A.D. 1308
unipg
DIPARTIMENTO
DI MATEMATICA E INFORMATICA

Artificial Intelligent Models - Progetto

Agents & Data Mining

Mouse, Cat and Cheese

Sara Geoli
383403

Adam Amzir
376607

Dipartimento di Matematica e Informatica,
Università degli Studi di Perugia

January 26, 2025

Contents

1 Descrizione del problema

Ci sono due classi di agenti Topi e Gatti, che si muovono in un ambiente contenenti pezzi di formaggio.

L'ambiente è rappresentato da una matrice $i \times j$, dove sono presenti M Topi, H pezzi di formaggio, C Gatti in posizioni randomiche.

Gli agenti hanno un raggio di visione ρ nella matrice. Ogni Gatto si muove verso il Topo con maggiore attrazione nel suo raggio visivo. Ogni Topo si muove o verso il formaggio o lontano dal Gatto. La funzione di attrazione/repulsione è inversamente proporzionale al quadrato della distanza tra l'agente e la cella interessata nel raggio visivo. Generalmente la repulsione è equivalente all'attrazione ma amplificata di un fattore moltiplicativo β maggiore o uguale di 1.

Gli agenti ad ogni step si muovono di una cella alla volta in una delle otto direzioni tra: Nord, Sud, Est, Ovest, Nord-Est, Nord-Ovest, Sud-Est, Sud-Ovest. La direzione in cui andare è ottenuta determinando la cella con maggiore repulsione/attrazione. L'ambiente è considerato come uno spazio convesso.

I Topi:

- calcolano il pezzo di formaggio con maggiore attrazione nel raggio visivo ρ.
- Calcolano il Gatto più repulsivo nel raggio visivo ρ.
- Si muovono nella direzione del massimo valore di attrazione/repulsione, quest'ultimo in caso di pareggio, per diminuire o aumentare la distanza dalla cella con formaggio/Gatto.
- Se il Topo raggiunge il formaggio viene posizionato randomicamente in una cella libera.

I Gatti:

- calcolano il Topo più attrattivo nel raggio di visione ρ.
- Si muovono nella direzione determinata dalla cella più attrattiva per diminuire la distanza dall'obiettivo.

L'ambiente deve essere inizializzato con: dimensioni della grigia, numero di Agenti e pezzi di formaggio, raggio di visione, random seed, numero di step e numero di esecuzioni totali.

Infine, deve essere prodotto un file csv con ogni step percorso in ogni iterazione della simulazione da ogni agente, da usare per addestrare il classificatore e sui cui effettuare eventuali predizioni e valutazioni.

La parte successiva del progetto prevede di usare il dataset creato per addestrare algoritmi di clustering o classificazione per identificare quali individui sono Gatti o Topi in base al loro comportamento.

2 Simulation and Data Collection

2.1 Assunzioni

Gli agenti si muovono a turno: ad ogni step si muovono prima tutti i Topi e successivamente tutti i Gatti. Se il numero di Topi è zero, allora la simulazione termina in quanto tutti i Topi sono stati mangiati dai Gatti. Se si continuasse il gioco, i Gatti si sposterebbero random nella griglia fino al raggiungimento massimo di step e esecuzioni passato in fase di inizializzazione dell'ambiente. Pertanto, ciò non genererebbe dati utili per la seconda parte del progetto inerente la classificazione degli agenti.

Inoltre, nell'inizializzazione dell'ambiente deve essere impostata la velocità degli agenti tramite un booleano. Se uniforme (True), l'agente si muove ad ogni step di massimo una cella. Altrimenti (False) viene selezionato un valore random (max_speed) tra 1 e il raggio di visione ρ e, quando deve essere determinata la cella di destinazione, l'agente si muove di un numero tra 1 e max_speed celle in base a quella che maggiormente riduce o aumenta la distanza dalla cella interessata.

In aggiunta, per valutare lo stato finale della simulazione, si è deciso di dichiarare vincitori i Gatti se hanno catturato tutti i Topi, mentre vincono i Topi se hanno consumato tutto il formaggio. Se, invece, sono presenti sia i Topi che il formaggio, la partita termina con un pareggio.

2.2 Logica utilizzata

Sono definite due classi Python: Agent e Envirement.

