# PROGETTO DI INTELLIGENZA ARTIFICIALE

## REINFORCEMENT LEARNING PER OPERAZIONI DI TRADING


**Docenti:**

**Prof. Vincenzo Deufemia**
**Dott. Gaetano Cimino**

**Studenti:**
**Cavaliere Mattia**
**Citro Carmine**
**Nunziata Vincenzo**

# Introduzione e Obiettivo del Progetto

- Il progetto ha come obiettivo l'applicazione di Reinforcement Learning (RL) nel trading finanziario.
- Si mira a massimizzare i profitti attraverso decisioni ottimali basate su dati storici.
- Il RL permette di apprendere strategie di trading efficienti bilanciando esplorazione e sfruttamento.

# Contesto e Dati di Mercato

- L'algoritmo è applicato al mercato finanziario e testato su dati storici di azioni i  dati utilizzati provengono da Yahoo Finance tramite la libreria Python yfinance.

- Il modello può essere esteso ad altri assets come criptovalute

# Background Teorico

## Esistono tre principali paradigmi di Machine Learning

Supervised Learning: Apprendimento con dati etichettati.
Unsupervised Learning: Trova pattern senza etichette.
Reinforcement Learning: Apprendimento basato su ricompense e penalità, il RL è il più adatto per problemi sequenziali e con incertezza temporale.

## Fondamenti del Reinforcement Learning

Elementi chiave:
Stato (s): rappresentazione dell'ambiente in un dato momento.
Azione (a): decisione presa dall'agente.
Ricompensa (r): segnale che indica il valore dell'azione.
Policy ($\pi$): strategia che guida le azioni dell'agente.
Oblettivo: massimizzare la ricompensa cumulativa.

## Q-Learning

Il Q-Learning è un algoritmo di RL che apprende il valore di ogni azione in ogni stato.
Aggiornamento basato sull'equazione di Bellman:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max'_a Q(s',a') - Q(s,a)]$$

L'agente bilancia esplorazione e sfruttamento con la strategia ε-greedy.

## Deep Q-Network (DQN)

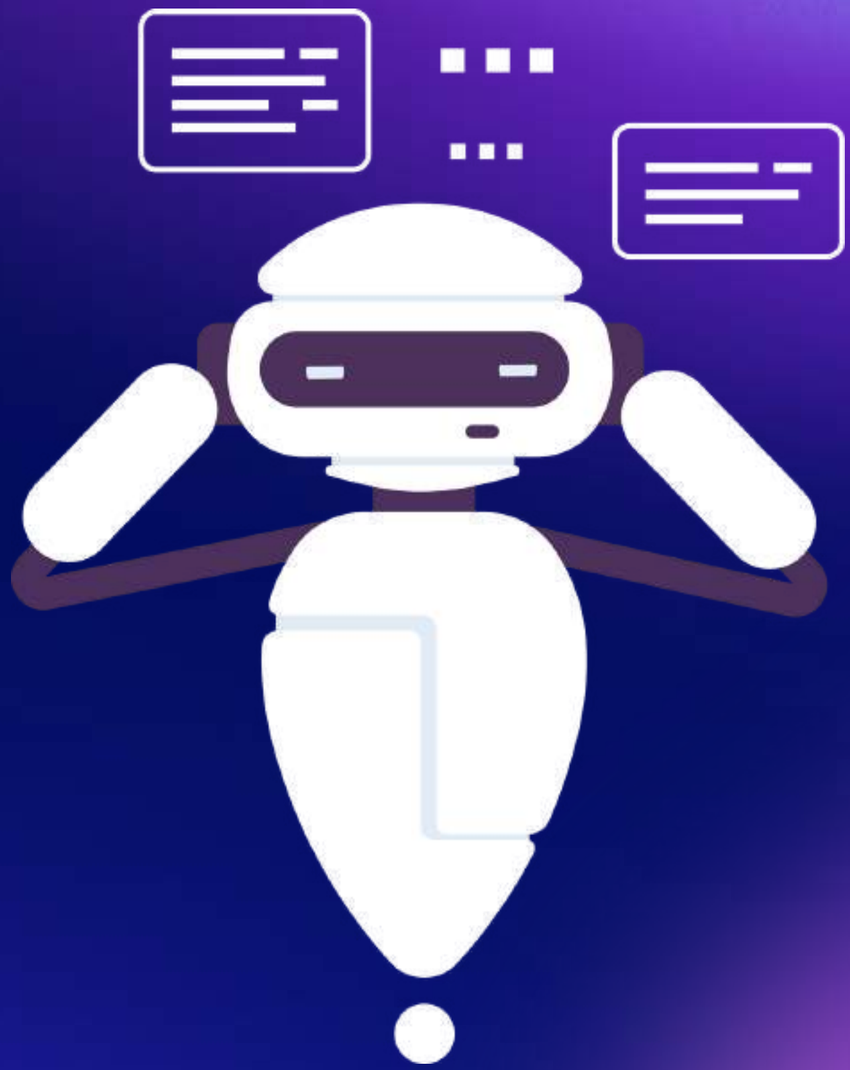
Migliora il Q-Learning utilizzando reti neurali per gestire spazi di stato molto grandi o continui
Elementi chiave:
Replay Buffer: evita la dipendenza sequenziale nei dati.
Rete Target Separata: stabilizza l'addestramento.
Strategia ε-greedy: migliora l'apprendimento bilanciando esplorazione/sfruttamento.

# Approccio - Due Strategie

- DQN: Approccio basato su reti neurali per gestire spazi di stato continui.
- Q-Learning con discretizzazione: Riduzione dello spazio degli stati con tecnica di binning.

# Formalizzazione del Problema (MDP)

- Il problema è modellato come un Markov Decision Process
- Stati (S): descrivono la situazione corrente.
- Azioni (A): decisioni possibili.
- Funzione di Transizione (P): regole di passaggio tra stati.
- Funzione di Ricompensa (R): segnala il valore di una decisione.
- Fattore di Sconto (γ): bilancia ricompense immediate e future.



# Spazio degli Stati

Lo spazio degli stati S è definito come:

$$S = \{s_1, s_2, \ldots, s_n\}$$

dove ogni stato $s \in S$ è una rappresentazione unica e completa del sistema, descritto da variabili osservabili:

$$s = (x_1, x_2, \ldots, x_k)$$

dove:

$X_i$ è una variabile rilevante per il problema.

La dimensionalità k dipende dal numero di variabili necessarie.

Nel problema del trading, lo spazio degli stati è continuo e multidimensionale:
$S \in R^k$

Ogni time step t corrisponde a un giorno e lo stato è determinato da:

- Prezzi di chiusura giornalieri
- Volume giornaliero
- Differenza di prezzo tra il time step precedente e quello corrente
- Profitto (o perdita) generato al time step precedent

Lo stato può essere formalizzato come:

$$s_t = (P_t, V_t, D_t, \Omega_t)$$

dove:

- $P_t \in R^{30}$ prezzi di chiusura della finestra temporale corrente.
- $V_t \in R^{30}$ volumi giornalieri della finestra temporale.
- $D_t \in R^{30}$ differenza di prezzo tra il time step t e t-1 con: $\Delta p_i = p_i - p_{i-1}$
- $\Omega_t \in R$ profitto o perdita accumulata al time step *t*

Quindi, ogni stato $S_t$ appartiene a:

$$s_t \in R^{91}$$

# Spazio delle Azioni

Lo spazio delle azioni è discreto e definito come:

$$A = \{a_1, a_2, \ldots, a_k\}$$

Nel nostro caso:

$$A = \{Vendi, Compra, Mantieni\}$$

quindi k=3.

L'agente seleziona un'azione $a_t \in A$ in base allo stato corrente $s_t$ e alla politica appresa $\pi(s_t)$ :

$$\pi(s_t) = \arg\max_a Q(s_t, a)$$

dove $Q(S_t, a)$ è la funzione Q appresa dal modello DQN.

