

APRENTATGE AUTOMÀTIC APLICAT AL BLACKJACK

Pau Amorós Faro
2n Batxillerat B
Bernat Vidal
Treball de Recerca II



Figura 1 - Las Vegas Strip. Bennet, A. (2 de novembre 2022). *The Las Vegas Strip: The Complete Guide*. <https://www.tripsavvy.com/what-is-the-las-vegas-strip-1679296>



Figura 2 - Monedas i or. Newheiser, J. (20 de juny del 2022). *5 consejos bíblicos para abandonar el afán por las riquezas.*

<https://www.coalicionporelevangelio.org/articulo/5-consejos-biblicos-para-abandonar-el-afan-por-las-riquezas/>



Figura 3 - Enfonsament econòmic. Muñoz, S. (11 d'agost 2021). *Seis señales de una bancarrota*.

<https://www.elsiglodetorreon.com.mx/noticia/2021/seis-senales-de-una-bancarota.html>

APRENENTATGE AUTOMÀTIC APLICAT AL BLACKJACK

Pau Amorós Faro

ÍNDEX

0. Motivacions

1. Blackjack.

2. Estratègia bàsica.

3. Python.

4. Formes de programació:

- Q-Learning.**
- Mètode Montecarlo.**

5. Programació.

6. Conclusió final.

MOTIVACIONES

BLACKJACK

Tenir cartes, el valor numèric de les quals sigui el més alt possible, però no major que 21.



Figura 4 - Taula de blackjack. Casino de Barcelona (s.d.) *Tabla de Blackjack: saber cuándo apostar.*

<https://www.casinobarcelona.es/blog/tabla-de-blackjack-saber-cuando-apostar/>

Normes de joc

Crupier contra jugador.

El crupier haurà de treure cartes fins a arribar als 17 o més i parar-se.

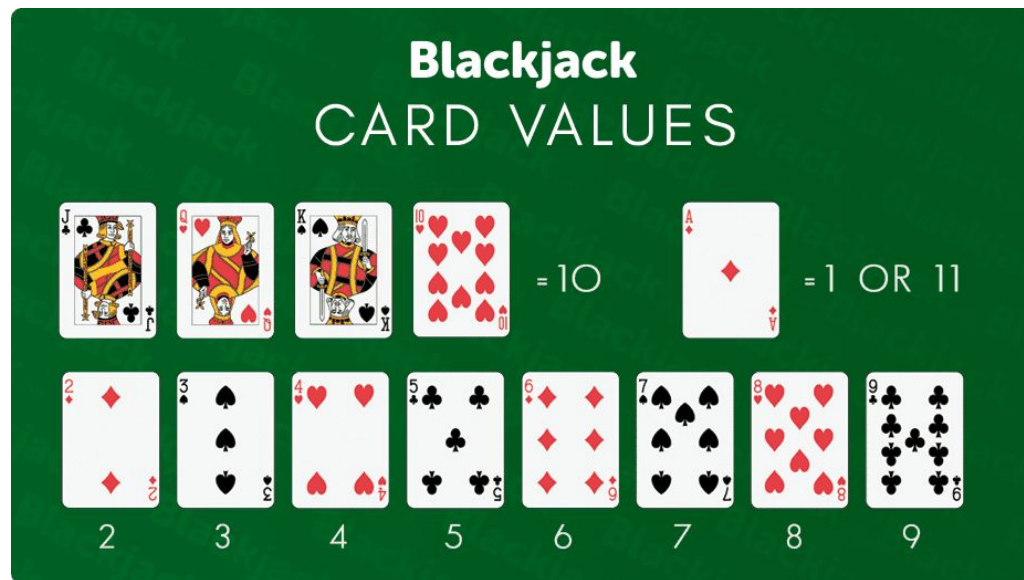


Figura 5 - Valors de les cartes al blackjack. OSU (s.d.). *User manual*. <https://u.osu.edu/sdp12d1/user-manual/>

Partida

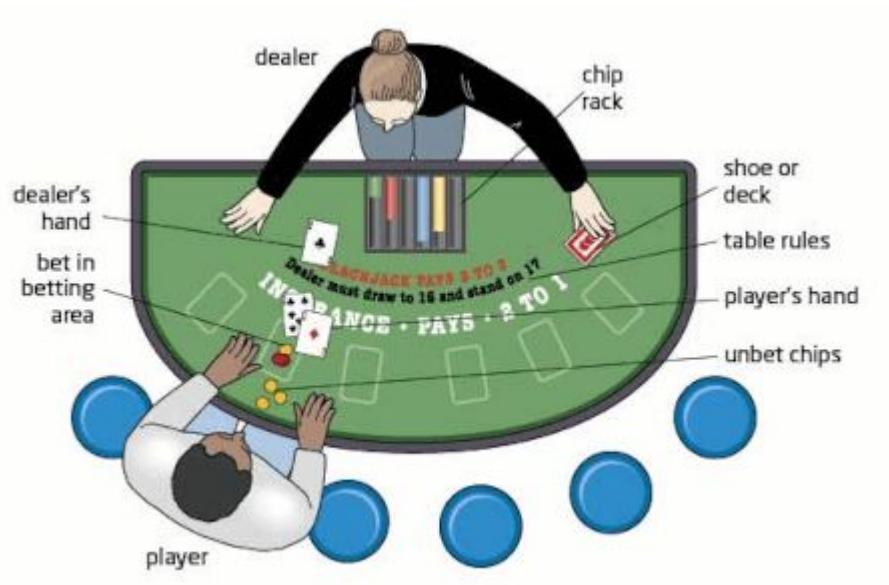


Figura 6 - Taula de blackjack. Las Vegas How-to Gaming Guide (s.d.). How to play blackjack.

<https://www.vegashowto.com/blackjack>

Jugador aposta.

Es reparteixen les cartes.

El jugador decideix com jugar.

El crupier treu la seva mà.

Desemborsament (3 a 2).

