UNIVERSITATEA TEHNICA
DIN CLUJ-NAPOCA
CENTRUL UNIVERSITAR NORD DIN BAIA MARE
FACULTATEA DE INGINERIE

DEPARTAMENTUL DE INGINERIE ELECTRICĂ, ELECTRONICĂ ȘI CALCULATOARE
PROGRAMUL DE STUDII CALCULATOARE

Dezvoltarea, Colectarea Datelor și Predicția deciziilor Jocului Blackjack Folosind Inteligența Artificială

LUCRARE DE LICENȚĂ

Absolvent: **Rareș Astaloș**

Coordonator științific: **Prof. Dr. Ing. Oliviu MATEI**

2025

UNIVERSITATEA TEHNICA
DIN CLUJ-NAPOCA
CENTRUL UNIVERSITAR NORD DIN BAIA MARE
FACULTATEA DE INGINERIE

DEPARTAMENTUL DE INGINERIE ELECTRICĂ, ELECTRONICĂ ȘI CALCULATOARE
PROGRAMUL DE STUDII CALCULATOARE

DECAN,
Conf. univ. dr. ing. Chiver Olivian

DIRECTOR DEPARTAMENT,
Conf. univ. dr. ing.

Absolvent: **Rareș Astaloș**

Dezvoltarea, Colectarea Datelor și Predicția deciziilor Jocului Blackjack Folosind Inteligența Artificială

1. **Enunțul temei:** *Dezvoltarea unei aplicații care să simuleze runde repetate de Blackjack folosind algoritmi de învățare automată, colectarea datelor pe baza algoritmului și dezvoltarea unei predicții pe baza datelor exportate a rezultatului optim în funcție de datele introduse de utilizator.*
2. **Conținutul lucrării:** *Introducere, Obiectivele proiectului, Studiu bibliografic, Analiză și Fundamentare teoretică, Proiectare în detaliu și implementare, Testare și validare, Manual de instalare si utilizare, Concluzii, Bibliografie.*
3. **Locul documentării:** *Centrul Universitar NORD din Baia Mare*
4. **Consultanți:**
5. **Data emiterii temei:** 29 martie 2024
6. **Data predării:** 10 iulie 2025

Absolvent: Astaloș Rareș

Coordonator științific: Prof. Dr. Ing. Oliviu MATEI

UNIVERSITATEA TEHNICA
DIN CLUJ-NAPOCA
CENTRUL UNIVERSITAR NORD DIN BAIA MARE
FACULTATEA DE INGINERIE

DEPARTAMENTUL DE INGINERIE ELECTRICĂ, ELECTRONICĂ ȘI CALCULATOARE
PROGRAMUL DE STUDII CALCULATOARE

Declarație pe propria răspundere privind autenticitatea lucrării de licență

Subsemnatul(a) Astaloș Rareș, legitimat(ă) cu Carte de Identitate seria XM nr. 061997 CNP 5030129245051, autorul lucrării Dezvoltarea, Colectarea Datelor și Predicția Deciziilor Jocului Blackjack Folosind Inteligența Artificială

elaborată în vederea susținerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea Calculatoare din cadrul Universității Tehnice din Cluj-Napoca, sesiunea Iulie a anului universitar 2024-2025, declar pe propria răspundere că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării și în bibliografie.

Declar că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență.

În cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de licență.*

Data

Nume, Prenume

Astaloș Rareș

Semnătura

Cuprins

Capitolul 1. Introducere

În ultimii ani, inteligența artificială a devenit o unealtă populară și foarte puternică, atât în zone de divertisment, cât și în cercetare sau dezvoltare a diferitelor domenii. Alegerea jocului Blackjack în baza testării capacității și limitelor inteligenței artificiale reprezintă un echilibru între noroc și strategie, între complexitate și simplitate, între intuiție și calcul matematic. Motivația alegerii temei este interesul personal pentru jocurile de strategie și dezvoltarea unor noi strategii de joc cât mai aproape de optim, maximizând șansele de câștig, dar și observarea capacității așa-zisului individ creat prin algoritmi de inteligență artificială în dezvoltarea acestuia. Jocul de cărți Blackjack este doar un exemplu în dezvoltarea strategiilor de joc, acest algoritm poate fi folosit în diferite jocuri de societate sau de strategie, scopul fiind același. În acest sens, învățarea prin întărire (Reinforcement Learning), o ramură a inteligenței artificiale care se potrivește subiectului, ajută la dezvoltarea rezultatelor prin antrenarea unui individ în luarea deciziilor proprii optime prin recompensarea acestuia în urma acțiunilor sale. Prin repetarea acestui proces, individul ajunge la un rezultat cât mai aproape de perfect, minimizând marja de eroare.