# Funzione di transizione

In un processo decisionale di Markov (MDP), le transizioni sono definite dalla funzione:

$$P(s'|s,a)$$

che rappresenta la probabilità di raggiungere lo stato s′ dallo stato s eseguendo l'azione a.

Nel DQN, le transizioni possono essere deterministiche o stocastiche.

Nel nostro caso, sono trattate come deterministiche, quindi lo stato successivo è determinato dalla funzione:

$$s' = f(s,a)$$

dove f(s,a) modella:

1. Lo shift della finestra temporale.
2. L'aggiornamento del portafoglio in base all'azione scelta.
3. L'evoluzione di prezzi, volumi e differenze di prezzo.

Le transizioni aggiornano ogni componente dello stato come segue:

Prezzi di chiusura:

$$P_{t+1} = [p_{t-28}, p_{t-27}, \ldots, p_t, p_{t+1}]$$

Volumi:

$$V_{t+1} = [v_{t-28}, v_{t-27}, \ldots, v_t, v_{t+1}]$$

Differenza di prezzo:

$$D_{t+1} = [p_{t-27} - p_{t-28}, \ldots, p_{t+1} - p_t]$$

Profitto/perdita:

$$\Omega_{t+1} = \Omega_t + \Delta P_{t+1}$$

dove $\Delta P_{t+1}$ è il cambiamento del valore del portafoglio.

In generale, la transizione dallo stato $s_t$ allo stato $s_{t+1}$ data l'azione $a_t$ si scrive come:

$s_{t+1} = f(s_t . a_t)$ con:

$$f(s_t, a_t) = \begin{cases} P_{t+1}, & \text{prezzi shiftati in avanti} \\ V_{t+1}, & \text{prezzi shiftati in avanti} \\ D_{t+1}, & \text{differenze di prezzo aggiornate} \\ \Omega_{t+1}, & \text{profitto/perdita aggiornato} \end{cases}$$

# Funzione di ricompensa

La ricompensa $R(s_t, a_t)$ associata all'azione $a_t$ eseguita nello stato $s_t$ è data da:

$$R(s_t, a_t) = \Delta P_t - penalita_h - penalita_{drawdown} - penalita_{transizione}$$

dove $\Delta P_t$ è la variazione del valore del portafoglio tra il time step attuale e quello dell'ultimo trade.

## Penalità della Ricompensa

### 1.Penalità sulle azioni e inattività:

$$penalita_h = \lambda_h \left( h_{azioni} + h_{inattivo} \right)$$

### 2.Penalità di Drawdown:

Si applica se il valore del portafoglio scende oltre il 50% del massimo valore raggiunto fino al tempo t :

$$\text{penalita}_{drawdown} = \begin{cases} \lambda_d \cdot \alpha \cdot \dfrac{V_{max} - V_t}{V_{max}}, & \text{se } \dfrac{V_{max} - V_t}{V_{max}} > 0.5 \\ 0, & \text{altrimenti} \end{cases}$$

dove:

- $V_{max} = max_{t' \leq t} V_{t'}$ è il massimo valore del portafoglio fino a $t$.
- $\lambda_d$ e $\alpha$ sono coefficienti di penalizzazione

### 3.Penalità di Transazione

Penalizza ogni azione proporzionalmente al prezzo corrente:

$$\text{penalita}_{transizione} = \lambda_t \cdot 0.05 \cdot \text{prezzo}_t$$

La funzione di ricompensa finale risulta quindi:

$$R(s_t, a_t) = (V_t - V_\tau) - \lambda_h (h_{azioni} - h_{inattivo}) - \begin{cases} \lambda_d \cdot \alpha \cdot \dfrac{V_{max} - V_t}{V_{max}}, & \text{se } \dfrac{V_{max} - V_t}{V_{max}} > 0.5 \\ 0, & \text{altrimenti} \end{cases} - \lambda_t \cdot 0.05 \cdot \text{prezzo}_t$$

I coefficienti $\lambda_h, \lambda_d, \lambda_t$ possono essere adattati per bilanciare l'influenza delle penalizzazioni

# Fattore di sconto

Il fattore di sconto $\gamma$ è un valore scalere che determina l'importanza delle ricompense future rispetto a quelle immediate:

$$\gamma \in [0, 1]$$

dove:

Se $\gamma = 0$ l'agente considera solo al ricompensa immediata, ignorando quelle future.

Se $\gamma \to 1$ l'agente dà grande peso alle ricompense future, considerandole quasi equivalenti a quelle immediate .

Nel presente progetto, dopo vari test, è stato trovato che il valore ottimale del fattore di sconto è $\gamma = 0.99$ per DQN e $\gamma = 0.9$ per Q-Learning

# Implementazione del DQN

- **Obiettivo:**
  - Usare il Deep Q-Network (DQN) per gestire il trading su azioni
- **Motivazione:**
  - Necessità di un algoritmo capace di apprendere in ambienti con spazi degli stati continui
- **Architettura della rete neurale:**
  - **Input:** Stato (91 caratteristiche)
  - **Hidden layers:** 2 strati con 320 neuroni ciascuno, attivazione ReLU
  - **Output:** 3 neuroni (Compra, Vendi, Mantieni)

# Funzione di perdita del DQN

- **Obiettivo:**
  - Minimizzare la differenza tra il valore stimato e il valore atteso della funzione $Q(s, a)$
- **Formula della funzione di perdita:**
  - DQN usa una rete target fissa per stabilizzare il training:

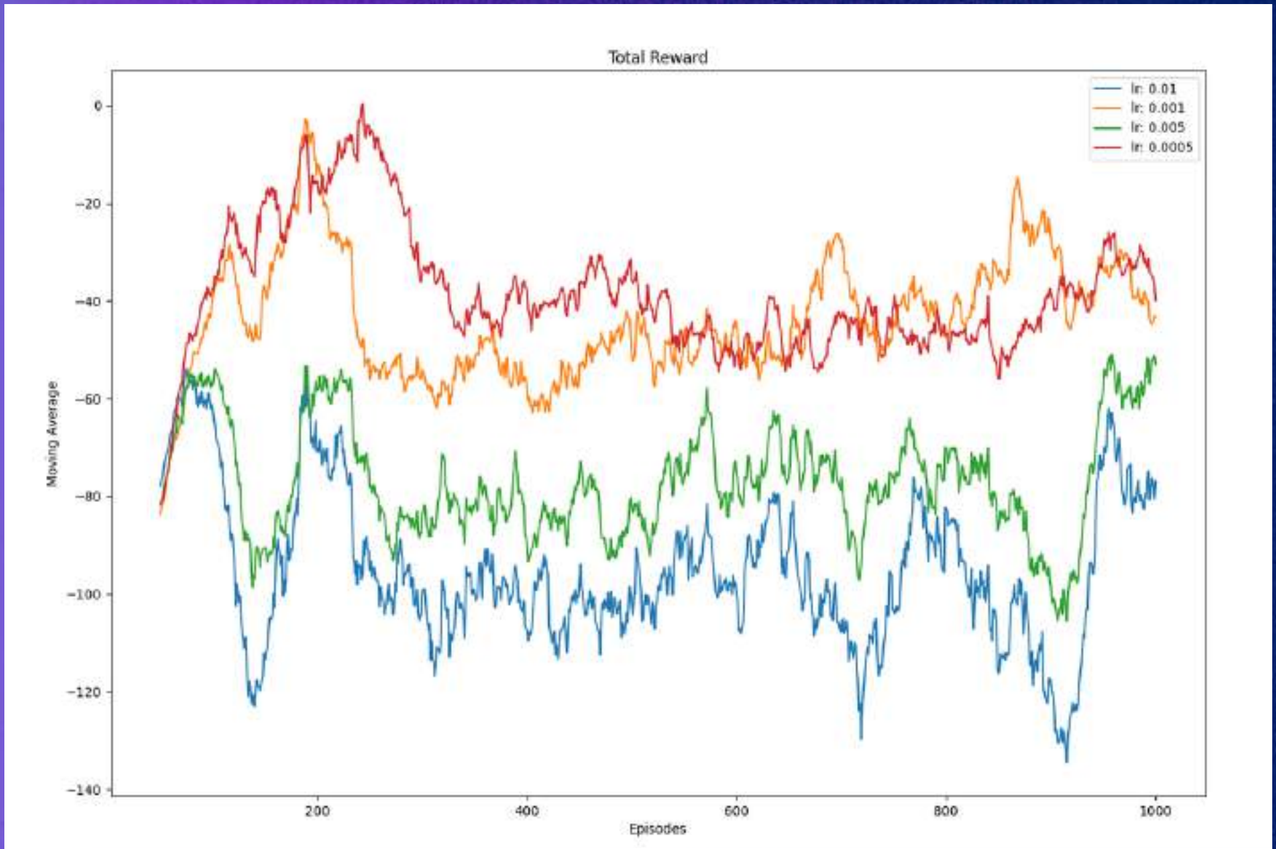
$$\text{Loss} = E[(r + \gamma \max_{a} Q(s', a'; \theta^{-}) - Q(s, a; \theta))^2]$$