ESTRATÈGIA BÀSICA

Estratègia que maximitza la mitjana de victòries d'un jugador.

BLACKJACK BASIC STRATEGY

		Dealer's Face Up Card										
Team Casino		2	3	4	5	6	7	8	9	10	A	
Player's Hand (hard)	5	H	H	H	H	H	H	H	H	H	H	
	6	H	H	H	H	H	H	H	H	H	H	
	7	H	H	H	H	H	H	H	H	H	H	
	8	H	H	H	H	H	H	H	H	H	H	
	9	H	Dh	Dh	Dh	Dh	H	H	H	H	H	
	10	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H	H	H	
	11	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H	H	
	12	H	H	S	S	S	H	H	H	H	H	
	13	S	S	S	S	S	H	H	H	H	H	
	14	S	S	S	S	S	H	H	H	H	H	
	15	S	S	S	S	S	H	H	H	H	H	
Player's Hand (soft)	A2	H	H	H	Dh	Dh	H	H	H	H	H	
	A3	H	H	H	Dh	Dh	H	H	H	H	H	
	A4	H	H	Dh	Dh	Dh	H	H	H	H	H	
	A5	H	H	Dh	Dh	Dh	H	H	H	H	H	
	A6	H	Dh	Dh	Dh	Dh	H	H	H	H	H	
	A7	S	Ds	Ds	Ds	Ds	S	S	H	H	H	
	A8	S	S	S	S	S	S	S	S	S	S	
	A9	S	S	S	S	S	S	S	S	S	S	
	AT	S	S	S	S	S	S	S	S	S	S	
Player's Hand (splits)	2,2	Sp	Sp	Sp	Sp	Sp	Sp	H	H	H	H	
	3,3	Sp	Sp	Sp	Sp	Sp	Sp	H	H	H	H	
	4,4	H	H	H	Sp	Sp	H	H	H	H	H	
	5,5	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H	H	H	
	6,6	Sp	Sp	Sp	Sp	Sp	H	H	H	H	H	
	7,7	Sp	Sp	Sp	Sp	Sp	Sp	H	H	H	H	
	8,8	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	
	9,9	Sp	Sp	Sp	Sp	Sp	S	Sp	Sp	S	S	
	T,T	S	S	S	S	S	S	S	S	S	S	
	A,A	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	

H Hit

S Stand

Sp Split

Dh Double if allowed, otherwise hit

Ds Double if allowed, otherwise stand

T= Ten, Jack, Queen Or King

Figura 7 - Estratègia bàsica blackjack. Hughes, J (s.d.). *Using the Blackjack Strategy*.

<https://www.teamcasino.com/blackjack-strategy-guide>

PYTHON

Creat per Guido van Rossum.

Usat en projectes d'aprenentatge automàtic.

Biblioteques com TensorFlow o Pytorch.

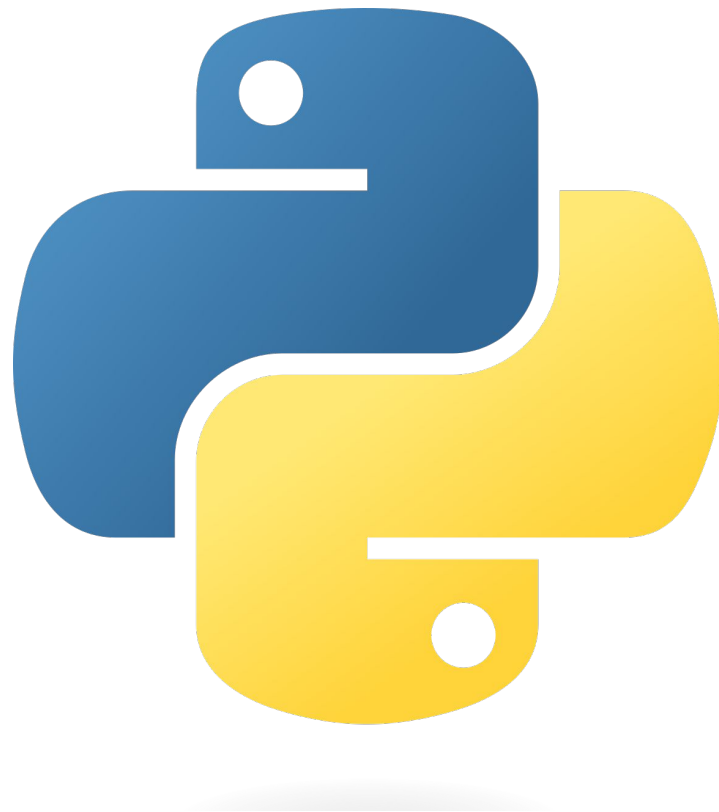


Figura 8 - Símbol de Python. Juni (Juny 2020). *What is Python coding?*
<https://junilearning.com/blog/guide/what-is-python-101-for-students/>

MÉTHODE MONTECARLO

Ús de nombres aleatoris i probabilitat per investigar problemes.