Jocul de cărți BlackJack, considerat de unii un joc de noroc, este o activitate de divertisment unde jucătorii au cea mai mare șansă de câștig în comparație cu celelalte jocuri de noroc dintr-un casino. Regulile jocului sunt simple, unul sau mai mulți jucători joacă un joc contra casei de casino printr-un dealer. Setul de cărți folosit este unul clasic de poker cu 52 de cărți, fiecare jucător primește două cărți vizibile după ce pachetul este amestecat. Dealer-ul de asemenea extrage două cărți, dar o carte dintre cele două este ascunsă.
Valorile cărților:

- Cărțile Jack, Queen și King au valoarea 10
- Cărțile de Ace(As) au valoarea 1 sau 11 în funcție de celelalte cărți.
- Restul cărților primesc valoarea proprie, de la 2 pana la 10.

Astfel, după ce runda a început, jucătorul are două decizii pe care le poate lua: Hit, care determină jucătorul să mai extragă o carte, și Stand, care determină jucătorul să rămână cu cărțile curente. După ce jucătorul a decis să mai extragă o carte, acesta poate lua încă o dată deciziile în funcție de cărțile pe care le are. Scopul final al jocului este de a ajunge cât mai aproape de suma cărților în valoare de 21 de puncte. Dacă jucătorul a extras o carte, iar suma cărților depășește 21 de puncte, rezultă pierderea rundei. Dacă jucătorul a luat decizia de Stand, dealer-ul este obligat să întoarcă cartea pentru a face-o vizibilă, el având regula de a extrage o carte până ajunge la valoarea minimă sau egală cu 17, chiar dacă aceasta rezultă în a pierde, astfel jocul având trei rezultate posibile: câștig, pierdere și egalitate.

Alte precizări:

- Dacă unul dintre jucători are în mână cartea de tip As, aceasta are valoarea implicită de 11 puncte, dar dacă în urma extragerii unei cărți care depășește valoarea 21, cartea de tip As se transformă din 11 puncte într-un punct.
- Am eliminat și decizia de Double, care permitea jucătorului să-și dubleze suma pariată pe primele două cărți, neexistând posibilitatea de a paria în exemplul nostru.

De asemenea, în parcursul acestui proiect am folosit un algoritm de învățare automată, și anume Q-learning.
Q-learning este un algoritm de învățare automată care permite unui individ, în cazul nostru jucătorul adversar dealer-ului, să învețe bazat pe propriile decizii, printr-o recompensă pentru fiecare acțiune. AI-ul explorează mediul, observă starea curentă, ia o acțiune, primește o recompensă și își actualizează cunoștințele pentru a îmbunătăți deciziile viitoare.

Pentru predicția deciziilor, unealta potrivită este Machine Learning (ML). ML este o ramură a Inteligenței Artificale (AI) care se axează pe construcția sistemelor care pot învăța, sau își pot îmbunătăți performanțele, în funcție de datele pe care le procesează. ML este folosit peste tot în jurul nostru, de la sisteme de recomandare până la interacțiunea cu băncile. Aceste sisteme pot detecta anomaliile în timp real, pot oferi predicții cu o acuratețe ridicată, toate cu ajutorul unui set de date, cu cât un set de date este mai vast, cu atât predicția este mai exactă.

Algoritmul Random Forest face parte din ramura ML. Acest algoritm combină rezultatul mai multor arbori de decizie pentru a ajunge la un singur rezultat. Acest algoritm s-a făcut cunoscut prin ușurința sa de utilizare și flexibilitate, deoarece se ocupă atât de problemele de clasificare, cât și de problemele de regresie.

A arborele de decizie este un algoritm util de învățare automată utilizat atât pentru sarcini de regresie, cât și pentru sarcini de clasificare. Denumirea „arborele de decizie" provine de la faptul că algoritmul continuă să împartă setul de date în porțiuni din ce în ce mai mici, până când datele au fost împărțite în instanțe unice, care sunt apoi clasificate. Dacă ar fi să vizualizați rezultatele algoritmului, modul în care sunt împărțite categoriile ar semăna cu un copac și multe frunze.