- **Tecniche usate per migliorare la stabilità:**
  - **Experience Replay Memory:** Evita correlazioni nei dati di addestramento
  - **Rete target separata:** Parametri aggiornati periodicamente (ogni 5 episodi)

- **Ottimizzazione degli iperparametri**
  - **Learning Rate** $\alpha$
  - **Epsilon-decay** $\epsilon$
  - **Fattore di sconto** $\gamma$

# Parametri chiave e tuning

## 1000 episodi di training su 11 asset finanziari.



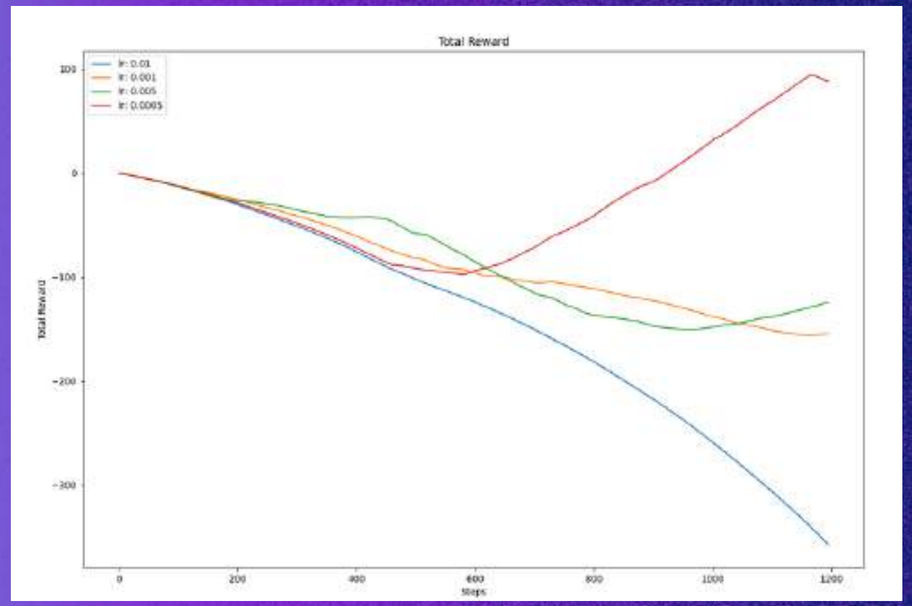

Per il tasso di apprendimento $\alpha$, sono stati presi in considerazione i valori 0.01, 0.001, 0.005 e 0.0005. La figura  mostra le curve di apprendimento per $\epsilon = 0.95$ e $\gamma = 0.9$.

- **Grafico:** curve di apprendimento per diversi valori di Learning Rate

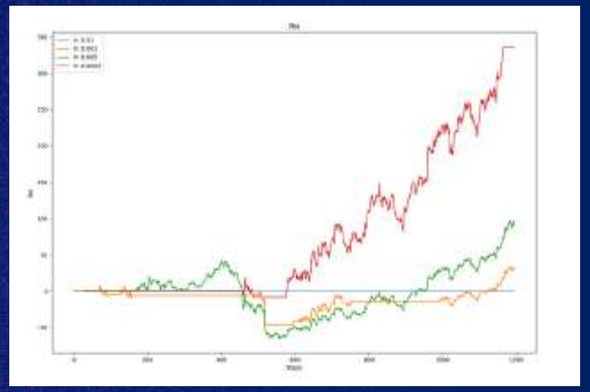
# Implementazione del DQN

## Risultati sull'ambiente di test

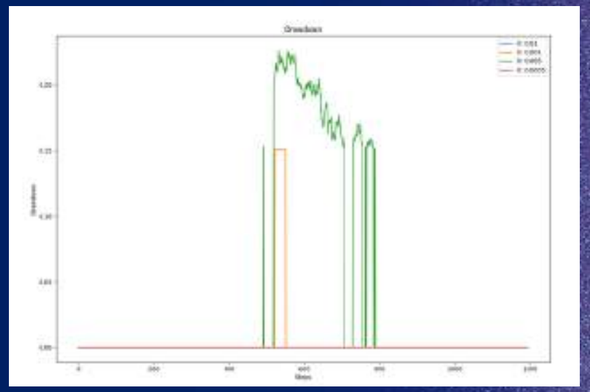
In conclusione i risultati migliori sono stati ottenuti con i valori $\alpha$ =0.001 e $\alpha$ = 0.0005.
Analizzando l'esecuzione sull'ambiente di test, si puo notare che quest'ultimo valore ha ottenuto risultati nettamente migliori in tutti gli aspetti di valutazione.