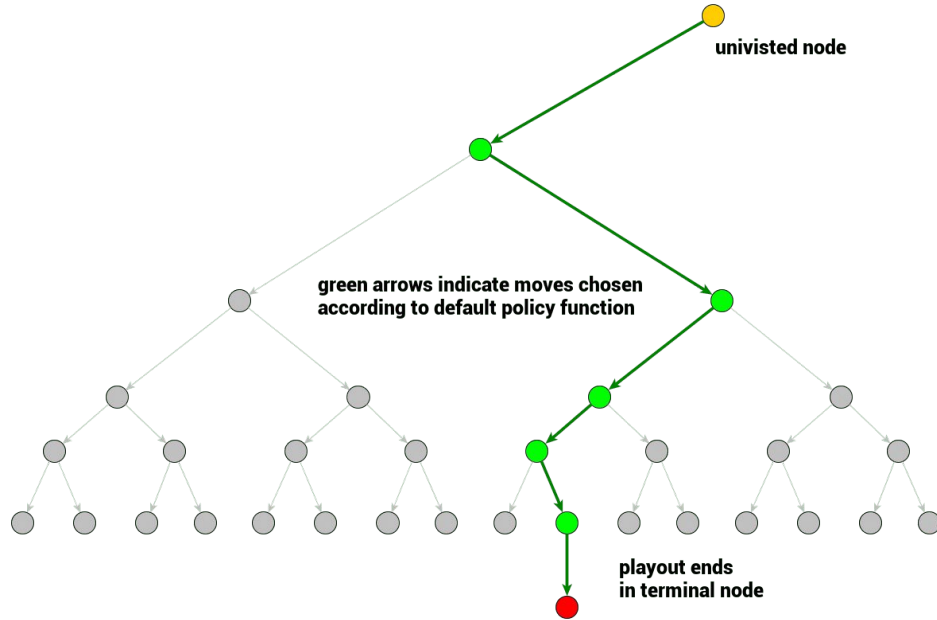


Figura 9 - Arbre de cerca Montecarlo. Das, Dj (24 septembre 2018). *Montecarlo tree search*.

<https://thirdeyedata.ai/monte-carlo-tree-search-beginners-guide/>

Múltiples aplicacions en
esdeveniments amb final concret,
com per exemple jocs de taula.

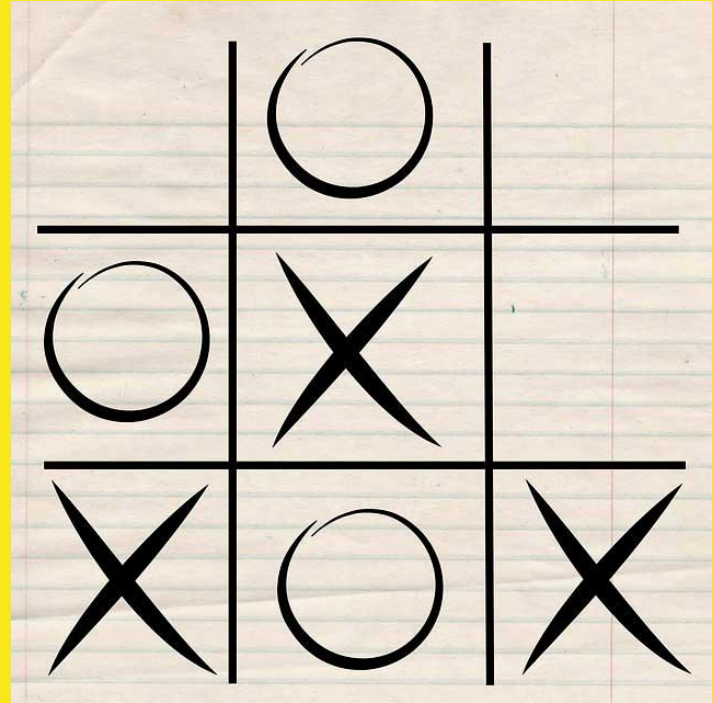


Figura 10 - Partida de tres en raya. Pequeocio (s.d.). *Tres en raya.*

<https://www.pequeocio.com/tres-en-raya/>

Q-LEARNING

Tipus d'aprenentatge automàtic on el feedback de l'entorn d'una simulació s'utilitza per optimitzar el procés d'aprenentatge segons l'equació Q.

$$\text{New } Q(s, a) = Q(s, a) + \alpha [R(s, a) + \gamma \max_{a'} Q'(s', a') - Q(s, a)]$$







-  New Q Value for that state and the action
-  Learning Rate
-  Reward for taking that action at that state
-  Current Q Values
-  Maximum expected future reward given the new state (s') and all possible actions at that new state.
-  Discount Rate

Figura 11 - Equació Q. ADL (3 setembre 2018). *An introduction to Q-Learning: reinforcement learning*.

<https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc/>

Es pot aprendre encara que es realitzi una acció subòptima “A”.

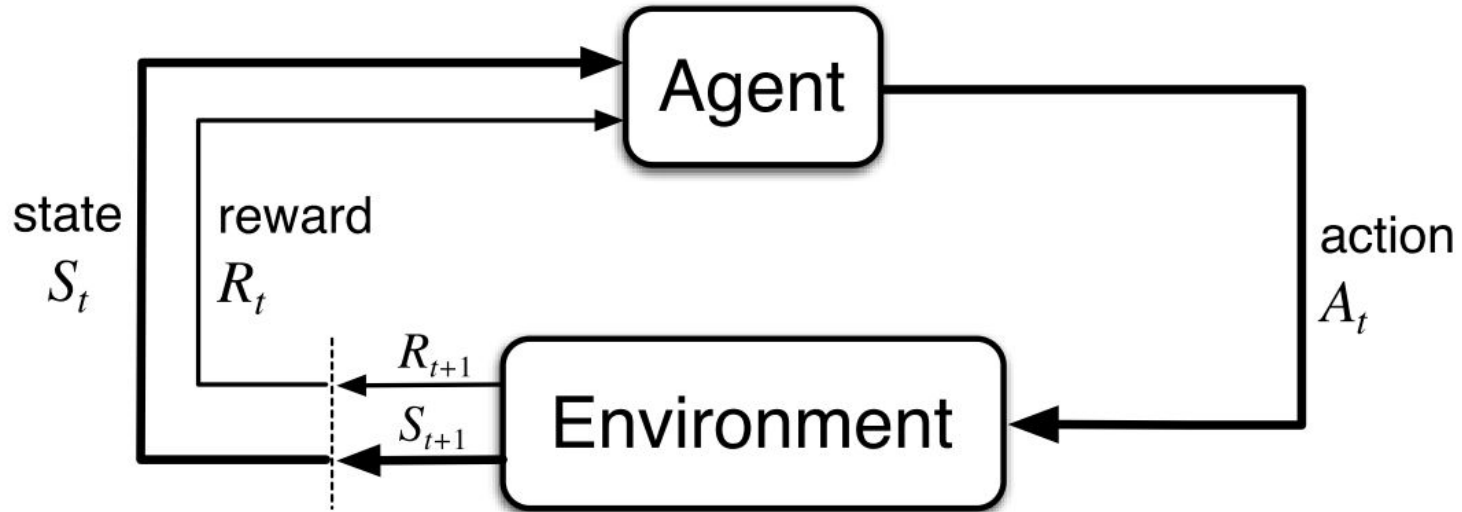


Figura 12 - Procés d'aprenentatge. Kraushek, P (20 septembre 2021). *Reinforcement Learning*.

