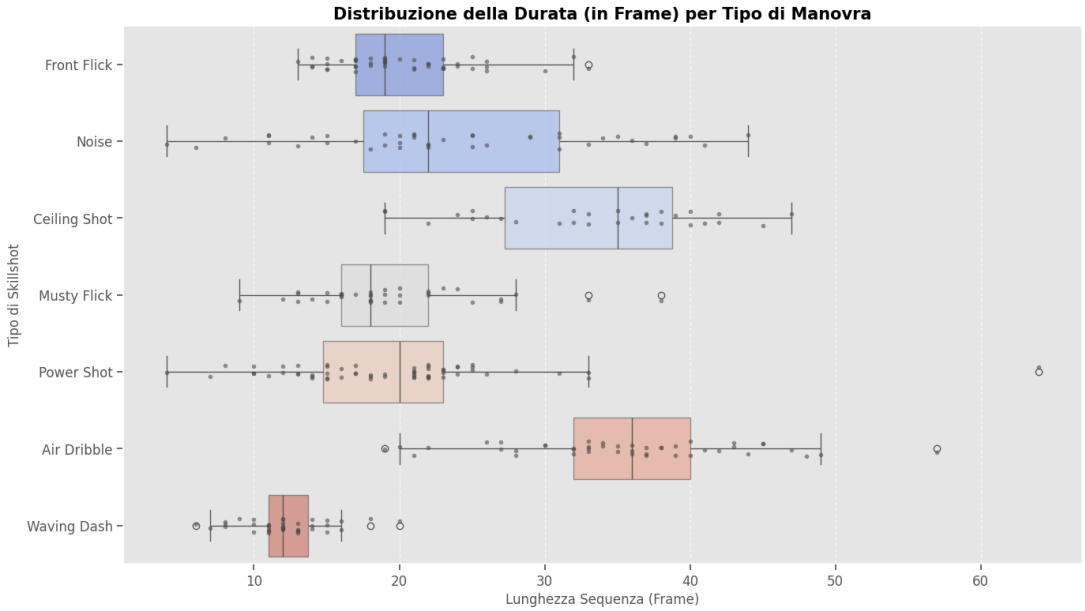

Figura 2: Distribuzione della durata (in frame) per tipo di manovra.

La durata media globale è di 23.13 frame. Come mostrato in Figura 2, manovre come *Air Dribble* tendono ad essere lunghe, mentre i *Flick* sono brevi. Questa eterogeneità rende necessario il Feature Engineering.

4 Feature Engineering: Strategia Fisica e Comportamentale

Per condensare le serie temporali in un dataset tabulare, abbiamo implementato funzioni specifiche nel nostro codice Python per generare metriche robuste.

4.1 Quartili e Toggle Count

Per le variabili fisiche, abbiamo sostituito Min/Max con i Quartili (q_{25}, q_{75}) per ignorare gli outlier. Per gli input del controller, abbiamo creato la metrica **Toggle Count**, definita nel codice sottostante, che conta quante volte un tasto cambia stato (da 0 a 1 e viceversa).

```
# Funzioni di aggregazione personalizzate
def q25(x): return x.quantile(0.25)
def q75(x): return x.quantile(0.75)

def count_toggles(x):
    """Conta quante volte il segnale cambia valore"""
    return (x.diff().abs() > 0).sum()

# Applicazione delle aggregazioni
X_physics = grouped[physics_cols].agg(['mean', q25, q75])
X_inputs = grouped[input_cols].agg(['mean', count_toggles])
```

Listing 2: Definizione delle metriche (Tratto dalla funzione `perform_advanced_feature_engineering`)

La metrica `count_toggles` è fondamentale per distinguere manovre che richiedono input frenetici ("spamming") da quelle che richiedono una pressione costante.

5 Modellazione e Ottimizzazione

5.1 Selezione dei Modelli

Abbiamo confrontato due famiglie di algoritmi:

1. **Random Forest Classifier:** Scelto per la capacità di gestire feature miste e per la robustezza all'overfitting.
2. **Support Vector Machine (SVM):** Scelta come termine di paragone, utilizzata con kernel RBF e scaling dei dati.

5.2 Protocollo di Validazione

Il dataset è stato suddiviso con una strategia **Stratified Hold-Out 80/20**. Sul training set (238 campioni), abbiamo applicato una **Stratified 5-Fold Cross-Validation**. I risultati preliminari hanno decretato la vittoria del Random Forest:

Modello	F1-Score Medio (CV)	Deviazione Std
Random Forest	**0.8764**	$\pm$ 0.0433
SVM (RBF Kernel)	0.7816	$\pm$ 0.0462

Tabella 1: Risultati della Cross-Validation.

5.3 Ottimizzazione (Grid Search)

Abbiamo ottimizzato il Random Forest tramite `GridSearchCV`. La configurazione migliore identificata dal nostro script è:

- `n_estimators`: 200
- `criterion`: 'gini'
- `max_depth`: None (espansione completa)

6 Risultati Sperimentali

6.1 Performance Quantitative

Il modello ottimizzato, valutato sul Test Set (60 campioni), ha ottenuto un'accuratezza del **86.67%**, superando la baseline casuale (13.33%).

6.2 Analisi della Matrice di Confusione

La matrice di confusione rivela che il modello distingue perfettamente manovre come *Power Shot* e *Waving Dash*. Si notano lievi confusioni tra *Musty Flick* e *Front Flick*, dovute alla similarità cinematica delle azioni.