Agent. La classe Agent memorizza lo stato dell'agente tramite le coordinate (x, y) nell'ambiente (matrice), il path percorso, l'ultima direzione di movimento e il numero di oggetti/agenti mangiati. Sono presenti altre variabili e strutture d'appoggio. Per ogni istanza della classe deve essere stabilito:

- la posizione iniziale;
- il tipo di agente, se Gatto o Topo;
- un ID;
- la velocità.

La classe contiene tre metodi: *move, results, relocate*. La funzione *move* permette di muovere l'agente differenziando se Gatto o Topo. Analizza l'ambiente entro il raggio di visione dell'agente e in caso di presenza di Topi, Gatti o formaggio calcola l'attrazione o repulsione, per poi determinare la migliore direzione in accordo con le specifiche del problema e quanto assunto. Stabilita la cella di destinazione è invocato il metodo *results* che effettua due azioni: controlla se l'agente ha mangiato un Topo o un pezzo di formaggio e successivamente aggiorna lo stato dell'agente in questione, eventualmente anche di quello mangiato e l'ambiente. Se un Topo ha mangiato un pezzo di formaggio, tramite *relocate* viene inizializzata la sua posizione randomicamente nelle matrice dove disponibile.

Envirement. Un'istanza della classe Envirement deve essere inizializzata con:

- i righe della matrice;
- j colonne della matrice:
- C numero di Gatti;
- H numero di pezzi di formaggio;
- M numero di Topi;
- random seed;
- raggio di visione;
- fattore beta;
- numero k di step;
- numero di esecuzioni totali;
- booleano per stabilire se velocità uniformi (*True*) o no (*False*).

L'ambiente genera una matrice $i \times j$ stanziando C oggetti Agent di tipo Gatto e M oggetti Agent di tipo Topo con velocità massima random o uno (in base all'ultimo booleano dell'istanza dell'oggetto Envirement), e posizionandoli casualmente nella griglia. Gli oggetti creati sono rispettivamente aggiunti alla lista *cats* e *mice*. La seguente classe contiene 5 metodi: *alternate_agents, kill_mouse, display_final_statistics, run_simulation, run*. Il primo permette di creare una lista di *agents* alternando gli Gatti e Topi creati. Mentre, tramite *kill_mouse*, il Topo con le coordinate passate viene tolto dall'ambiente e dichiarato morto. Inoltre, una variabile di supporto è aggiornata per interrompere l'esecuzione della simulazione appena i Topi sono finiti. L'avvio dell'esecuzione avviene con la chiamata del metodo *run*, che crea il file csv inizializzandolo con gli headers e invoca *run_simulation* per il numero di esecuzioni passato in fase di inizializzazione dell'ambiente. Infatti, quest'ultimo metodo muove gli agenti a turno per k steps mostrando a display la griglia in ogni stato. Infine, il metodo *run* salva i dati della simulazione nel file csv ritornando il nome. La funzione *display_final_statistics* stampa i vincitori della simulazione e lo stato finale di tutti gli agenti e il path percorso.

Pertanto, per avviare la simulazione è sufficiente stanziare un oggetto della classe *Envirement* con i parametri desiderati e invocare il metodo *run*, come mostrato nel Listato 1. L'unico parametro da passare a *run* è un booleano per indicare se visualizzare a display ogni step dell'esecuzione. Successivamente, può essere richiamato il metodo di visualizzazione delle statistiche specificando se stampare le informazioni su ogni agente e il path percorso da ognuno.

```
1 RANDOM_SEED = 47
2 env = Envirement(i = 8, j= 8, C = 3, H = 4, M = 3, random_seed = RANDOM_SEED,
     vision_range= 4, beta = 3.2, k = 3, total_run = 2, uniform_speed = False)
3
4 file_name = env.run(True)
5 env.display_final_statistics(verbose = True, path = True)
```

Listing 1: Esempio avvio simulazione.

Un esempio di visualizzazione di uno step di esecuzione sull'istanza di ambiente mostrata nel Listato 1 può essere visionata nella Figura 1. In alto è mostrato l'ID dell'agente e lo step attuale e run in cui si trova. La griglia visualizza la posizione di

Figure 1: Esempio visualizzazione step esecuzione.

tutti gli agenti e dei pezzi di formaggio. Nella parte inferiore sono riportate le informazioni sull'agente: stato (vivo o morto), posizione attuale, velocità massima consentita, numero di agenti o pezzi di formaggio consumati, e il conteggio degli step completati fino a quel momento.

