Creación de modelos de ML supervisado para Behavioral Cloning en el entorno Pacman UCB 3.6

Inteligencia Artificial para juegos

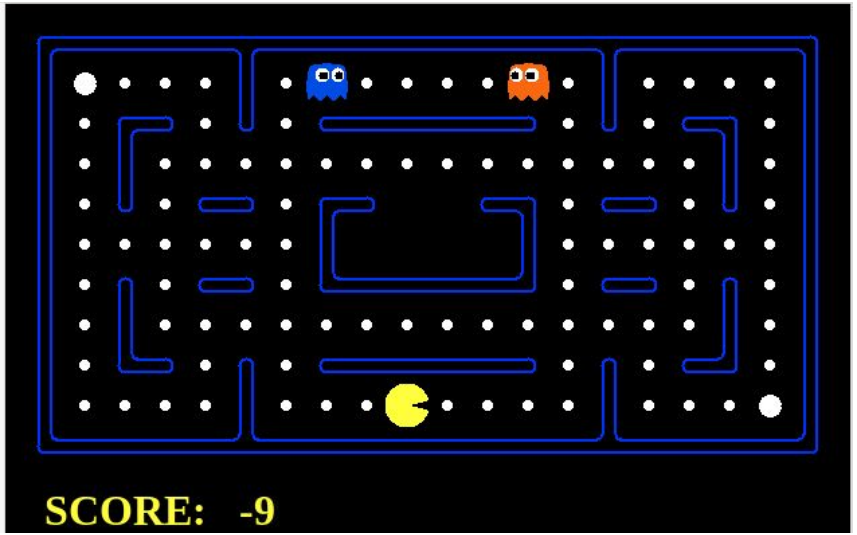
VALVERDE SORIANO, WALTER ANDRES
RAYMUNDO LUYO, CARLOS MIGUEL
VEGA GASTAÑAGA, MIGUEL ANGEL

Grupo 3

Introducción

El desafío comprende los siguientes pasos:

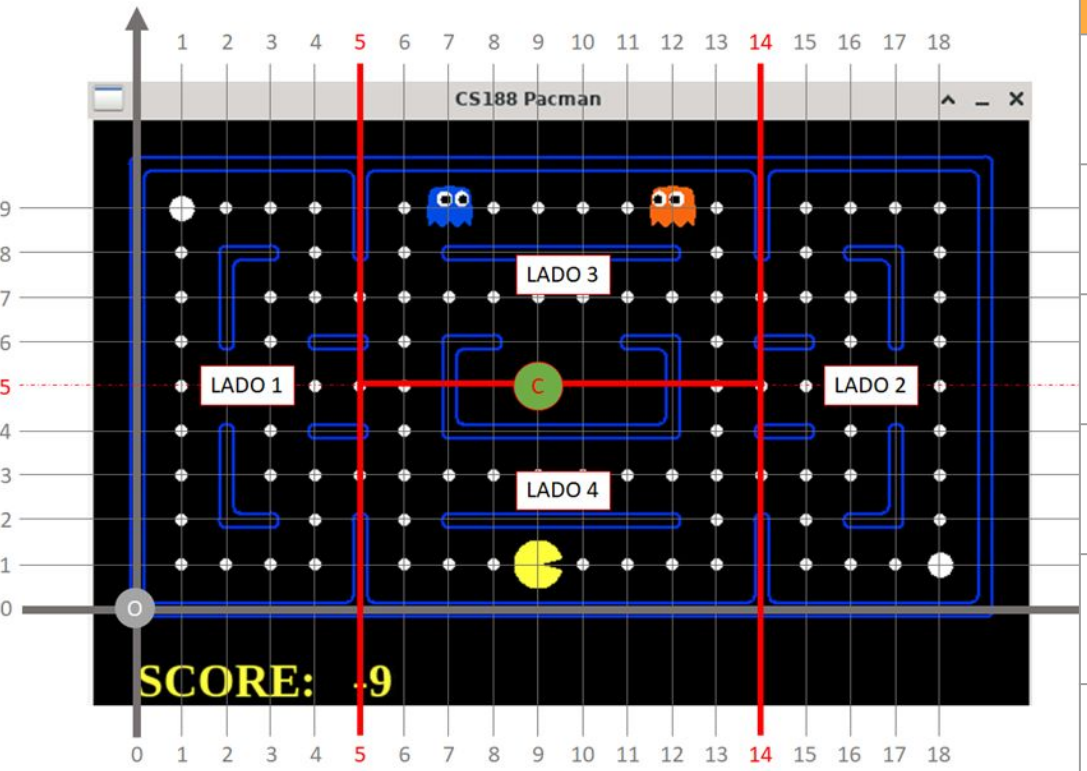
- Grabar juegos del KeyboardAgent a imitar
- Extraer features de jugadas guardadas (en replayFolder/)
- Entrenar y optimizar modelo
- Utilizar el agente clonado