Capitolul 2. Obiectivele proiectului

Acest capitol reprezintă obiectivele propuse de-a lungul proiectului din punct de vedere tehnic. Proiectul are ca scop principal dezvoltarea unui sistem de inteligență artificială care să învețe și să ia decizii optime în jocul de Blackjack, cu scopul de a depăși șansele medii ale unui jucător uman.

Studiul își are ca scop și depășirea șanselor de câștig date de probabilitatea matematică prin intermediul inteligenței artificiale. Potrivit lui Catalin Barboianu în studiul "Probability Guide to Gambling: The Mathematics of Dice, Slots, Roulette, Baccarat, Blackjack, Poker, Lottery and Sport Bets" [1] șansele de câștig al jucătorului sunt de 42% cu diferite statistici de extragere a cărților, pe care o să le comparăm. De exemplu, Catalin Barboianu prezintă probabilitatea de a obține 20 de puncte din primele două cărți este de 10.25641%, probabilitatea de a obține 19 de puncte din primele două cărți este de 6.03318%, probabilitatea de a obține 17 de puncte din primele două cărți este de 6.48567%, iar studiul o să compare aceste statistici, cât și altele pentru a arăta o comparație.

	Dealer									
Player	2	3	4	5	6	7	8	9	10	Ace
19	0.3863	0.4044	0.4232	0.4395	0.4960	0.6160	0.5939	0.2876	−0.0187	−0.1155
18	0.1217	0.1483	0.1759	0.1996	0.2834	0.3996	0.1060	−0.1832	−0.2415	−0.3771
17	−0.1530	−0.1172	−0.0806	−0.0449	0.0117	−0.1068	−0.3820	−0.4232	−0.4644	−0.6386
16	−0.2928	−0.2523	−0.2111	−0.1672	−0.1537	−0.4148	−0.4584	−0.5093	−0.5752	−0.6657
15	−0.2928	−0.2523	−0.2111	−0.1672	−0.1537	−0.3698	−0.4168	−0.4716	−0.5425	−0.6400
14	−0.2928	−0.2523	−0.2111	−0.1672	−0.1537	−0.3213	−0.3719	−0.4309	−0.5074	−0.6123
13	−0.2928	−0.2523	−0.2111	−0.1672	−0.1537	−0.2691	−0.3236	−0.3872	−0.4695	−0.5825
12	−0.2534	−0.2337	−0.2111	−0.1672	−0.1537	−0.2128	−0.2716	−0.3400	−0.4287	−0.5504
11	0.2384	0.2603	0.2830	0.3073	0.3337	0.2921	0.2300	0.1583	0.0334	−0.2087

Table 3. Expectations, conditioned on the dealer's first card, for the player's profit, when the player draws.

Figura 2.1: Așteptări, condiționate de prima carte a dealerului, pentru profitul jucătorului, atunci când jucătorul trage.

Imaginea 2.1 este un exemplu de statistică preluată din studiul lui Catalin Barboianu [1] în care prezintă șansele de a extrage o carte anume, bazată pe cartea vizibilă a dealer-ului.

Pentru îndeplinirea acestui scop, primul obiectiv este dezvoltarea unei aplicații care să folosească algoritmi de inteligență artificială, mai exact Q-Learning, care să dezvolte o rată de câștig cât mai apropiată de cea matematică, de 42%. În sensul acesta, aplicația o să fie împărțită în pași simpli:

- implementarea cărților de joc
- adăugarea lor într-un pachet
- implementarea regulilor de joc
- implementarea logicii jocului
- implementarea dealer-ului
- implementarea individului
- implementarea algoritmului.

- simularea repetitivă a unei runde.
- exportarea datelor într-un tabel/bază de date.

De asemenea, implementarea unei interfețe grafice care să arate ascendența sau descendența rezultatelor pentru a avea o statistică vizibilă în timp real.

În acest scop, studiul își propune colectarea de date prin intermediul algoritmului, care să simuleze o rundă de BlackJack, pe care să o salveze ulterior într-o tabelă sau bază de date, repetând acest proces până în momentul în care setul de date este destul de larg încât să prezinte o acuratețe relevantă. În acest fel, eliminăm posibilitatea de eroare umană, cât și timpul de colectare a datelor.