Per la parte di machine learning è stato generato un dataset di circa 50 mila riga eseguendo la simulazione sull'istanza di Envirement riportata nel Listato 2.

```
RANDOM_SEED = 47
env = Envirement(i = 1000, j= 1500, C = 800, H = 600 , M = 1000, random_seed =
     RANDOM_SEED, vision_range= 100, beta = 3.2, k = 7, total_run = 3,
    uniform_speed = False)

file_name = env.run(False)
env.display_final_statistics(verbose = True, path = True)
```

Listing 2: Esempio avvio simulazione.

2.3 Analisi complessità

Per analizzare la complessità dell'algoritmo, si noti che un agente calcola la migliore direzione in base al raggio di visione passato nell'istanza dell'Envirement. Infatti, per determinare la migliore posizione in cui andare, deve essere calcolata la repulsione e/o attrazione per tutte le celle nel raggio di visione contenenti un Gatto, Topo o formaggio.

Pertanto, la complessità approssimata dell'algoritmo in tempo è riportata nella Formula 1, dove T indica il numero di esecuzioni, K il numero di step per esecuzione, M il numero di Topi, C il numero di gatti, ρ il raggio di visione e c il costo delle varie sotto-procedure. Per di più, analizzando il parametro ρ, esso varia tra 1 e $min(i, j)$, con i, j valore delle righe e colonne della matrice, che per semplicità a fini di calcolo consideriamo entrambi uguali a n (approssimazione a matrice quadrata). Pertanto, si possono definire Lower Bound e Upper Bound dell'algoritmo rispetto a ρ, visibili rispettivamente nella Formula 2 e 3.

$$T(mice\_cats\_cheese) = T \times K \times (M + C) \times 4\rho^2 + c \tag{1}$$

$$\Omega(mice\_cats\_cheese) = T \times K \times (M + C) + c \quad \text{per } \rho = 1 \tag{2}$$

$$O(mice\_cats\_cheese) = T \times K \times (M + C) \times 4n^2 + c \quad \text{per } \rho = n \tag{3}$$

In conclusione, se il numero di esecuzioni, il numero di step per esecuzione e il numero di agenti è mantenuto costante, variando il raggio di visione ρ e il numero di righe e colonne n, la complessità aumenta in modo quadratico rispetto al raggio di visione $O(\rho^2)$. L'upper bound complessivo dipende inoltre dalle dimensioni della matrice, $n \times n$, che influisce sul numero totale di celle da analizzare. Nel caso peggiore la complessità è $O(n^2)$.

3 Data Mining and Machine Learning

3.1 Introduzione

Questa parte descrive l'applicazione di algoritmi di Machine Learning al dataset ottenuto nella prima parte, con l'obiettivo di analizzare il comportamento degli agenti (Topi e Gatti) in un ambiente simulato. Sono stati utilizzati diversi algoritmi, tra cui K-Means, KNN, Random Forest e Decision Tree, per classificare e clusterizzare i dati. La valutazione dei modelli è stata effettuata utilizzando metriche come accuratezza, precision, recall e F1 score.

3.2 Algoritmi utilizzati

K-Means Clustering. Il K-Means è stato utilizzato per suddividere i dati in cluster basati sulle feature disponibili. L'obiettivo era identificare gruppi naturali di agenti con comportamenti simili attraverso il machine learning unsupervised. Pertanto, sono state utilizzate tutte le colonne del dataset ad eccetto di quella contenente il tipo di agente. Un esempio di porzione di dataset è riportato nel Listato 3.

K-Nearest Neighbors (KNN). Il KNN è stato utilizzato per la classificazione delle istanze in base alle etichette di cluster generate dal K-Means. L'algoritmo ha raggiunto un'accuratezza del 0.98 durante la cross-validation.

Random Forest. Il Random Forest è stato utilizzato per la classificazione, sfruttando un ensemble di alberi decisionali. Le feature più importanti sono state identificate come le direzioni di movimento degli agenti.