Características originales

	Identificador
Posición X del fantasma 1 respecto a Pacman	g1x
Posición Y del fantasma 1 respecto a Pacman	g1y
Posición X del fantasma 2 respecto a Pacman	g2x
Posición Y del fantasma 2 respecto a Pacman	g2y
Cápsulas restantes	caps
Distancia Manhattan hasta el fantasma más cercano	dist_ghost
Distancia Manhattan hasta la cápsula más cercana	dist_caps
Promedio de distancias Manhattan hasta los 5 puntos más cercanos	dist_5dots
Puntaje	score
Cantidad de fantasmas asustados	scared
Acción (Target / Etiqueta)	action

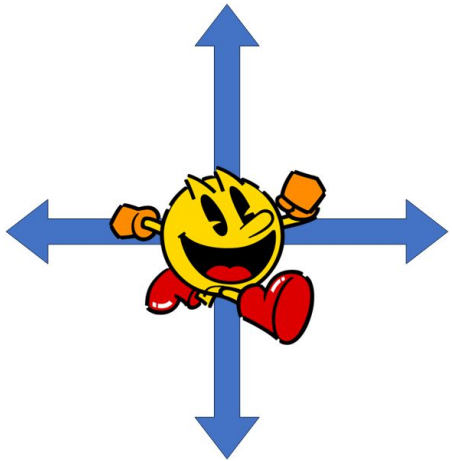
Características propuestas (1)



Característica	Identificador
Posición X de Pacman respecto al centro ©	pacx
Posición Y de Pacman respecto al centro ©	pacy
Cantidad de puntos en el Lado 1 (Ver mapa)	l1_dots
Cantidad de puntos en el Lado 2 (Ver mapa)	l2_dots
Cantidad de puntos en el Lado 3 (Ver mapa)	l3_dots
Cantidad de puntos en el Lado 4 (Ver mapa)	l4_dots

Características propuestas (2)

	Identificador
Elemento de la casilla izquierda (Oeste)	ady_w
Elemento de la casilla derecha (Este)	ady_e
Elemento de la casilla superior (Norte)	ady_n
Elemento de la casilla inferior (Sur)	ady_s