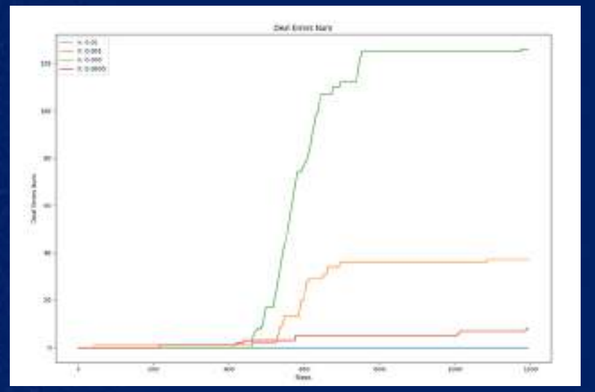

Ricompensa totale



ROI



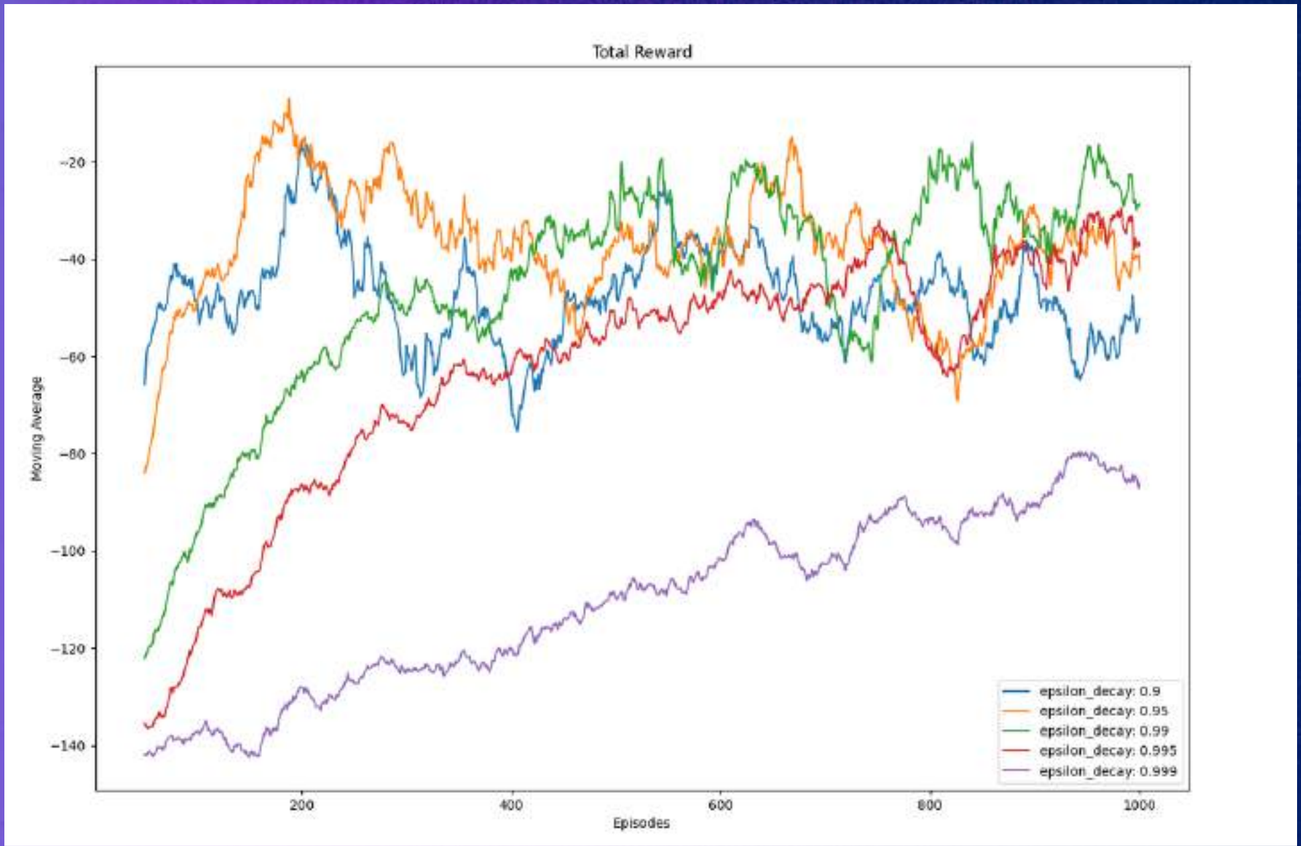
Drawdown



Errori

# Parametri chiave e tuning

**1000 episodi di training su 11 asset finanziari.**




Per il tasso di decadimento ε, sono stati presi in considerazione i valori 0.9, 0.95, 0.99, 0.995, 0.999.

- **Grafico:** curve di apprendimento per diversi valori di Learning Rate

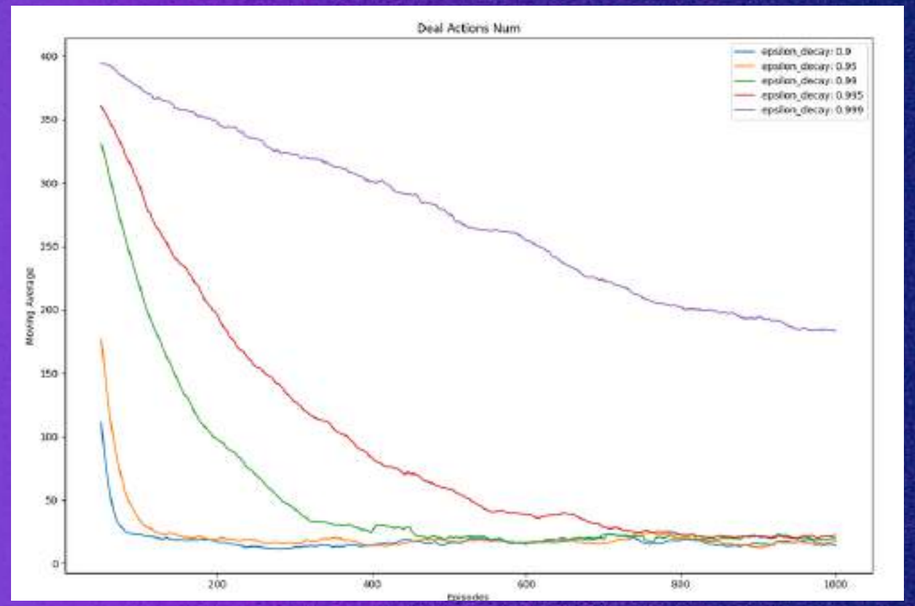
L'agente riesce ad ottenere mediamente risultati migliori con una maggiore esplorazione.

- L'unica eccezione avviene per ε=0.999

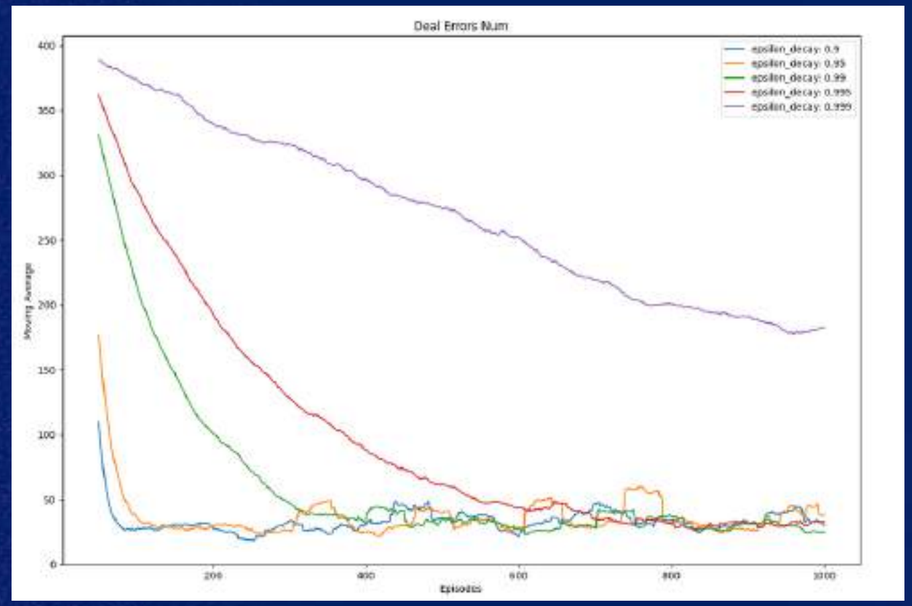
# Implementazione del DQN

## Dettagli sull’addestramento

Un agente casuale tende a eseguire molte azioni e a commettere numerosi errori, ottenendo ricompense fortemente negative.
Terminata la fase di esplorazione, l'agente apprende rapidamente a ridurre sia il numero di azioni di trading eseguite sia il numero di errori commessi.