Decision Tree. L'albero decisionale è stato utilizzato per classificare i dati in base alle etichette di cluster. L'albero ha mostrato una perfetta separazione delle classi utilizzando la feature last-movement-direction-NE.

```
Run,Agent_id,Agent_kind,Path_length,x_position,y_position,
    last_movement_direction,n_eaten

0,0,mouse,0,584,1214,0,0
0,0,mouse,1,520,1150,NW,0
0,0,mouse,2,456,1086,NW,0
0,0,mouse,3,455,1087,NE,0
0,0,mouse,4,391,1023,NW,0
0,901,cat,0,159,718,0,0
0,901,cat,1,189,748,NW,1
0,901,cat,2,217,776,NW,1
0,901,cat,3,235,758,NE,1
```

Listing 3: Esempio dataset.

3.3 Valutazione modelli

K-Means Clustering Metrics:

- Inertia: 516669.6197323083
- Silhouette Score: 0.1362238558839633
- Accuracy: 0.6048225755501737;
- Precision: 0.7737108655616943;
- Recall: 0.5481976839014843;
- F1 Score: 0.6417183770883055;

- Confusion Matrix:

```
 [[2383  983]
[2770 3361]];
```

KNN Metrics:

- Accuracy: 0.9909445087922502
- Precision: 0.9912321181356715
- Recall: 0.988950276243094
- F1 Score: 0.9900898824613966
- Confusion Matrix:

```
[[5115   38]
 [  48 4296]]
```

Random Forest Metrics:

- Accuracy: 0.9978940718121512
- Precision: 0.9967830882352942
- Recall: 0.9986187845303868
- F1 Score: 0.9977000919963201
- Confusion Matrix:

```
[[5139   14]
 [  6 4338]]
```

Decision Tree Metrics:

- Accuracy: 0.9975781825839739
- Precision: 0.9979257893523853
- Recall: 0.9967771639042358
- F1 Score: 0.9973511459173097
- Confusion Matrix:

```
[[5144    9]
[  14 4330]]
```

3.4 Visualizzazioni

Visualizzazione Cluster Nella Figura 2 è riportata la visualizzazione in un piano bi-dimensionale dei cluster delineati tramite k-means e relativi centroidi. Si noti che i cluster non sono ben separati e isolati, indicando che potrebbero esserci sovrapposizioni tra i gruppi o che le caratteristiche scelte per il clustering non sono sufficientemente discriminanti.

Visualizzazione dell'Albero Decisionale. La visualizzazione dell'albero decisionale in Figura 3 mostra una perfetta separazione delle classi utilizzando la feature last-movement-direction-NE.