<https://www.mcg.ai/post/reinforcement-learning>

PROGRAMACIÓ

REPRESENTACIÓ DELS ESTATS

Total actual del jugador

Carta visible del crupier

ACCIÓ

Hit (demanar una carta més)

Stick (finalitzar el torn)

RECOMPENSA:

Guanyar \Rightarrow 1

Perdre \Rightarrow 0

Les recompenses es donen al final de l'episodi.

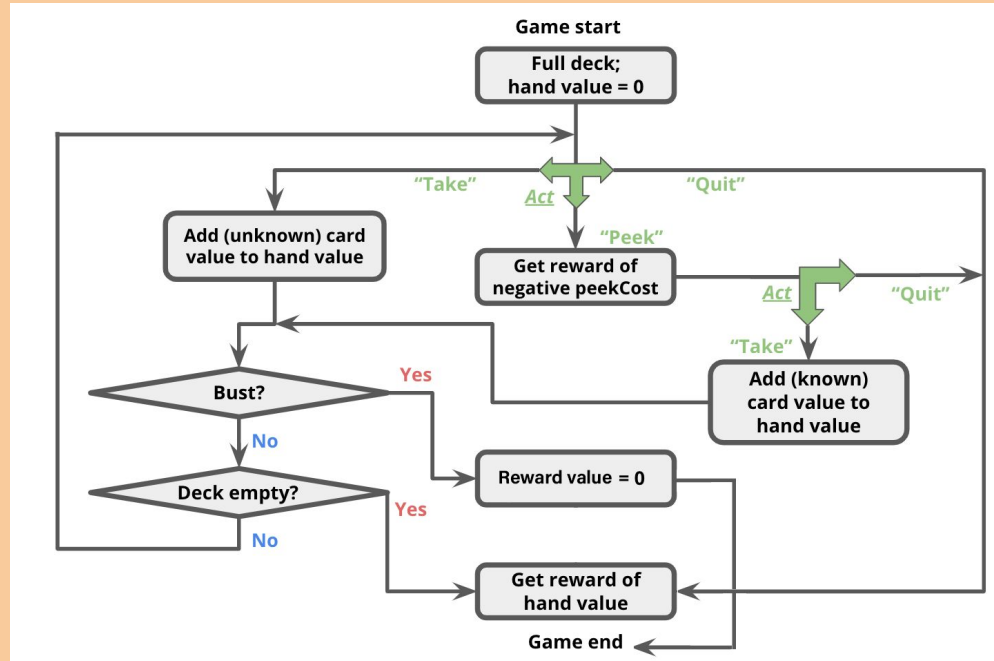


Figura 13 - Fluxograma d'una partida de blackjack. Hong, H (2019). *Peeking Blackjack*.

<https://stanford-cs221.github.io/autumn2019/assignments/blackjack/index.html>

PROGRAMA 1

OBJECTIUS

Es pot crear una IA que aprengui a jugar a blackjack?

Tindrà resultats favorables?



Figura 14 - Logotip RL Card. RL Card (s.d.). *RL Card: A toolkit for reinforcement learning in Card games.* <https://rlcard.org/>

PSEUDOCODI

1. Importar llibreries.
2. Importar l'entorn del blackjack (programat pels creadors de la llibreria).
3. Determinar les característiques de l'agent seguint les restriccions de l'entorn.
4. Establir que l'agent creat és l'únic a l'entorn.
5. Importar components crucials per l'aprenentatge (programats pels creadors de la llibreria).

```
[1]: import rlcard  
     from rlcard.agents import DQNAgent
```

```
[3]: env = rlcard.make("blackjack")
```

```
[4]: agent = DQNAgent(  
      num_actions=env.num_actions,  
      state_shape=env.state_shape[0],  
      mlp_layers=[64,64],  
  )
```

```
[5]: env.set_agents([agent])
```

```
[6]: from rlcard.utils import (  
      tournament,  
      reorganize,  
      Logger,  
      plot_curve,  
  )
```

Figura 15 - Captura de codi del meu ordinador.

Figura 16 - Captura de codi del meu ordinador.