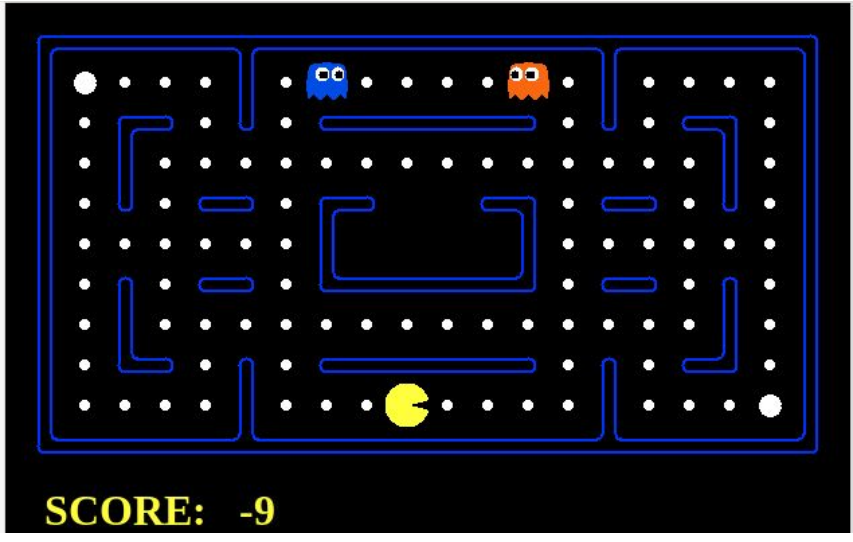


Elemento	Identificador
Fantasma	1
Muro	2
Otro	3
Comestible	4

* cumple principio de jerarquía, evita dicotonomía y elemento absorbente

Entrenamiento del modelo

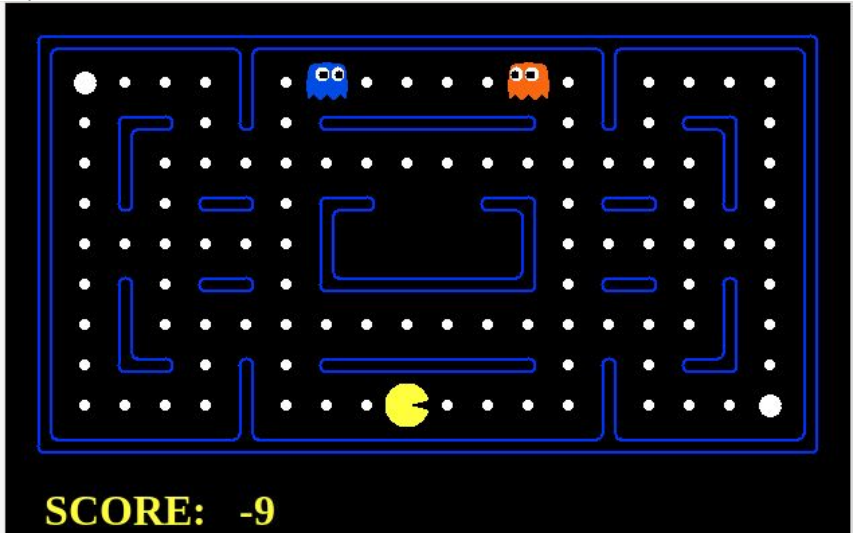
- Se eligieron 250 jugadas al azar entre los integrantes del grupo para restringir el tamaño del modelo exportado (< 100MB)
- Se realizó el siguiente flujo para la generación del modelo:
  - Se escalaron los datos usando el MinMaxScaler
  - Se entrenó una lista de modelos
  - Se eligió el modelo con mejor *accurracy*
  - Se eligieron los hiperparámetros para encontrar el mejor balance entre el tamaño del modelo (< 100MB) y el *accuracy*



Evaluación de modelos

- Se evaluaron los siguientes modelos utilizando la estrategia de K-Fold Cross Validation (10 folds)

Modelo	Accuracy
Regresión logística (LR)	59.73%
Support Vector Machine (SVC)	77.32%
Naives Bayes Gaussiano (GaussianNB)	55.55%
K-Neighbors (KNeighborsClassifier)	78.57%
Random Forest	97.18%
XG Boost (XGBClassifier)	86.68%



Modelo elegido

Random Forest Classifier:

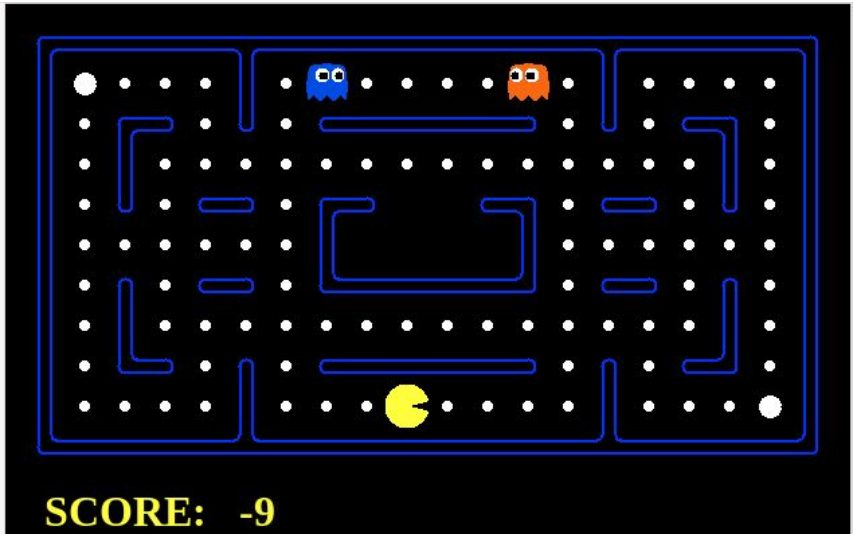
- criterion = entropy
- max_depth = 25
- n_estimators = 70