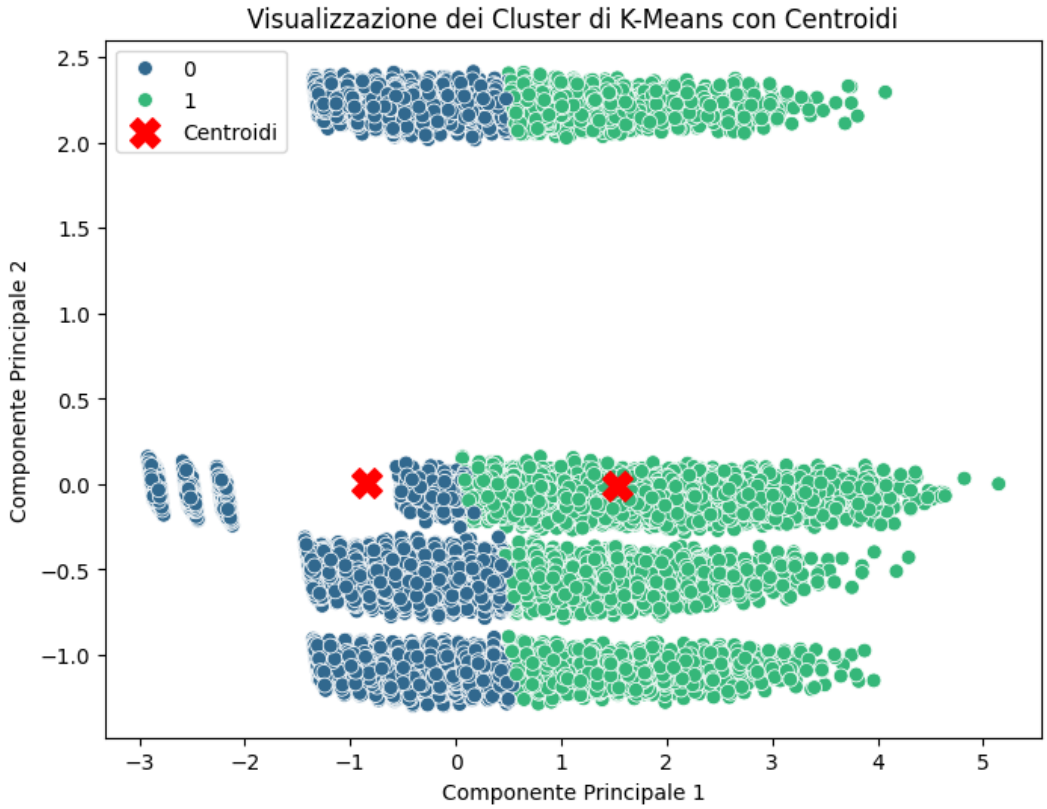

Figure 2: Visualizzazione clustering con centroidi.

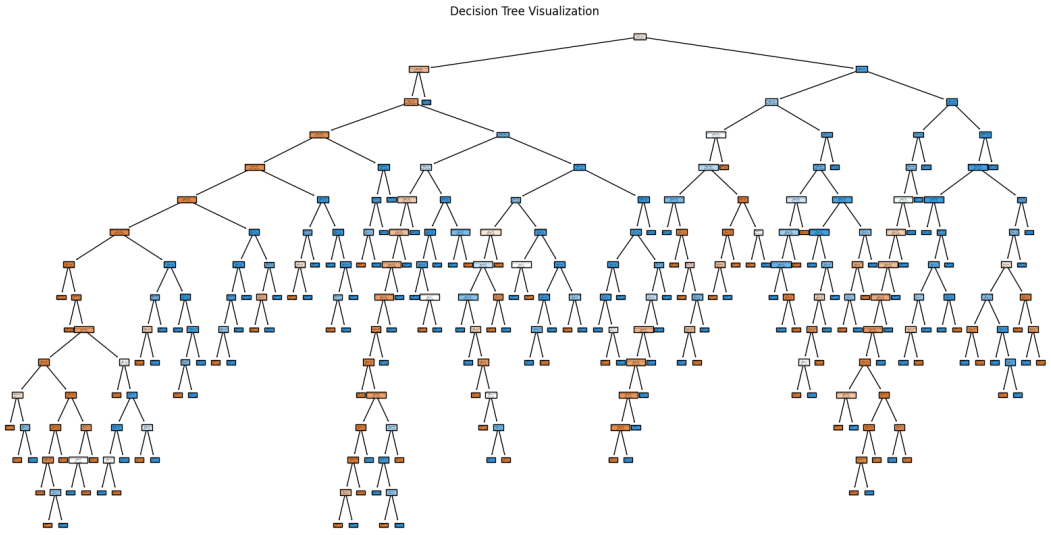

Figure 3: Visualizzazione del Decision Tree.

Feature Importances nel Random Forest. La visualizzazione delle feature importances in Figura 4 nel Random Forest evidenzia che le direzioni di movimento sono le feature più informative.