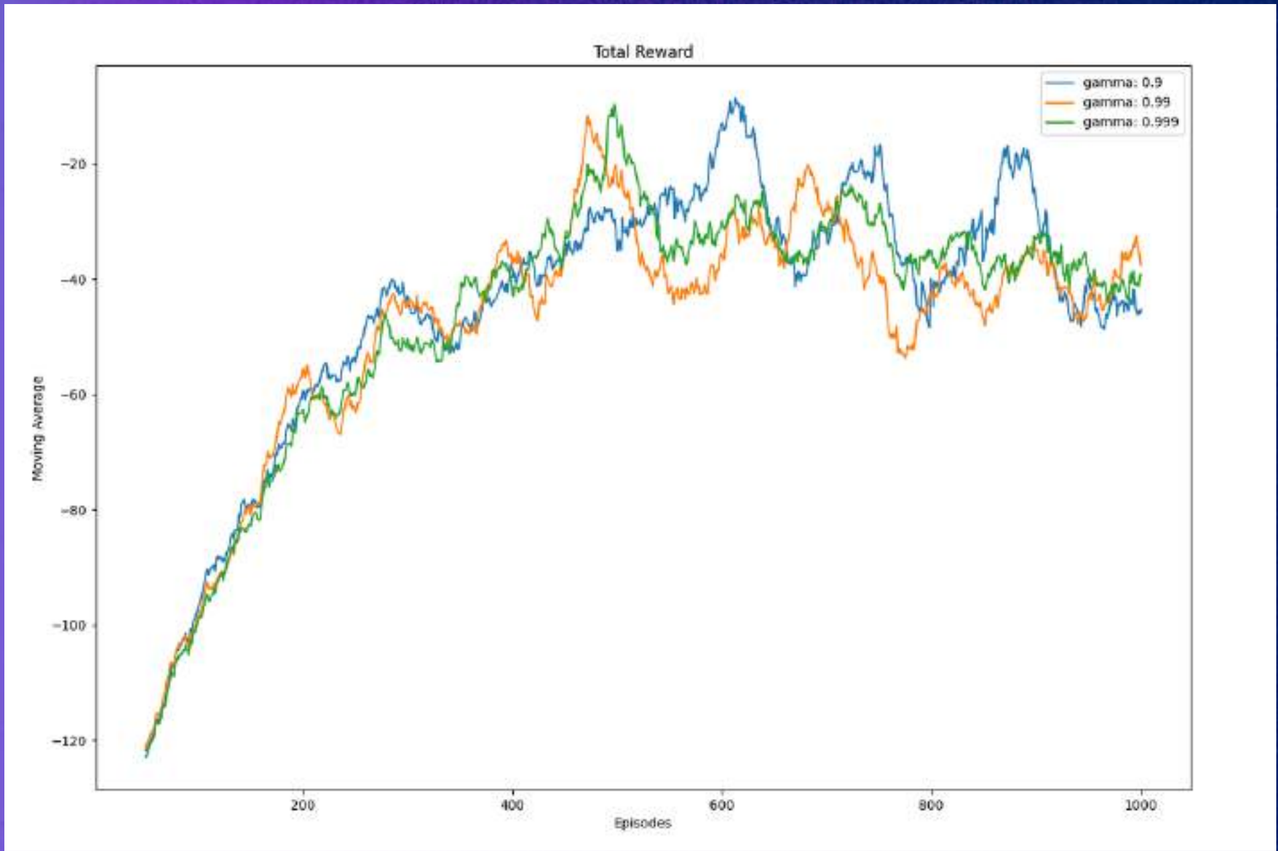

Numero di azioni di trading




Numero di errori

# Parametri chiave e tuning

**1000 episodi di training su 11 asset finanziari.**




Per il fattore di sconto γ, sono stati presi in considerazione i valori 0.9, 0.99, 0.999

- **Valore basso**: tendeze generalmente locali
  - Utile al trader in posizione “corta”
- **Valore alto**: tendenze nel lungo periodo
  - Utile al trader in posizione “lunga”

Si è scelto di affrontare il problema considerando operazioni di compravendita “classiche”.

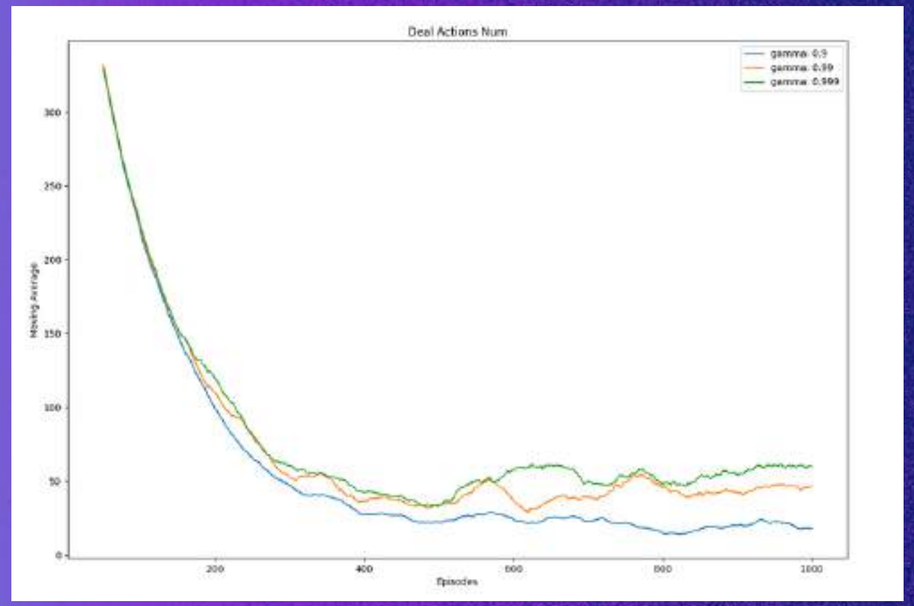
- **Grafico:** curve di apprendimento per diversi valori di Learning Rate

# Implementazione del DQN

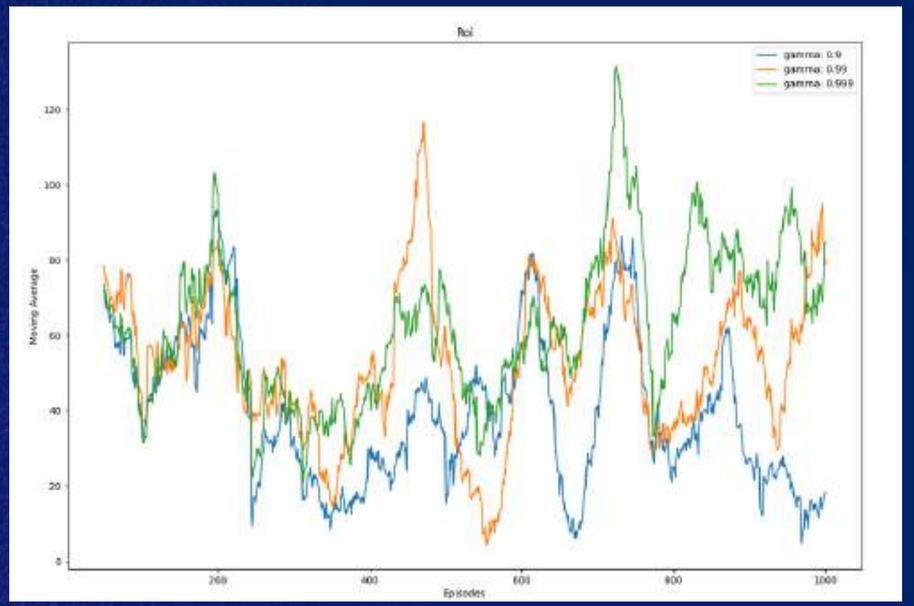
## Dettagli sull’addestramento

Poiché l'obiettivo principale dell'agente dovrebbe essere quello di massimizzare i profitti, in questo caso si è scelto di sacrificare il numero di azioni eseguite, privilegiando un approccio più redditizio.

Un agente più “miope” tende ad avere un atteggiamento più prudente




**Numero di azioni di trading**




**ROI**

Tuttavia, la strategia adottata dagli agenti più “lungimiranti” consente di ottenere profitti nettamente più elevati.