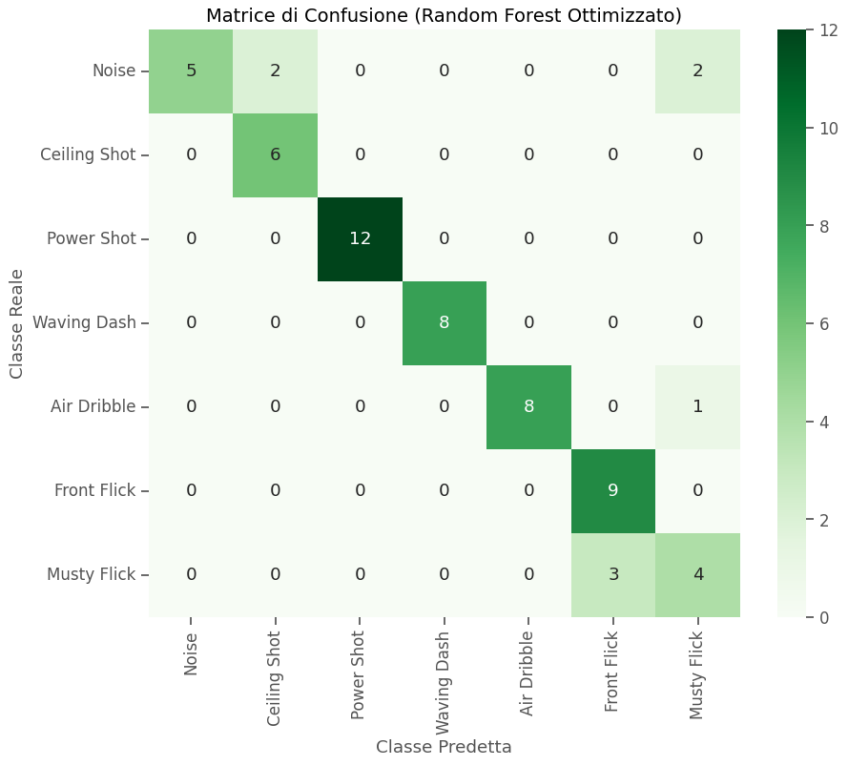

Figura 3: Matrice di Confusione sul Test Set.

6.3 Interpretazione: Feature Importance

Sfruttando l'attributo `feature_importances_` del Random Forest, abbiamo identificato i fattori decisionali chiave.

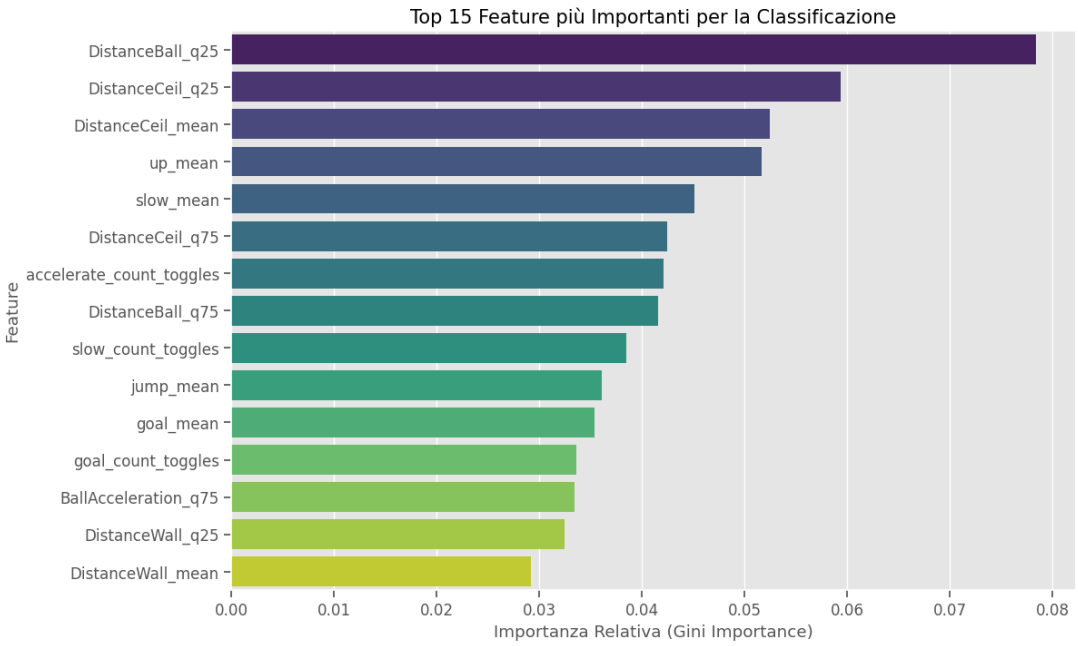

Figura 4: Top 15 Feature per Gini Importance.

L'analisi conferma che le metriche introdotte nel codice (`q25` e `count_toggles`) sono risultate le più predittive, validando la nostra strategia di Feature Engineering.

7 Conclusioni

Il progetto ha dimostrato l'efficacia dell'apprendimento supervisionato per la classificazione di skillshot in *Rocket League*. L'implementazione di un parser vettoriale personalizzato e l'introduzione di feature comportamentali hanno permesso di ottenere prestazioni elevate (F1-Weighted 0.86) utilizzando un modello Random Forest.

Riferimenti bibliografici

[1] Mathonat, R., Boulicaut, J.-F., & Kaytoue, M. (2020). *Rocket League Skillshots Data Set*. UCI Machine Learning Repository. Disponibile online: `https://archive.ics.uci.edu/dataset/858/rocket+league+skillshots`