PSEUDOCODI

1. Determinar que el bucle d'entrenament sirà de x episodis.
2. Fer que es gravin la trajectòria (acció) i els pagaments (recompensa) per a cada episodi.
3. Reorganitzar les dues variables mencionades per la màxima eficiència (funció de la llibreria `reorganize()`). Actualitzar l'agent amb aquests resultats.
4. Cada 50 episodis el rendiment es guarda a la classe `Logger` i s'emmagatzema a un arxiu CSV.
5. Es representa un gràfic del rendiment utilitzant els valors de l'arxiu CSV.

```

[7]: with Logger("experiments/leduc_holdem_dqn_result/") as logger:
    for episode in range(1000):

        trajectories, payoffs = env.run(is_training=True)

        trajectories = reorganize(trajectories, payoffs)

        for ts in trajectories[0]:
            agent.feed(ts)

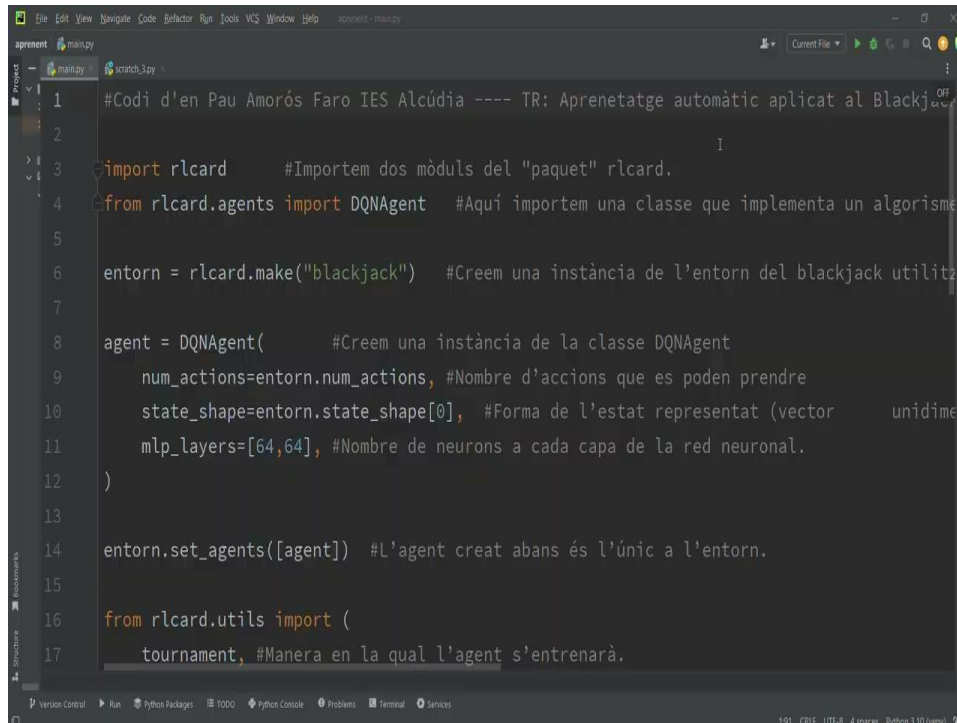
        if episode % 50 == 0:
            logger.log_performance(
                env.timestep,
                tournament(
                    env,
                    10000,
                )[0]
            )

    csv_path, fig_path = logger.csv_path, logger.fig_path

```

Figura 17 - Captura de codi del meu ordinador.

EXECUCIÓ



```
1 #Codi d'en Pau Amorós Faro IES Alcúdia ---- TR: Aprenetatge automàtic aplicat al Blackj...
2
3 import rlc card #Importem dos mòduls del "paquet" rlc card.
4 from rlc card.agents import DQN Agent #Aquí importem una classe que implementa un algorisme
5
6 entorn = rlc card.make("blackjack") #Creem una instància de l'entorn del blackjack utilitz
7
8 agent = DQN Agent( #Creem una instància de la classe DQN Agent
9     num_actions=entorn.num_actions, #Nombre d'accions que es poden prendre
10    state_shape=entorn.state_shape[0], #Forma de l'estat representat (vector unidime
11    mlp_layers=[64,64], #Nombre de neurons a cada capa de la red neuronal.
12 )
13
14 entorn.set_agents([agent]) #L'agent creat abans és l'únic a l'entorn.
15
16 from rlc card.utils import (
17     tournament, #Manera en la qual l'agent s'entrenarà.
```

Figura 18 - Vídeo de l'execució del primer programa gravat des del meu ordinador.

RESULTATS

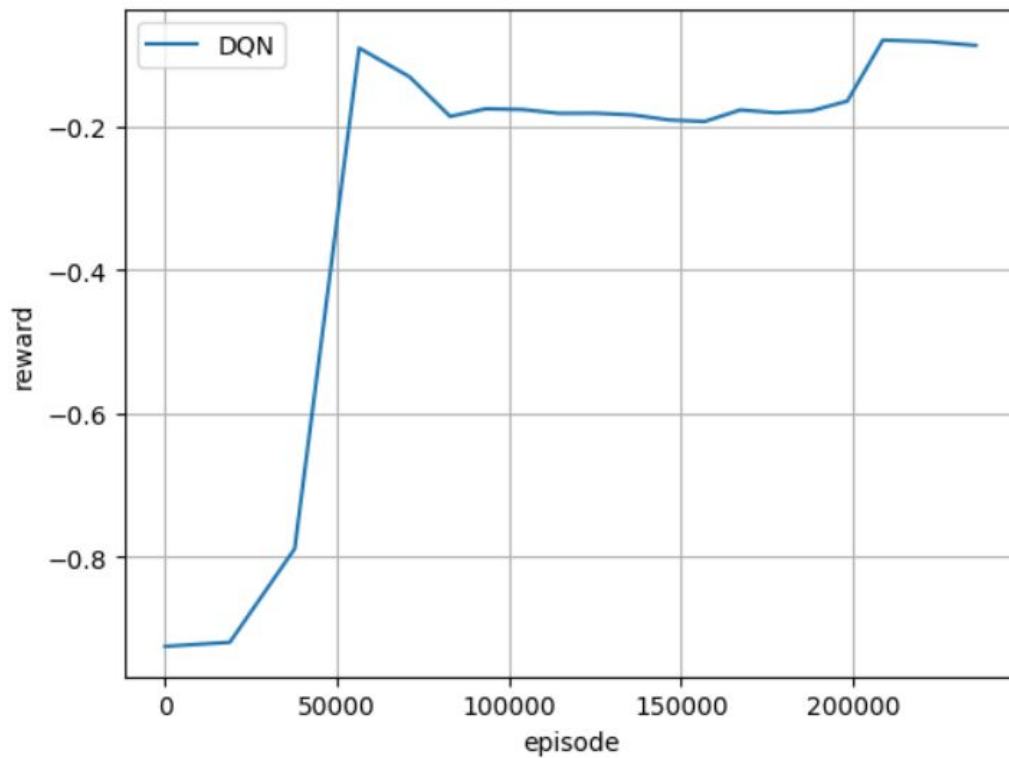


Figura 19 - Gràfica de la funció Q del primer programa.