Initial Hand Value	Has Ace	Number of Cards	Final Hand Value	Dealer Visible Card	Dealer Final Value	Number of Hits	Final Action	Result	Result Type
13	FALSE	4	23	10	18	2	TRUE	-1	bust
12	FALSE	3	18	7	20	1	FALSE	-1	normal
6	FALSE	4	23	9	20	2	TRUE	-1	bust
7	FALSE	3	15	2	21	1	FALSE	-1	normal
13	FALSE	3	17	3	19	1	FALSE	-1	normal
18	TRUE	3	17	6	17	1	FALSE	0	tie
9	FALSE	3	19	10	20	1	FALSE	-1	normal
10	FALSE	3	18	2	23	1	FALSE	1	normal
8	TRUE	3	19	7	18	1	FALSE	1	normal
20	FALSE	2	20	7	21	0	FALSE	-1	normal

Figura 2.2: Exemplu de date exportate.

Un alt scop al acestui studiu este evidențierea ușurinței și eficienței inteligenței artificiale, care poate învăța regulile unui joc, sau chiar a altor domenii cât mai aproape de perfecțiune, eliminând astfel eroarea umană. Acest lucru poate duce la analize complexe, colectare de date cât și alte date relevante în subiectul inițial. Astfel, inteligența artificială poate fi un instrument foarte important în eliminarea greșelilor, a anomaliilor sau a factorilor externi care pot influența rezultatele inițiale.

Un ultim scop al acestui proiect este prezentarea statisticilor care să ofere o comparație studiului prezentat anterior, și de asemenea oferirea unor predicții bazate pe cărțile extrase și pe cartea vizibilă a dealer-ului. În acest scop o să folosim algoritmi de Machine Learning, mai exact algoritmul Random Forest pentru a oferi o predicție a cărților care pot fi extrase. În sfârșit, proiectul va avea o interfață grafică care să-ți ofere posibilitatea de a introduce cărțile din propria mână și cartea vizibilă a dealer-ului, astfel să-ți ofere o statistică și o predicție a șanselor de câștig al fiecărei decizii.

Capitolul 3. Studiu bibliografic

3.1. Istoricul și fundamentele jocului Blackjack

3.1.1. Originea jocului

În articolul "The History of Blackjack" [2], există o teorie care sugerează că Blackjack-ul a fost inventat de romani. Această teorie este plauzibilă, având în vedere pasiunea romanilor pentru jocuri de noroc, deși rămâne neverificată. Se crede că romanii jucau acest joc folosind blocuri de lemn marcate cu diverse numere, în loc de cărți de hârtie.

Pe lângă numeroasele variante de Vingt-et-Un, popularitatea jocurilor de cărți s-a răspândit în toată America de Nord. Jocul de cărți a ajuns pe țărmurile americane în secolul al XVIII-lea, ajutat de coloniștii francezi. Cu toate acestea, s-a chinuit să câștige teren și să evolueze în Franța în timpul secolului al XIX-lea [3]. În schimb, în această perioadă, jocul a înflorit și a devenit popular în America. Până în 1820, jocul era prezent în New Orleans, în cazinourile legalizate. Interesant este că regulile de la acea vreme diferă de Blackjack-ul modern pe care îl recunoaștem astăzi. De exemplu, în versiunea anterioară de Blackjack, doar dealerul avea privilegiul de a dubla.

În această epocă, există o poveste notabilă despre Eleanor Dumont. Născută în Franța, a imigrat în America și a devenit o dealer pricepută, călătorind până când a înființat o sală de jocuri de noroc în Nevada City, California. În mod ironic, localul a fost numit Vingt-et-Un. Jucători din întreaga țară s-au adunat să concureze împotriva lui Eleanor, deoarece aceasta era considerată o figură unică printre dealerii de cărți.