# Risultati DQN – Test su ambiente reale

**Iperparametri ottimali:** α=0.0005 ε=0.99 γ=0.999


- **Risultati ottenuti dopo 2000 episodi di addestramento:**
  - **ROI:** +274.53%
  - **Durata media di un'operazione:** 46 giorni
  - **Numero di operazioni:** 11
  - **Errori commessi:** 163 (pochi rispetto a Q-Learning)

# Implementazione del Q-Learning con Discretizzazione

- **Obiettivo:**
  - Verificare se il Q-Learning può essere applicato con successo al problema del trading
  - Problema principale: il Q-Learning non può gestire direttamente spazi degli stati continui

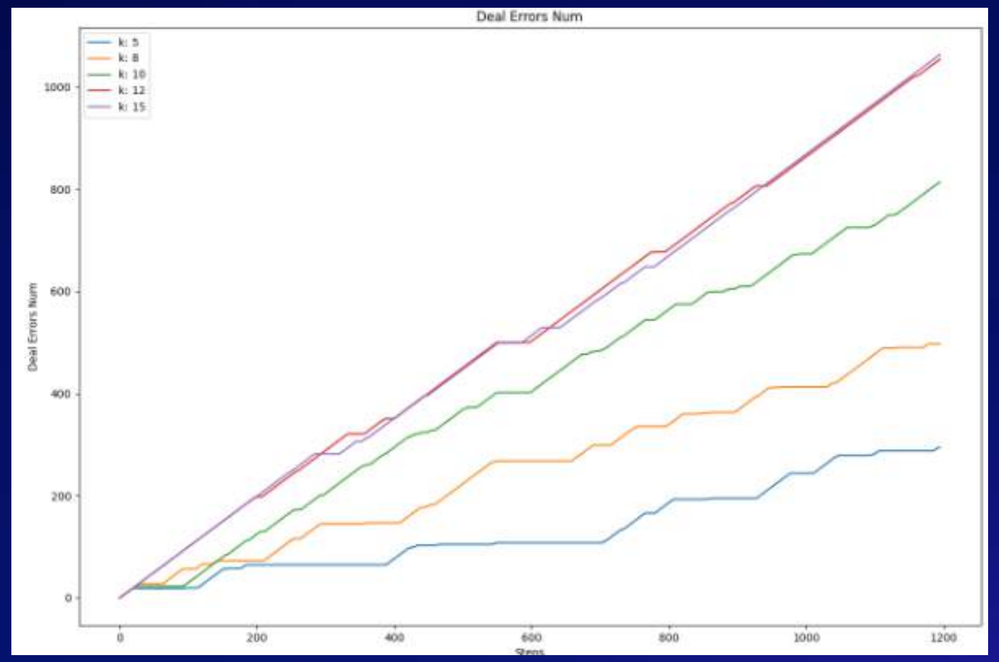
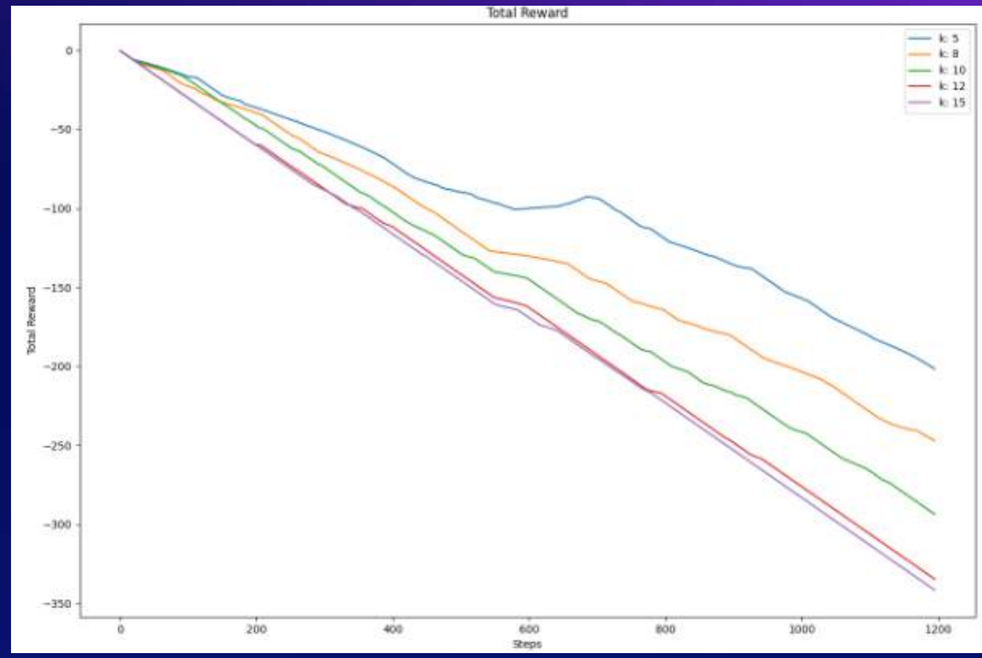