CONCLUSIÓ 1

A mesura que el nombre d'episodis incrementa, la recompensa augmenta.
Arriba un punt on la progressió s'estanca.

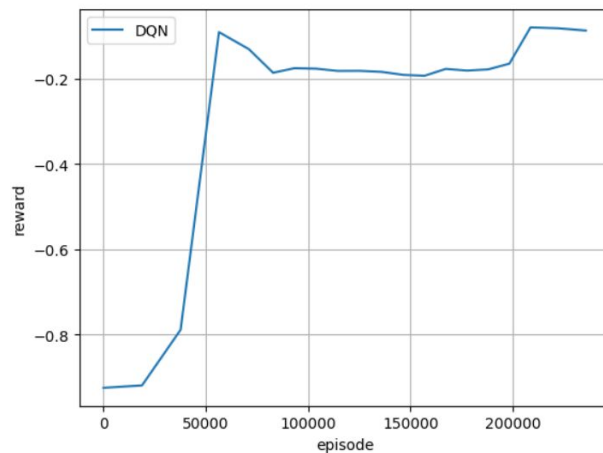


Figura 19 - Gràfica de la funció Q del primer programa.

PROGRAMA 2

OBJECTIUS

Quina és l'aproximació de joc òptima?

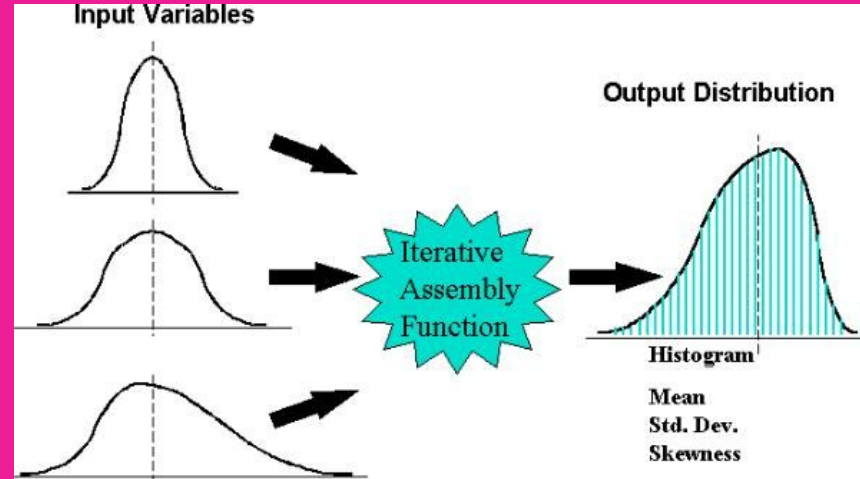


Figura 20 - Distribució de l'output. Trade GPD Strategy (18 de gener 2023). How can Monte Carlo simulations be used in financial models. <https://www.quora.com/How-can-Monte-Carlo-simulation-be-used-in-financial-models>

PSEUDOCODI GENERAL

1. Importar llibreries (random i matplotlib.pyplot).
2. Definir les accions (1=demanar carta, 0=quedar-se igual).
3. Crear una classe que defineixi el blackjack com a joc.
4. Crear una classe per a determinar com s'estructura l'estat i actualitza.
5. Realitzar les simulacions Monte-Carlo.
6. Concentrar els resultats de les simulacions per determinar l'estratègia òptima.
7. Representar-la a gràfics utilitzant (matplotlib.pyplot).

```

67 class Estat:
68
69     def __init__(self, jugador_sum, crupier_visible, as_11):
70         self.jugador_sum = jugador_sum
71         self.crupier_visible = crupier_visible
72         self.as_11 = as_11
73         self.n_DEMANAR_UNA_CARTA_MÉS = 1
74         self.n_QUEDAR_IGUAL = 1
75         self.Q_DEMANAR_UNA_CARTA_MÉS_total = 0
76         self.Q_QUEDAR_IGUAL_total = 0
77         self.policy = QUEDAR_IGUAL
78
79     def actualitza(self, recompensa, acció):
80         if acció == QUEDAR_IGUAL:
81             self.n_QUEDAR_IGUAL = self.n_QUEDAR_IGUAL + 1
82             self.Q_QUEDAR_IGUAL_total = self.Q_QUEDAR_IGUAL_total + recompensa
83         else:
84             self.n_DEMANAR_UNA_CARTA_MÉS = self.n_DEMANAR_UNA_CARTA_MÉS + 1
85             self.Q_DEMANAR_UNA_CARTA_MÉS_total = self.Q_DEMANAR_UNA_CARTA_MÉS_total + recompensa
86
87         if self.Q_DEMANAR_UNA_CARTA_MÉS_total / float(self.n_DEMANAR_UNA_CARTA_MÉS) > self.Q_QUEDAR_IGUAL_total / float(self.n_stick):
88             self.policy = DEMANAR_UNA_CARTA_MÉS
89         else:
90             self.policy = QUEDAR_IGUAL
91

```

Figura 21 - Captura de codi del meu ordinador.