3.1.2. Strategii clasice

Tacticile câștigătoare în Blackjack necesită ca jucătorul să joace fiecare mână în mod optim, iar o astfel de strategie ia întotdeauna în considerare cartea deschisă a dealerului [4]. Când cartea deschisă a dealerului este una bună, de exemplu un 7, 8, 9, 10 sau un as, jucătorul nu ar trebui să se oprească din tragerea cărților până când nu se atinge un total de 17 sau mai mult. Când cartea deschisă a dealerului este una slabă, 4, 5 sau 6, jucătorul ar trebui să se oprească din tragerea cărților imediat ce obține un total de 12 sau mai mult. Strategia aici este să nu iei niciodată o carte dacă există vreo șansă de a da faliment [5]. Dorința cu această mână slabă este de a lăsa dealerul să lovească și, sperăm, să depășească 21. În cele din urmă, când cartea deschisă a dealerului este una corectă, 2 sau 3, jucătorul ar trebui să se oprească cu un total de 13 sau mai mult.

Cu o mână slabă, strategia generală este de a continua să lovești până când se atinge un total de cel puțin 18. Astfel, cu un as și un șase (7 sau 17), jucătorul nu s-ar opri la 17, ci ar da lovitura.

Strategia de bază pentru dublarea cărților este următoarea: Cu un total de 11, jucătorul ar trebui să dubleze întotdeauna. Cu un total de 10, ar trebui să dubleze, cu excepția cazului în care dealerul arată o carte de zece sau un as. Cu un total de 9, jucătorul ar trebui să dubleze doar dacă cartea dealerului este corectă sau slabă (de la 2 la 6).

Pentru împărțire, jucătorul ar trebui să împartă întotdeauna o pereche deași sau de 8; cărțile de zece identice nu ar trebui împărțite și nici o pereche de 5, deoarece două cărți de 5 sunt un total de 10, ceea ce poate fi folosit mai eficient la dublarea cărților. Nici o pereche de 4 nu ar trebui împărțită, deoarece un total de 8 este un număr bun pentru a trage. În general, cărțile de 2, 3 sau 7 pot fi împărțite, cu excepția cazului în care dealerul are un 8, un 9, o carte de zece sau un as. În cele din urmă, cărțile de 6 nu ar trebui împărțite decât dacă cartea dealerului este slabă (de la 2 la 6).

3.2. Probabilitatea matematică în Blackjack

3.2.1. Probabilitățile de obținere a anumitor mâini

Pentru jocurile jucate cu pachete complete de cărți, prezentăm mai întâi probabilitățile asociate cu împărțirea cărților și predicțiile inițiale.[1] Probabilitățile de mai jos sunt calculate pentru jocuri care utilizează unul sau două pachete de cărți. Să analizăm probabilitățile de obținere a unei mâini inițiale favorabile (primele două cărți împărțite). Numărul total de combinații posibile pentru fiecare dintre cele două cărți este:

$$C(52, 2) = 1326 \quad \text{(pentru jocul cu 1 pachet)}, \tag{3.1}$$

$$C(104, 2) = 5356 \quad \text{(pentru jocul cu 2 pachete)}. \tag{3.2}$$

Probabilitatea de a obține un blackjack natural este:

$$P = \frac{8}{663} = 1{,}20663\% \quad \text{(joc cu 1 pachet)}, \tag{3.3}$$

$$P = \frac{16}{1339} = 1{,}19492\% \quad \text{(joc cu 2 pachete)}. \tag{3.4}$$

Probabilitatea de a obține un blackjack din primele două cărți este:

$$P = \frac{32}{663} = 4{,}82654\% \quad \text{(joc cu 1 pachet)}, \tag{3.5}$$

$$P = \frac{64}{1339} = 4{,}77968\% \quad \text{(joc cu 2 pachete)}. \tag{3.6}$$

În mod similar, putem calcula următoarele probabilități:

Pentru 20 de puncte:

$$P = \frac{68}{663} = 10{,}25641\% \quad \text{(joc cu 1 pachet)}, \tag{3.7}$$

$$P = \frac{140}{1339} = 10{,}45556\% \quad \text{(joc cu 2 pachete)}. \tag{3.8}$$

Pentru 19 puncte:

$$P = \frac{40}{663} = 6{,}03318\% \quad \text{(joc cu 1 pachet)}, \tag{3.9}$$

$$P = \frac{80}{1339} = 5{,}97460\% \quad \text{(joc cu 2 pachete)}. \tag{3.10}$$