Tamaño del modelo: 96MB

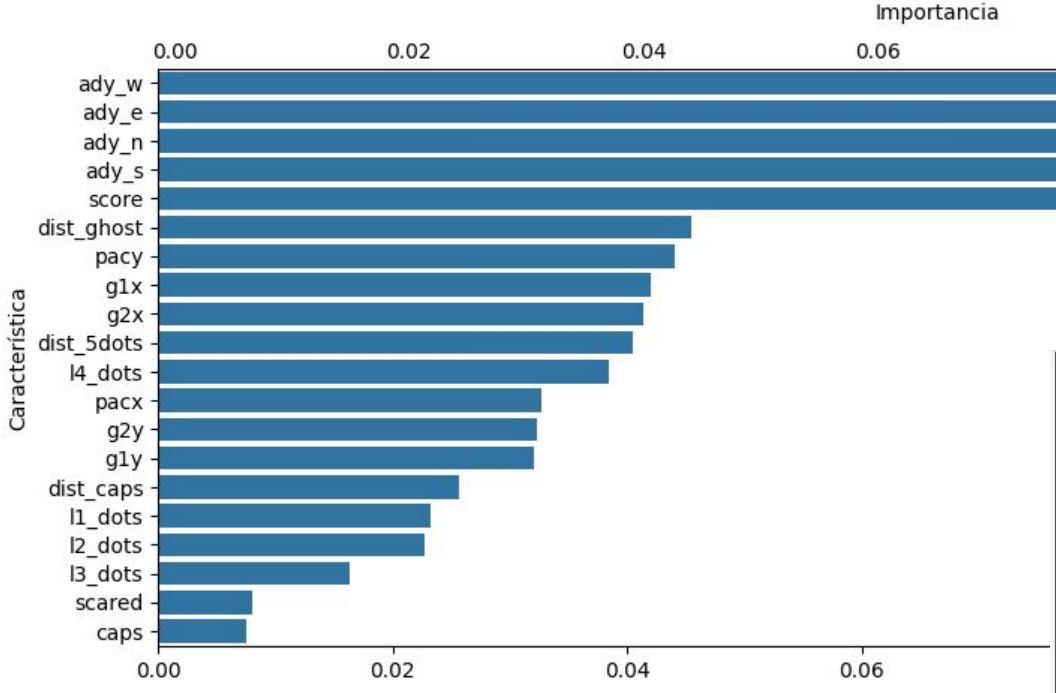
* También se exportó el MinMaxScaler

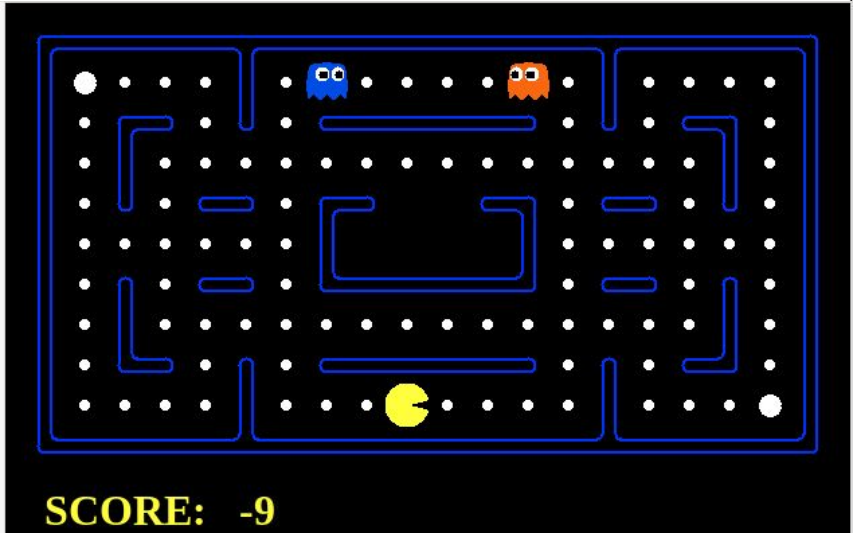
```
Accuracy score: 0.9799960388195682
[[4462   16   37   28    6]
 [  42 2866    1   13    0]
 [  35    3 2628    1    7]
 [  64   14    6 2900    6]
 [  17    0    4    3 1988]]
```

	precision	recall	f1-score	support
0.0	0.97	0.98	0.97	4549
1.0	0.99	0.98	0.98	2922
2.0	0.98	0.98	0.98	2674
3.0	0.98	0.97	0.98	2990
4.0	0.99	0.99	0.99	2012
accuracy			0.98	15147
macro avg	0.98	0.98	0.98	15147
weighted avg	0.98	0.98	0.98	15147