PSEUDOCODI

1. Inicialitzar la classe “Estat” amb els paràmetres propis d’una partida de blackjack.
2. Si no es demana una carta, sumar 1 al comptador de quedar-se igual i utilitzar la recompensa en qüestió.
3. Si es demana una carta, fer el mateix amb “cartamés”.
4. Comprova quina política té una recompensa mitjana és més alta i actuar segons aquesta.


```

96
97 v def monteCarloES(nombre_episodis=7500000):
98     states = [Estat(i, j, l) for i in range(11, 22) for j in range(1, 11) for l in reversed(range(2))]
99 v     for i in range(0, nombre_episodis):
100         s = random.choice(states)
101         episodi = []
102         bj = BlackJack(s.jugador_sum, s.crupier_visible, s.as_11)
103         acció= random.randint(0, 1)
104         episodi.append([s, acció])
105 v         while True:
106             jugador_sum, crupier_visible, as_11, recompensa, joc_acaba = bj.torn(acció)
107 v             if joc_acaba == False:
108                 s = states[getEstatIdx(jugador_sum, crupier_visible, as_11)]
109                 acció= s.policy
110                 episodi.append([s, acció])
111 v             else:
112 v                 for e in episodi:
113                     e[0].actualitza(recompensa, e[1])
114                 break
115     return states
116

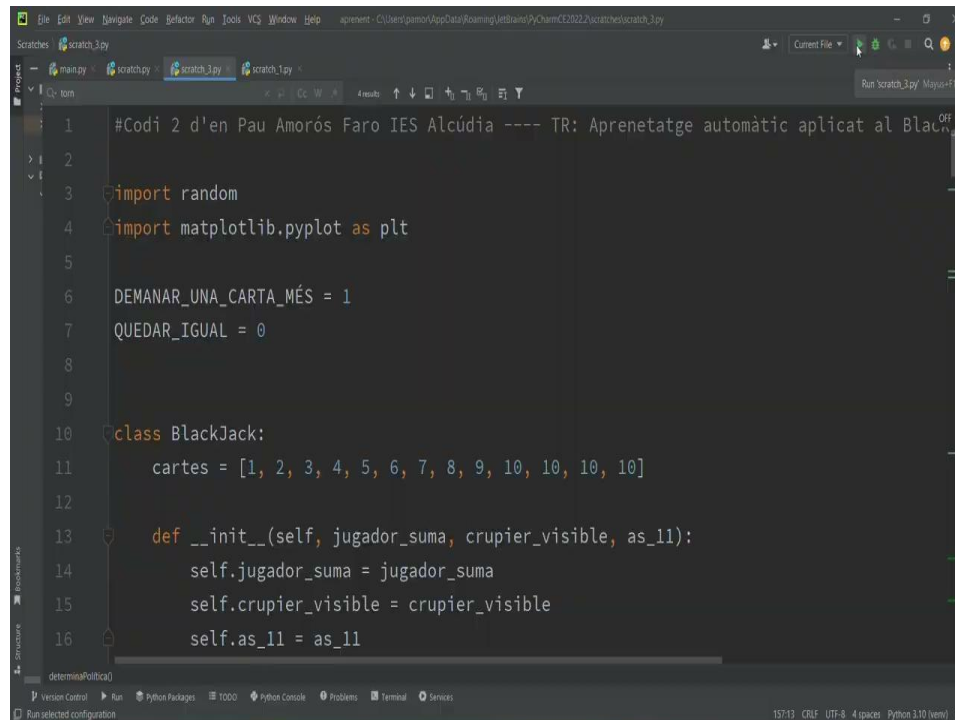
```

Figura 22 - Captura de codi del meu ordinador.

PSEUDOCODI

1. Crear una llista tridimensional per guardar els possibles estats segons la suma del jugador, del crupier i si tenen un as o no.
2. Entrar en un bucle del nombre d'episodis seleccionats on es comença per crear una partida de blackjack a partir de la classe "Estat".
3. Generar una acció aleatòria (0 / 1).
4. Executar torns de joc fins que la partida acabi actualitzant les accions i estats produïts.
5. Actualitza les recompenses d'acord amb les accions.

EXECUCIÓ

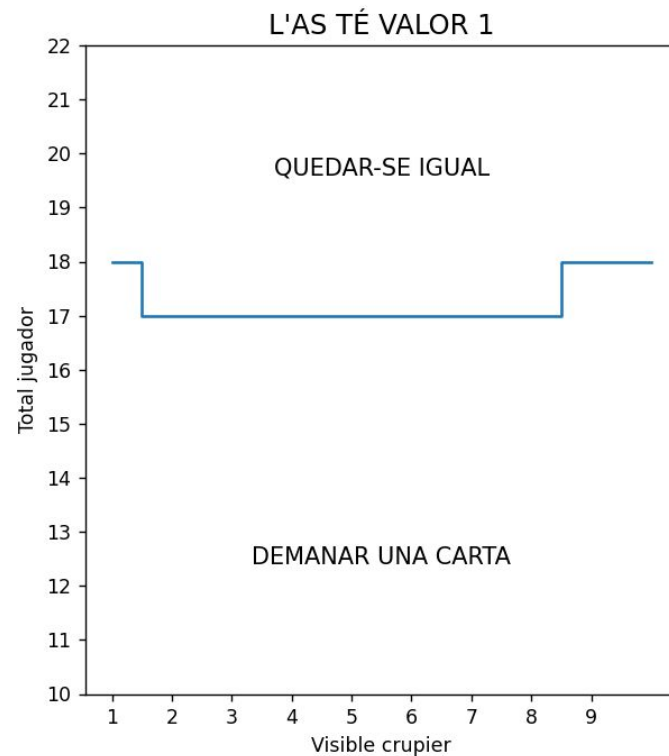
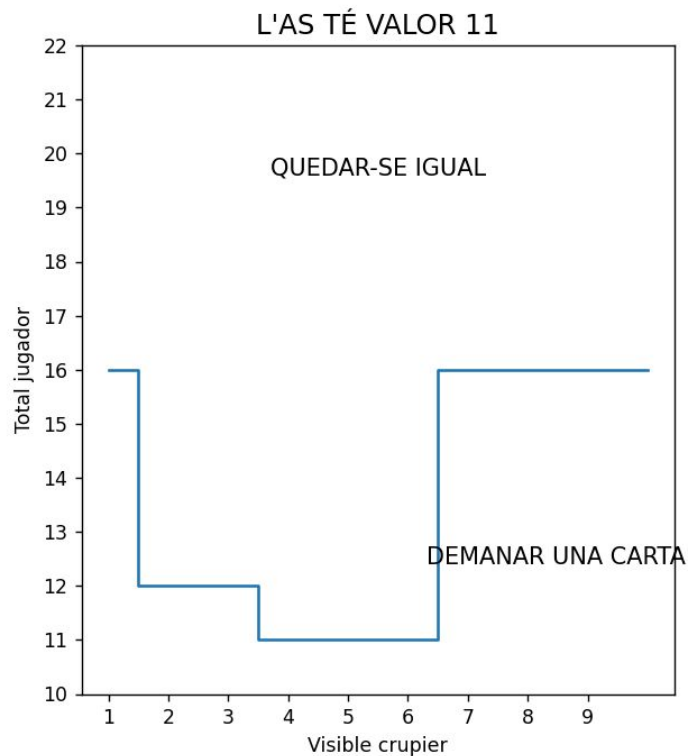


```
1 #Codi 2 d'en Pau Amorós Faro IES Alcúdia ---- TR: Aprenetatge automàtic aplicat al BlackJack
2
3 import random
4 import matplotlib.pyplot as plt
5
6 DEMANAR_UNA_CARTA_MÉS = 1
7 QUEDAR_IGUAL = 0
8
9
10 class BlackJack:
11     cartes = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10]
12
13     def __init__(self, jugador_suma, crupier_visible, as_11):
14         self.jugador_suma = jugador_suma
15         self.crupier_visible = crupier_visible
16         self.as_11 = as_11
```