Pentru 18 puncte:

$$P = \frac{43}{663} = 6{,}48567\% \quad \text{(joc cu 1 pachet)}, \tag{3.11}$$

$$P = \frac{87}{1339} = 6{,}4973\% \quad \text{(joc cu 2 pachete)}. \tag{3.12}$$

Pentru 17 puncte:

$$P = \frac{16}{221} = 7{,}23981\% \quad \text{(joc cu 1 pachet)}, \tag{3.13}$$

$$P = \frac{96}{1339} = 7{,}16952\% \quad \text{(joc cu 2 pachete)}. \tag{3.14}$$

O mână inițială bună (cu care poți rămâne) poate fi un blackjack sau o mână de 20, 19 sau 18 puncte. Probabilitatea obținerii unei astfel de mâini este:

$$P = \frac{32 + 68 + 40 + 43}{663} = \frac{183}{663} = 27{,}60180\% \quad \text{(joc cu 1 pachet)}, \tag{3.15}$$

$$P = \frac{64 + 140 + 80 + 87}{1339} = \frac{371}{1339} = 27{,}70724\% \quad \text{(joc cu 2 pachete)}. \tag{3.16}$$

Probabilitățile evenimentelor prezise în timpul jocului sunt calculate pe baza cărților jucate (cărțile care apar) de la un anumit moment. Aceasta necesită numărarea anumitor cărți favorabile pentru dealer și pentru ceilalți jucători, precum și a cărților din propria mână. Spre deosebire de un joc de baccarat, unde se joacă maximum trei cărți pentru fiecare jucător, la blackjack se pot juca mai multe cărți la un anumit moment, mai ales când sunt mai mulți jucători la masă. Astfel, atât urmărirea, cât și memorarea anumitor cărți necesită o anumită abilitate și un antrenament prealabil din partea jucătorului.

3.2.2. Noțiunea de House Edge

Avantajul casei este un termen folosit în industria cazinourilor pentru a se referi la avantajul matematic pe care cazinoul îl are față de jucător în orice joc dat.[6] În contextul blackjack-ului, avantajul casei este procentul din fiecare pariu pe care cazinoul se așteaptă să-l câștige pe termen lung. Aceasta înseamnă că, dacă pariezi 100 USD pe o mână de blackjack cu un avantaj al casei de 1%, cazinoul se așteaptă să câștige 1 USD din acel pariu pe termen lung. [7]

Avantajul casei la blackjack este determinat de o serie de factori, inclusiv regulile specifice ale jocului, numărul de pachete de cărți utilizate și nivelul de calificare al jucătorului. În general, avantajul casei la blackjack variază de la 0,5% la 2%, unele variante ale jocului având un avantaj al casei mai mare decât altele. Înțelegerea conceptului de avantaj al casei este esențială pentru a vă îmbunătăți șansele de câștig la blackjack, deoarece vă poate ajuta să luați decizii informate cu privire la momentul în care să pariați, să rămâneți, să dublați sau să împărțiți cărțile.

3.3. Introducere în inteligența artificială aplicată jocurilor

3.3.1. AI în jocuri: scurt istoric

Deep Blue a fost un supercomputer de mare putere dedicat exclusiv jocului de șah construit de IBM și care a susținut în anii 1996 și 1997 două meciuri, de câte 6 partide

fiecare, împotriva campionului mondial de șah Garry Kasparov. După meciul din 1996, în care Deep Blue a fost învins, calculatorul a fost îmbunătățit masiv și la data de 11 mai 1997, computerul încheia cu o victorie cel de-al doilea turneu, învingând campionul mondial după două victorii, o înfrângere și trei partide încheiate la egalitate. Kasparov a sugerat că IBM ar fi trișat și a cerut revanșa, dar compania a refuzat și a dezasamblat supercomputerul. [8]

AlphaGo este un program de calculator care joacă jocul de societate Go. A fost dezvoltat de DeepMind Technologies, cu sediul la Londra, o filială achiziționată de Google. Versiunile ulterioare ale AlphaGo au devenit din ce în ce mai puternice, inclusiv o versiune care concura sub numele Master. După retragerea din jocurile competitive, AlphaGo Master a fost succedat de o versiune și mai puternică, cunoscută sub numele de AlphaGo Zero, care a fost complet autodidactă, fără a învăța din jocurile umane. AlphaGo Zero a fost apoi generalizat într-un program cunoscut sub numele de AlphaZero, care a jucat jocuri suplimentare, inclusiv șah și shogi. AlphaZero a fost la rândul său succedat de un program cunoscut sub numele de MuZero, care învață fără a fi învățat regulile. [9]