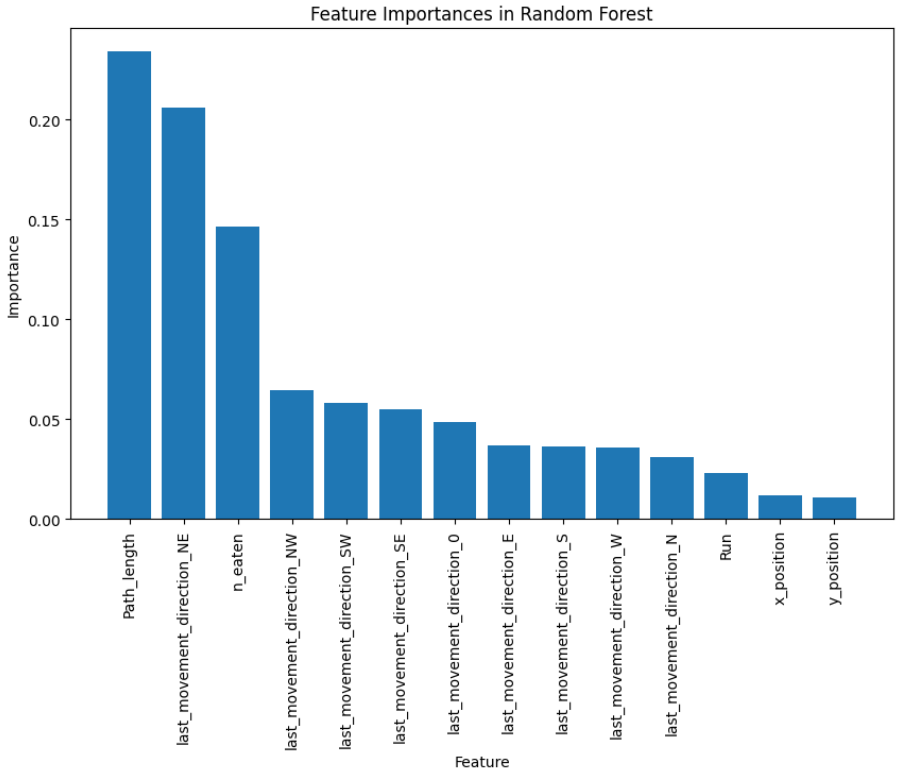

Figure 4: Visualizzazione del Random Forest.

Visualizzazione del KNN. La Figura 5 rappresenta i confini decisionali generati dall'algoritmo K-Nearest Neighbors (KNN) in uno spazio bidimensionale ottenuto tramite l'analisi delle componenti principali (PCA). La PCA è stata utilizzata per ridurre la dimensionalità dei dati, proiettandoli sulle prime due componenti principali, che sono rappresentate sugli assi X e Y.

3.5 Conclusioni

Il modello basato sull'algoritmo di clustering K-means presenta un'accuratezza del 60.48%, indicando che poco più della **metà delle previsioni** totali (sia positive che negative) sono **corrette**. La precision, pari al 77.37%, evidenzia che i cluster identificati come positivi sono per lo più accurati. Tuttavia, il **recall relativamente basso (54.82%)** suggerisce che il modello fatica a individuare tutti i campioni positivi. L'F1 score moderato (64.17%) riflette un compromesso tra precisione e recall. Inoltre, la matrice di confusione mostra un numero significativamente maggiore di falsi negativi (2770) rispetto ai falsi positivi (983), evidenziando una tendenza del modello a trascurare molti campioni positivi. La non eccellente performance dell'algoritmo K-means nel nostro dataset è evidenziata anche

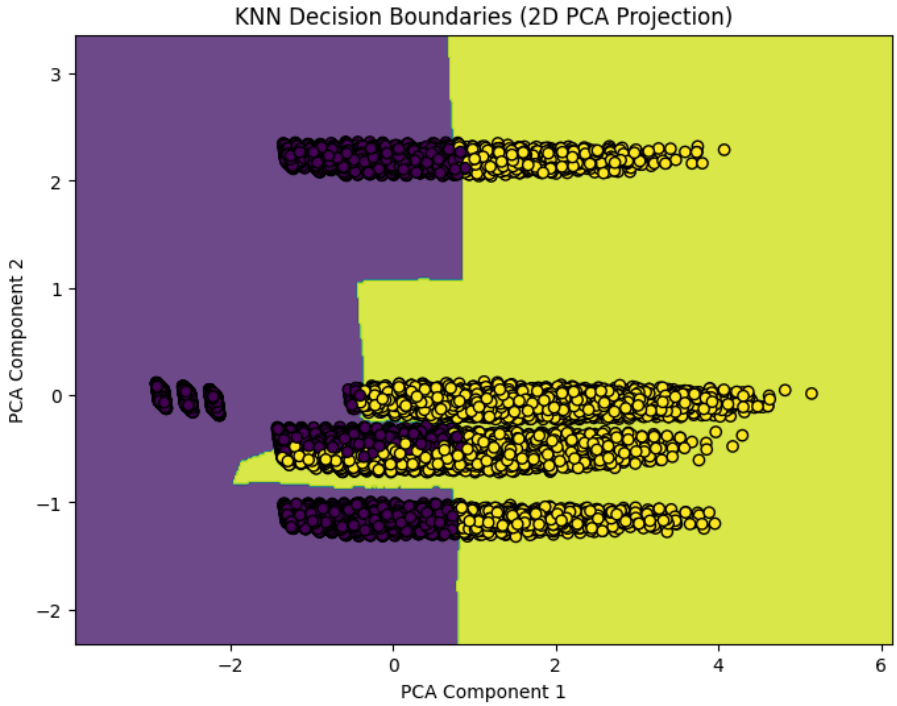

Figure 5: Visualizzazione del KNN.

dalle metriche di **inertia e silhouette**: il valore elevato dell'inertia indica che i cluster non sono compatti e ben definiti, mentre il basso valore della silhouette, appena sopra lo 0, suggerisce che i punti potrebbero essere stati assegnati ai cluster sbagliati o che la separazione tra i cluster è insufficiente. Per migliorare il valore di inerzia si sarebbe potuto utilizzare il metodo del gomito per determinare il migliore numero k di cluster, ma ciò sarebbe andato contro il nostro obiettivo di classificazione tra i due soli agenti Gatto e Topo.