Figura 23 - Vídeo de l'execució del segon programa gravat des del meu ordinador.

RESULTATS

Figura 24 - Resultats del segon programa.



CONCLUSIÓ 2

Es pot obtenir l'estratègia de joc òptima del blackjack utilitzant la programació.

Lleugeres discrepàncies amb l'estratègia tradicional, la causa de les quals és desconeguda.

Figura 24 - Resultats del segon programa.

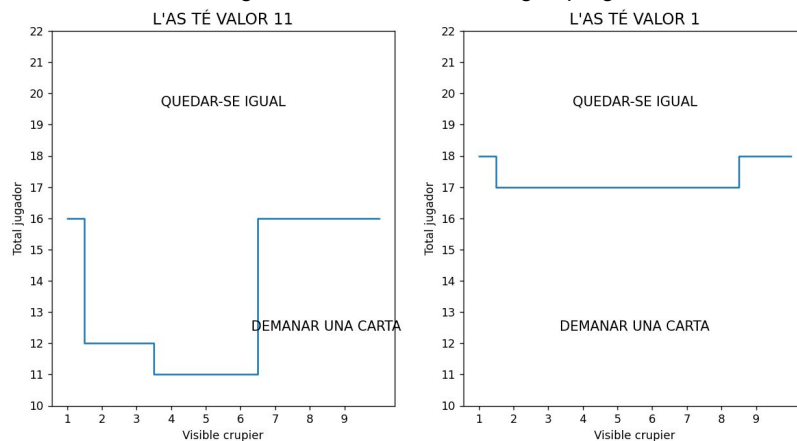


Figura 7 - Estratègia bàsica blackjack. Hughes, J (s.d.). *Using the Blackjack*

Strategy. <https://www.teamcasino.com/blackjack-strategy-guide>

BLACKJACK BASIC STRATEGY

Dealer's Face Up Card

Team	Casino	2	3	4	5	6	7	8	9	10	A
Player's Hand (hard)	5	H	H	H	H	H	H	H	H	H	H
	6	H	H	H	H	H	H	H	H	H	H
	7	H	H	H	H	H	H	H	H	H	H
	8	H	H	H	H	H	H	H	H	H	H
	9	H	Dh	Dh	Dh	Dh	H	H	H	H	H
	10	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H	H
	11	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H	H
	12	H	H	S	S	S	S	H	H	H	H
	13	S	S	S	S	S	S	S	H	H	H
	14	S	S	S	S	S	S	S	H	H	H
	15	S	S	S	S	S	S	S	H	H	H
Player's Hand (soft)	A2	H	H	H	Dh	Dh	H	H	H	H	H
	A3	H	H	H	Dh	Dh	H	H	H	H	H
	A4	H	H	Dh	Dh	Dh	H	H	H	H	H
	A5	H	H	Dh	Dh	Dh	H	H	H	H	H
	A6	H	Dh	Dh	Dh	Dh	H	H	H	H	H
	A7	S	Ds	Ds	Ds	Ds	S	S	H	H	H
	A8	S	S	S	S	S	S	S	S	S	S
	A9	S	S	S	S	S	S	S	S	S	S
	AT	S	S	S	S	S	S	S	S	S	S

Dealer's Face Up Card

Team	Casino	2	3	4	5	6	7	8	9	10	A
Player's Hand (splits)	2,2	Sp	Sp	Sp	Sp	Sp	Sp	H	H	H	H
	3,3	Sp	Sp	Sp	Sp	Sp	Sp	H	H	H	H
	4,4	H	H	H	Sp	Sp	H	H	H	H	H
	5,5	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H	H	H
	6,6	Sp	Sp	Sp	Sp	Sp	H	H	H	H	H
	7,7	Sp	Sp	Sp	Sp	Sp	H	H	H	H	H
	8,8	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp
	9,9	Sp	Sp	Sp	Sp	Sp	S	Sp	Sp	S	S
	T,T	S	S	S	S	S	S	S	S	S	S
	A,A	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp

H Hit

S Stand

Sp Split

Dh Double if allowed, otherwise hit

Ds Double if allowed, otherwise stand

T= Ten, Jack, Queen Or King

CONCLUSIONS

L'aplicació de l'aprenentatge automàtic a blackjack ha demostrat ser un enfocament prometedori per millorar el rendiment dels jugadors.

Gran potencial de la IA per ser aplicada a altres jocs i escenaris.

La banca sempre té l'avantatge matemàtic en el moment de jugar.

Es pot crear programa que aprengui a jugar a blackjack i doni una estratègia òptima.