AlphaGo și succesorii săi folosesc un algoritm de căutare în arbore Monte Carlo [10] pentru a-și găsi mutările pe baza cunoștințelor dobândite anterior prin învățarea automată, în special printr-o rețea neuronală artificială (o metodă de învățare profundă) printr-un antrenament extins, atât din jocul uman, cât și din jocul pe calculator. O rețea neuronală este antrenată să identifice cele mai bune mutări și procentele de câștig ale acestor mutări. Această rețea neuronală îmbunătățește puterea căutării în arbore, rezultând o selecție mai puternică a mutărilor în următoarea iterație.

În octombrie 2015, într-un meci împotriva lui Fan Hui, AlphaGo original a devenit primul program de Go pe calculator care a învins un jucător profesionist de Go uman fără handicap pe o tablă de joc de dimensiuni normale de 19×19. În martie 2016, l-a învins pe Lee Sedol într-un meci de cinci jocuri, prima dată când un program de Go pe calculator a învins un profesionist de 9 dan fără handicap. Deși a pierdut în fața lui Lee Sedol în al patrulea joc, Lee a demisionat în jocul final, dând un scor final de 4 jocuri la 1 în favoarea AlphaGo. Ca recunoaștere a victoriei, AlphaGo a primit un premiu onorific de 9 dan din partea Asociației Coreene Baduk. Pregătirile și meciul provocator cu Lee Sedol au fost documentate într-un film documentar intitulat, de asemenea, AlphaGo, regizat de Greg Kohs. Victoria AlphaGo a fost aleasă de Science ca unul dintre vicecampionii la Descoperirea Anului pe 22 decembrie 2016.

La Summitul Future of Go din 2017, versiunea Master a AlphaGo l-a învins pe Ke Jie, jucătorul numărul unu mondial la acea vreme, într-un meci de trei jocuri, în urma căruia AlphaGo a primit gradul profesionist de 9-dan din partea Asociației Chineze Weiqi.

După meciul dintre AlphaGo și Ke Jie, DeepMind a retras AlphaGo, continuând în același timp cercetările în domeniul inteligenței artificiale în alte domenii. AlphaGo Zero, un jucător autodidact, a obținut o victorie cu 100-0 împotriva versiunii competitive timpurii a AlphaGo, iar succesorul său, AlphaZero, a fost perceput ca fiind cel mai bun jucător mondial la Go până la sfârșitul anilor 2010. [11]

3.3.2. Reinforcement Learning (RL) – teorie de bază.

Învățarea prin consolidare (RL) reprezintă un domeniu multidisciplinar în cadrul învățării automate și al controlului optim, concentrându-se pe modul în care un agent inteligent ar trebui să acționeze într-un mediu dinamic pentru a maximiza un semnal de recompensă [12]. Este recunoscută ca una dintre cele trei paradigme fundamentale ale

învățării automate, alături de învățarea supravegheată și învățarea nesupravegheată.

Spre deosebire de învățarea supravegheată, învățarea prin consolidare nu necesită furnizarea de perechi de intrare-ieșire etichetate și nici nu necesită corectarea explicită a acțiunilor suboptimale. Accentul principal este pus pe atingerea unui echilibru între explorare (investigarea zonelor neexplorate) și exploatare (utilizarea cunoștințelor existente) cu obiectivul maximizării recompenselor cumulative, care pot fi incomplete sau întârziate. Această căutare a echilibrului este denumită dilema explorare-exploatare.

De obicei, mediul este reprezentat ca un proces decizional Markov (MDP), deoarece mulți algoritmi de învățare prin consolidare utilizează tehnici de programare dinamică. Distincția cheie dintre abordările tradiționale de programare dinamică și algoritmii de învățare prin consolidare constă în faptul că aceștia din urmă nu presupun cunoașterea unui model matematic precis al procesului decizional Markov și sunt concepuți pentru a aborda MDP-uri mari în care metodele exacte devin impracticabile. [13]