L'applicazione degli algoritmi di **Machine Learning** al dataset generato ha dimostrato l'efficacia di tecniche come il **K-Nearest Neighbors (KNN)**, il **Random Forest** e gli **Alberi Decisionali** nell'analisi e nella classificazione dei dati. I risultati ottenuti, in particolare l'accuratezza del 0.98 per KNN, Random Forest e Decision Tree, evidenziano la capacità di questi modelli di apprendere pattern complessi e di separare perfettamente le classi nel contesto specifico del dataset.

Le feature relative alle direzioni di movimento (last-movement-direction-NE, last-movement-direction-NW, ecc.) si sono rivelate le più importanti per il modello Random Forest, suggerendo che il comportamento dinamico degli agenti è un fattore chiave per la classificazione. Al contrario, le coordinate di posizione (X-position e Y-position) hanno avuto un impatto minimo, indicando che la posizione statica degli agenti è meno rilevante rispetto al loro movimento.

In conclusione, il Machine Learning si è dimostrato uno strumento sufficientemente potente per l'analisi di dati complessi e dinamici, come quelli relativi a simulazioni di agenti. Tuttavia, si potrebbe esplorare l'uso di una CNN (Convolutional Neural Network) per ottenere migliori prestazioni. Addestrando il modello su un dataset costituito da

immagini o mappe di colore, si avrebbe una maggiore comprensione dell'ambiente globale e dei movimenti degli agenti.

4 Esempio di esecuzione

Durante le esecuzioni si è notato che se il numero di Topi è poco maggiore o uguale a quello dei Gatti, quest'ultimi vincono. Ciò è dovuto all'immunità dei Gatti, il cui unico scopo è mangiare Topi, mentre quest'ultimi devono mangiare formaggio e scappare: inevitabilmente, se sono in numero simile, i Gatti sterminano tutti i Topi.

Nel Listato 4 è riportato l'output di una breve simulazione, inizializzata come descritto nel Listato 1. Lo stato iniziale e lo stato finale dell'esecuzione sono rappresentati rispettivamente nelle Figure 6 e 7. La simulazione si è conclusa con la vittoria dei Gatti al termine dello step 2 di 3 nell'ultimo run (Figura 7).

Sul dataset *simulation_data_2025-01-15_19-11.csv* generato dalla simulazione si è applicato il modello addestrato con l'algoritmo K-means sul dataset da 50 mila righe per classificare gli agenti. I risultati mostrano che l'algoritmo ha una accuracy del 50%, indicando che solo la metà delle assegnazioni dei cluster corrisponde alle etichette reali. La precisione del 55% suggerisce che, tra i dati assegnati a un cluster, poco più della metà è correttamente classificata, mentre il recall del 37.9% indica che molte istanze della classe reale non vengono identificate. L'F1 score del 44.9% evidenzia una performance complessiva piuttosto bassa, suggerendo problemi nel bilanciare precisione e recall.

```
Execution terminated.
Data have been successfully collected in simulation_data_2025-01-15_19-11.csv.


------------FINAL STATISTICS------------
All Mice are DEAD!
CATS HAVE WON!

                /\_/\
                ( o.o )
                > ^ <



~~~~~~~~~~~More details~~~~~~~~~~~


----------MICE---------
Mouse with ID 0 death status is: True and has eaten 0 pieces of cheese.
 It traveled for 6 cells.
Mouse speed: 2
Mouse Path (path length, x, y, last movement direction, eaten):
[(0, 3, 0, 'O', 0), (1, 5, 6, 'SW', 0), (2, 4, 7, 'NE', 0), (3, 2, 5, 'NW', 0)
    , (4, 0, 7, 'NE', 0), (5, 1, 7, 'S', 0)]
```