Importancia de las características

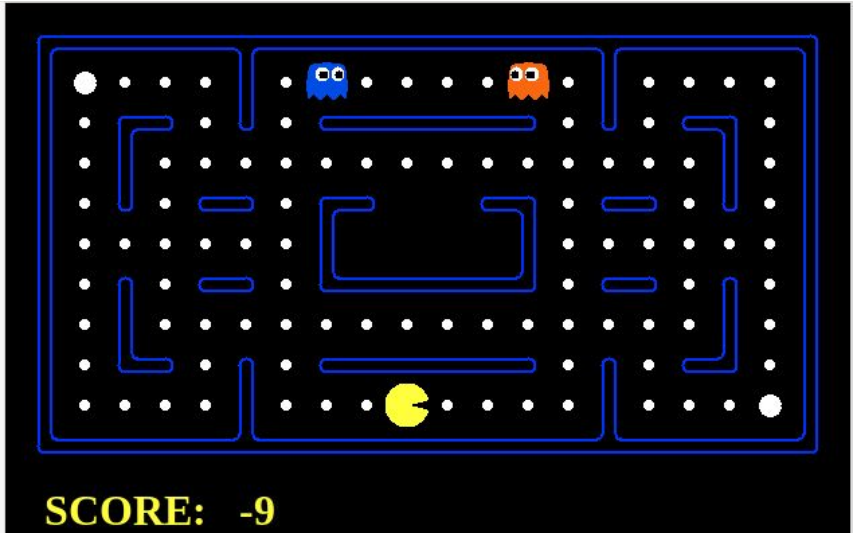




Agente implementado

Se cargó el modelo y el scaler. Luego se realizan las siguientes acciones por cada frame del juego:

- Se extraen y escalan las características
- Se obtiene el arreglo ordenado (de mayor a menor) de las probabilidades de cada movimiento
- Recorremos dicho arreglo desde el inicio:
  - Si la acción no es legal, se pasa al elemento siguiente
  - Si es el 5to stop seguido, se pasa al elemento siguiente
  - Caso contrario, se ejecuta dicha acción



Agente implementado (código)

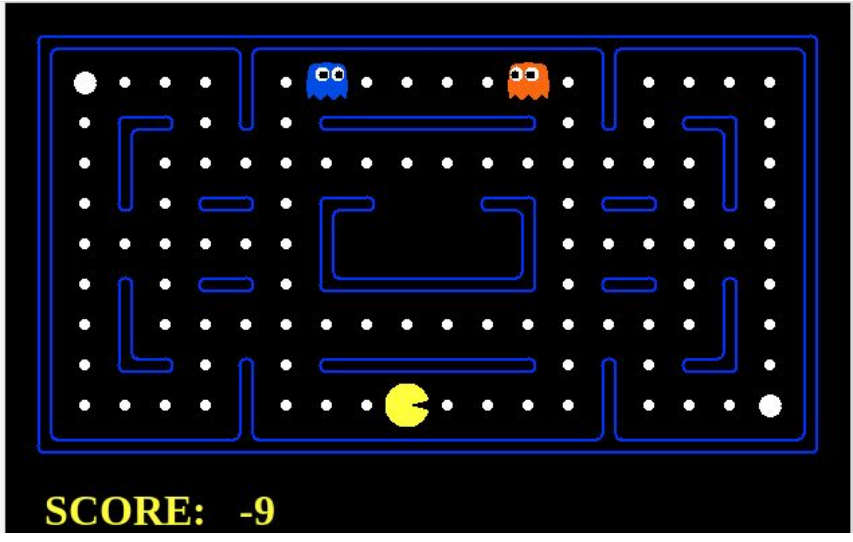
```
features = obtenerFeatures(state).reshape(1,-1)
X = self.scaler.transform(features)
```

```
probas = self.modelo.predict_proba(X)[0]
accionesProbables = np.flip(np.argsort(probas))
maxStops = 5 # colocamos un tope a los stops seguidos
```

```
movelist = ['Stop','East','North','West','South']
accionRetorno = 0 # acción por defecto
for accion in accionesProbables:
    if movelist[accion] in legalActions:
        accionRetorno = accion
        # validar que no haga más de 5 stops
        seguidos
        if accionRetorno == 0:
            self.stops_seguidos = self.
            stops_seguidos + 1
        else:
            self.stops_seguidos = 0
        if(self.stops_seguidos > maxStops):
            continue
        break
return movelist[accionRetorno]
```

Win Rate:

- Data Set: 12/250 (4.8%)
- Agente : 6/100 (6.0%)



¡Gracias por su atención!