**Grafico: Numero di errori**

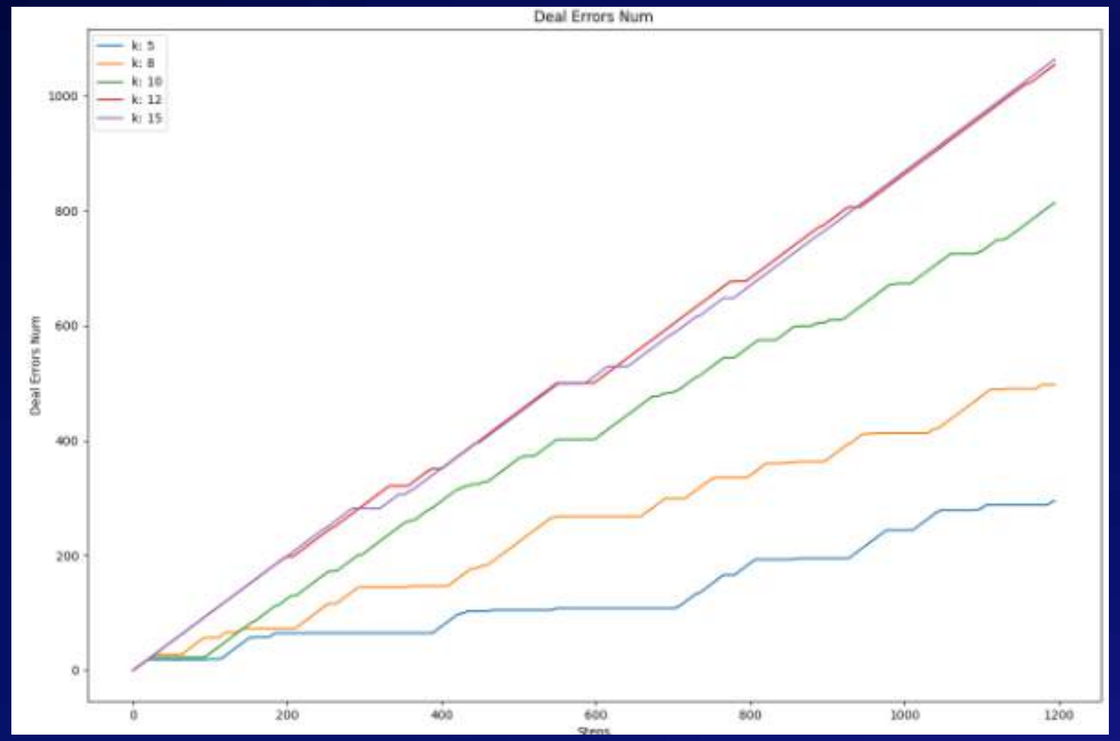
**Grafico: Ricompensa totale**

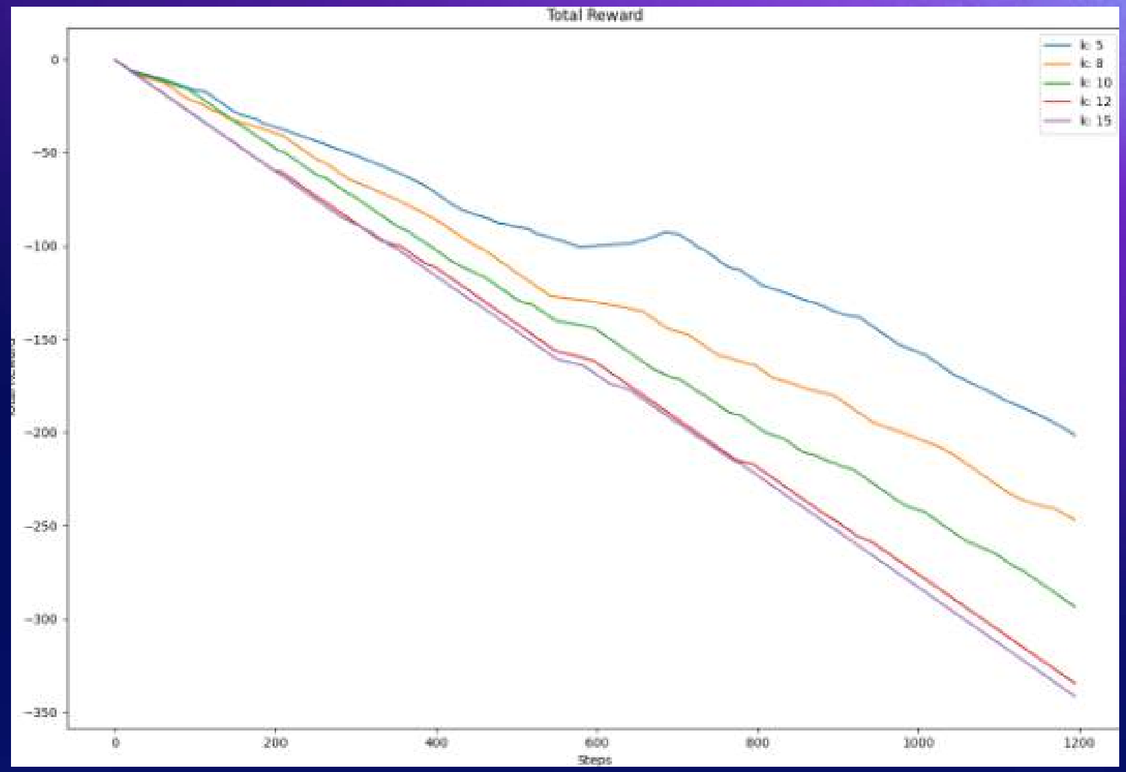



- **Soluzione: Discretizzazione dello spazio degli stati**
  - **Tecncia utilizzata: Equal Width Binning**
    - I dati vengono suddivisi in *k* bin di uguale ampiezza
    - Gli stati vengono rappresentati in base al bin in qui ricade il loro valore

# Tecnica di Discretizzazione

**Grafici: Risultati ottenuti sull'ambiente di test per diversi valori di k.**




- **Scelta del valore di k:**
  - ***k* troppo basso (es. 5):** Perdita di informazioni, trend di prezzo "appiattiti"
  - ***k* troppo alto (es. 15):** Aumento della complessità dello spazio degli stati, difficile esplorarlo completamente
  - ***k*=8:** Compromesso ottimale tra accuratezza e efficienza

# Addestramento dell’Agente Q-Learning

- **Iperparametri ottimizzati:**
  - **Learning Rate alpha:** 0.0005
  - **Taasso di esplorazione epsilon:** 0.995 (elevato per garantire sufficiente esplorazione
  - **Fattore di sconto gamma:** 0.9 (priorità alle ricompense immediate)
- **Differenza con il DQN:**
  - Il DQN utilizza una rete neurale per approssimare la funzione Q, mentre il Q-Learning memorizza direttamente i valori Q in una tabella
  - Il Q-Learning necessita di più esplorazione per imparare efficacemente

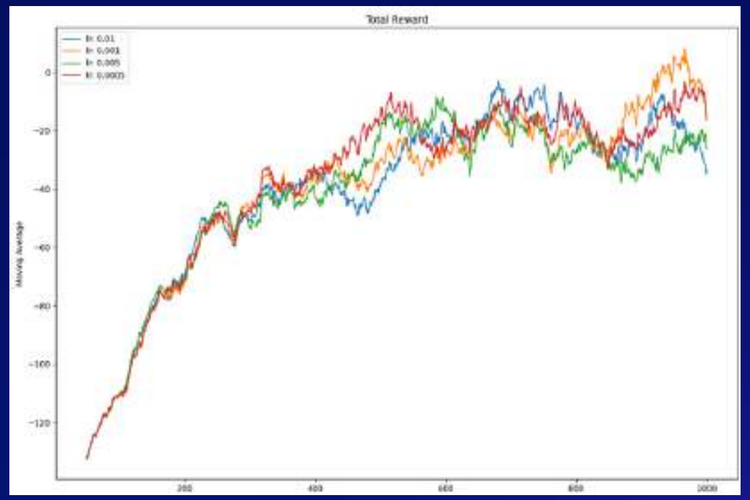



**Grafico: alpha**


**Grafico: gamma**

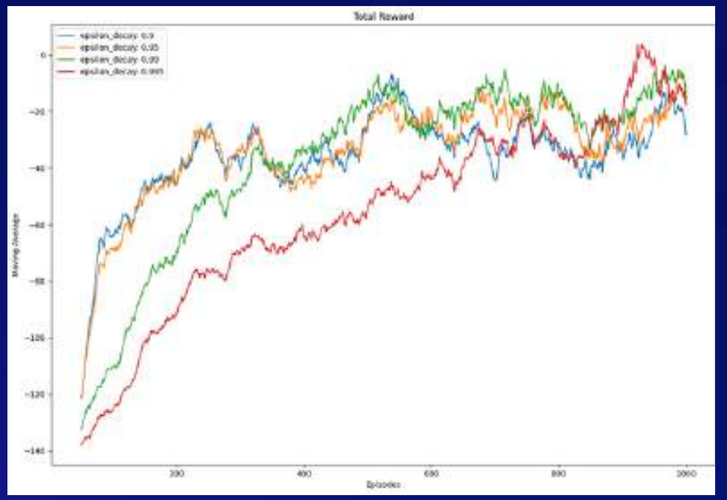



**Grafico: epsilon**

# Risultati Finali del Q-Learning

- **Analisi delle prestazioni su ambiente di test:**
  - **Numero di opreazioni eseguite:** 12
  - **Durata media di un'operazione:** 18 giorni (più breve rispetto a DQN
  - **ROI finale:** +71.40% (nettamente inferiore rispetto a DQN)
  - **Errori commessi:** 491 (quasi il triplo degli errori rispetto a DQN)

- **Problema principale:**
  - **Alta sensibilità alla discretizzazione:** Se uno stato non è stato visitato in fase di addestramento, il Q-Learning non sa come comportarsi

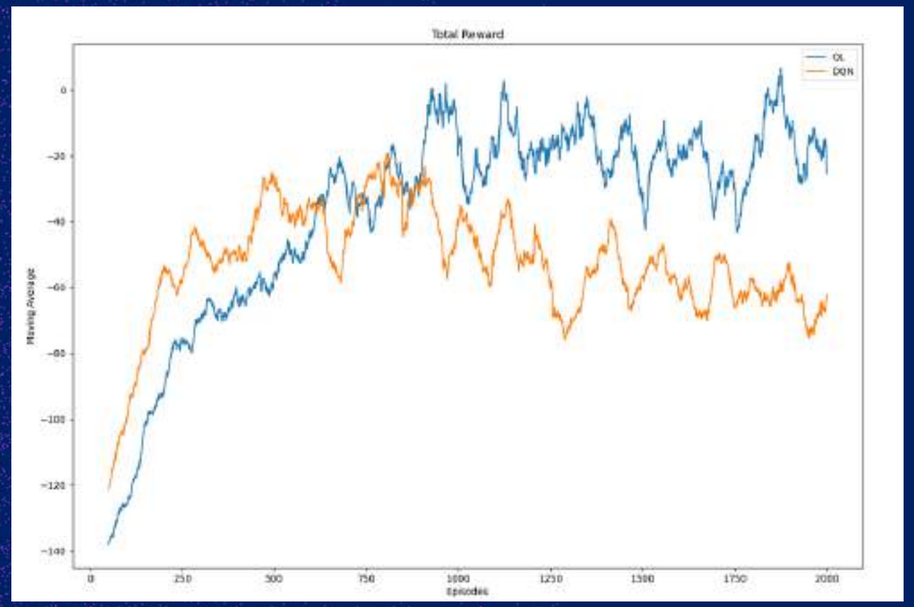
# Confronto tra DQN e Q-Learning – Analisi

| Metrica | DQN | Q-Learning |
|---|---|---|
| ROI (%) | +274.53% | +71.40% |
| Numero di operazioni | 11 | 12 |
| Durata media operazione (giorni) | 46 | 18 |
| Errori commessi | 163 | 491 |