```
Mouse with ID 1 death status is: True and has eaten 1 pieces of cheese.
 It traveled for 5 cells.
Mouse speed: 3
Mouse Path (path length, x, y, last movement direction, eaten):
[(0, 1, 7, 'O', 0), (1, 4, 2, 'S', 1), (2, 7, 7, 'SW', 1), (3, 2, 2, 'SE', 1),
    (4, 7, 7, 'NW', 1)]


Mouse with ID 2 death status is: True and has eaten 0 pieces of cheese.
 It traveled for 3 cells.
Mouse speed: 2
Mouse Path (path length, x, y, last movement direction, eaten):
[(0, 2, 2, 'O', 0), (1, 4, 4, 'SE', 0), (2, 6, 6, 'SE', 0)]




----------CATS---------
Cat with ID 3 has eaten 0 mice.
 It traveled for 6 cells.
Cat speed: 1
Cat Path (path length, x, y, last movement direction, eaten):
[(0, 7, 3, 'O', 0), (1, 6, 2, 'SE', 0), (2, 7, 1, 'NE', 0), (3, 0, 2, 'NW', 0)
   , (4, 0, 1, 'E', 0), (5, 1, 0, 'NE', 0)]


Cat with ID 4 has eaten 3 mice.
 It traveled for 6 cells.
Cat speed: 3
Cat Path (path length, x, y, last movement direction, eaten):
[(0, 0, 1, 'O', 0), (1, 6, 3, 'SW', 0), (2, 6, 6, 'W', 1), (3, 4, 4, 'SE', 1),
    (4, 7, 7, 'NW', 2), (5, 1, 7, 'N', 3)]


Cat with ID 5 has eaten 0 mice.
 It traveled for 5 cells.
Cat speed: 2
Cat Path (path length, x, y, last movement direction, eaten):
[(0, 1, 1, 'O', 0), (1, 3, 3, 'NW', 0), (2, 5, 1, 'NE', 0), (3, 3, 3, 'SW', 0)
   , (4, 1, 1, 'SE', 0)]

---- Machine Learning Results ----
K-means Metrics on input dataset:
Accuracy: 0.5
Precision: 0.55
Recall: 0.3793103448275862
```

```
F1 Score: 0.4489795918367347
Confusion Matrix:
    [[16   9
      18   11]]
```

Listing 4: Output simulazione di esempio

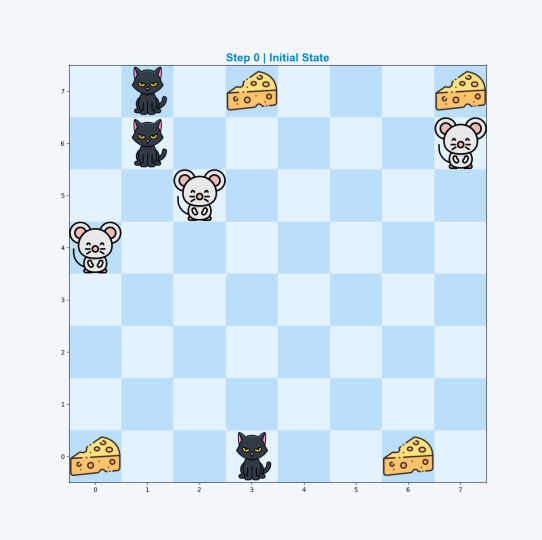

Figure 6: Stato iniziale esecuzione di esempio.

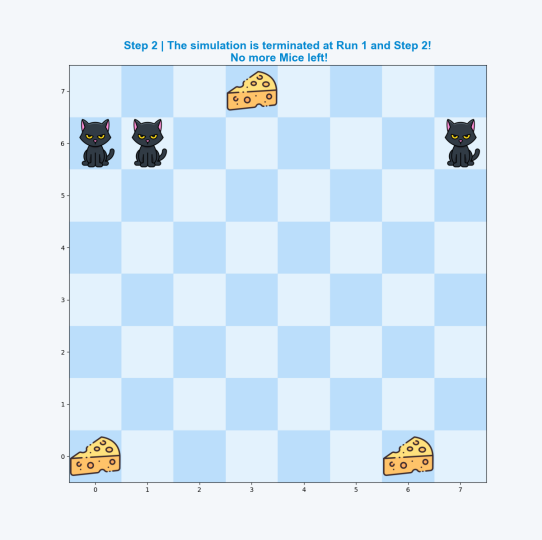

Figure 7: Stato finale esecuzione di esempio.