- **Osservazioni principali:**
  - DQN ottiene un ROI più alto e commette meno errori
  - Q-Learning esegue più operazioni, ma meno efficaci
  - DQN generalizza meglio su nuovi dati

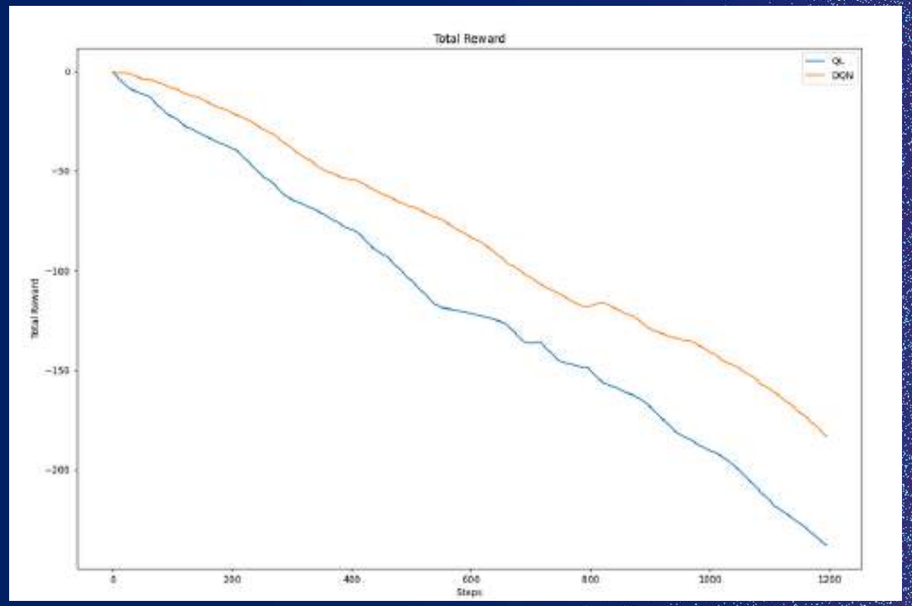
# Confronto tra DQN e Q-Learning – Analisi

L’agente Q-Learning ha prestazioni migliori durante l’addestramento




**Ambiente di addestramento**

**Ambiente di test**




L’agente DQN ha prestazioni migliori su stati mai visitati

# Impatto della Discretizzazione sulla Generalizzazione

**Perché il Q-Learning ha risultati peggiori?**

- **Ridotta capacità di generalizzazione:**
  - Il Q-Learning memorizza gli stati esplorati ma fatica con nuovi stati
  - Se uno stato non è stato visto durante l'addestramento il modello non sa cosa fare
- **Dipendenza dalla discretizzazione:**
  - **Troppi bin:** Troppi stati da esplorare, apprendimento difficile
  - **Pochi bin:** Si perdono informazioni chiave sui trend di mercato

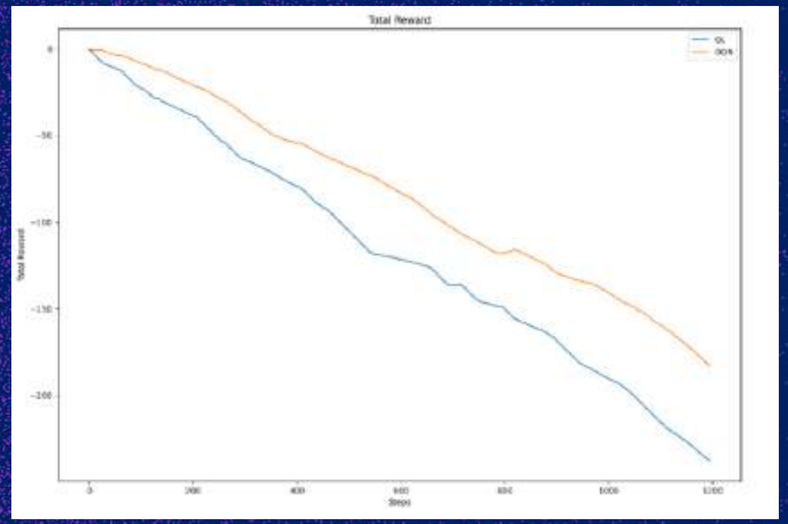



**Grafico: Ricompensa Totale**

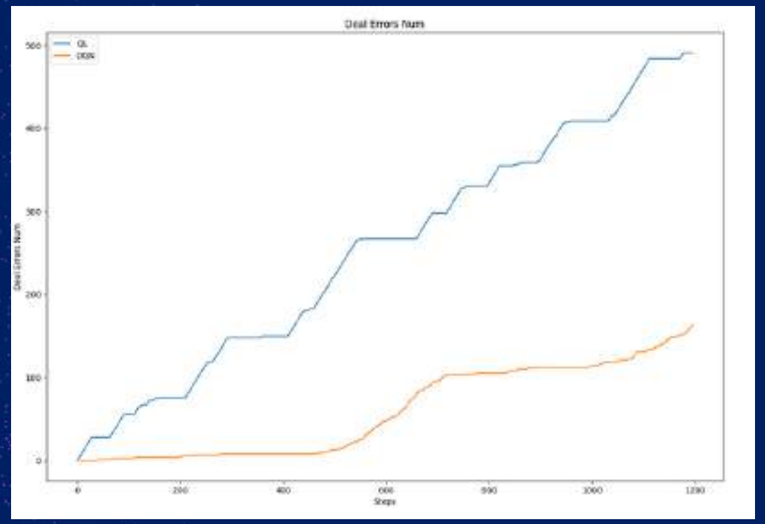



**Grafico: Numero di azioni**

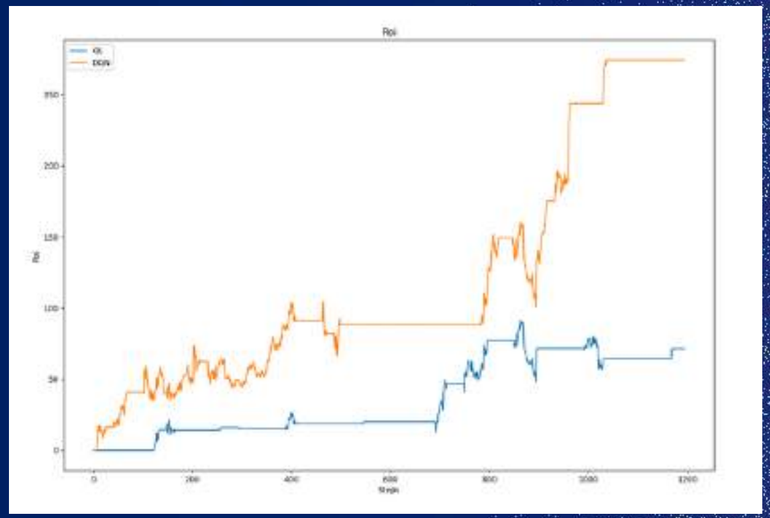



**Grafico: ROI**

# Conclusioni e Sviluppi Futuri

## 📝 Cosa abbiamo imparato?

✅ DQN è nettamente superiore al Q-Learning per problemi con spazi degli stati continui.

✅ L’uso della discretizzazione introduce troppe limitazioni nell’apprendimento.

✅ DQN ha ottenuto ROI più alto e meno errori grazie alla capacità di apprendere rappresentazioni più complesse degli stati.

## 📌 Come possiamo migliorare il modello?

- Utilizzo di Proximal Policy Optimization (PPO). Algoritmo avanzato che combina il meglio del DQN e dei metodi basati su politiche.
- Integrazione con NLP per analizzare news di mercato. Le informazioni finanziarie non sono solo nei dati storici. Integrare articoli e notizie potrebbe migliorare le decisioni dell’agente.
- Adattamento del modello a mercati reali. Test su mercati con maggiore volatilità. Implementazione su dati in tempo reale con API di trading.

# Grazie mille per l’attenzione