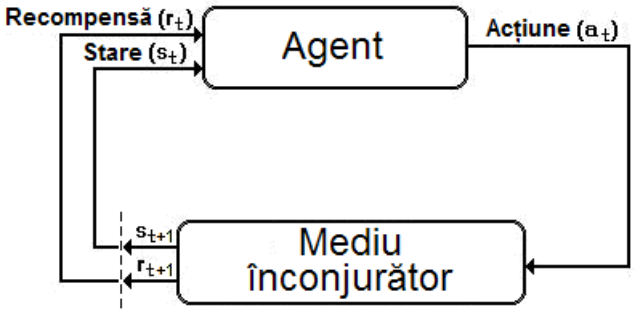

Figura 3.1: Schema Reinforcement Learning

3.3.3. Algoritmul Q-Learning

Q-learning este un algoritm de învățare prin consolidare bazat pe valori. Scopul Q-learning este de a învăța politica optimă de selecție a acțiunilor pentru un agent care interacționează cu un mediu. Agentul învață acest lucru prin actualizarea unui tabel de valori - numit Q-tabel - unde fiecare intrare reprezintă valoarea efectuării unei anumite acțiuni într-o anumită stare.[14]

Algoritmul Q-learning folosește următoarea formulă pentru a actualiza valoarea Q pentru o pereche stare-acțiune:

$$Q(s,a) = Q(s,a) + \alpha \left[R + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a) \right] \tag{3.17}$$

Unde:

- Q(s, a) este valoarea Q pentru starea s și acțiunea a.
- alfa este rata de învățare, care controlează cât de mult informațiile noi suprascriu informațiile vechi.
- R este recompensa imediată pentru luarea acțiunii a în starea s.
- gamma este factorul de reducere, reprezentând importanța recompenselor viitoare.
- $max_a Q(s', a')$ este valoarea Q maximă pentru următoarea stare s', reprezentând cea mai bună recompensă posibilă obținută din acea stare.

3.3.4. Exemple de utilizare a RL în jocuri de noroc

Acest studiu [15] scris de Diane Litman servește ca o prezentare generală introductivă a învățării prin consolidare (RL), o tehnică prin care un agent învață să ia decizii corecte prin implicarea în încercări repetate și primirea de recompense din mediul înconjurător. Obiectivul este ca agentul să învețe independent, prin experiență, cum să se comporte pentru a obține cele mai bune rezultate pe termen lung. Sunt oferite exemple simple, cum ar fi un joc în care un agent trebuie să atingă o țintă sau să selecteze cea mai profitabilă monedă. Studiul demonstrează că RL poate fi o metodă eficientă pentru învățarea comportamentelor inteligente chiar și fără o înțelegere detaliată a modului în care funcționează mediul în care operează agentul.

3.4. Machine Learning aplicat în analiză și predicție

3.4.1. Machine Learning: scurtă introducere

Machine Learning [16] (ML) este un domeniu de studiu în inteligența artificială care se ocupă de dezvoltarea și studiul algoritmilor statistici care pot învăța din date și pot generaliza la date nevăzute și, astfel, pot efectua sarcini fără instrucțiuni explicite. Într-o subdisciplină a învățării automate, progresele în domeniul învățării profunde au permis rețelelor neuronale, o clasă de algoritmi statistici, să depășească multe abordări anterioare de învățare automată în ceea ce privește performanța.

ML își găsește aplicații în multe domenii, inclusiv procesarea limbajului natural [17], viziunea computerizată, recunoașterea vorbirii, filtrarea e-mailurilor, agricultura și medicina. Aplicarea ML la problemele de afaceri este cunoscută sub numele de analiză predictivă.

Statistica și metodele de optimizare matematică (programare matematică) constituie fundamentele învățării automate. Extragerea datelor este un domeniu de studiu conex, concentrat pe analiza exploratorie a datelor (EDA) prin învățare nesupravegheată.

Dintr-un punct de vedere teoretic, învățarea probabil aproximativ corectă oferă un cadru pentru descrierea învățării automate.

3.4.2. Diferențele dintre supervised learning, undsupervised learning și Reinforcement Learning

1. Învățarea supravegheată se ocupă de două sarcini principale: regresia și clasificarea. Învățarea nesupravegheată se ocupă de problemele de clustering și extragere a regulilor asociative. În timp ce învățarea prin consolidare se ocupă de exploatare sau explorare, procesele decizionale ale lui Markov, învățarea politicilor, învățarea profundă și învățarea valorii.
2. Învățarea supravegheată lucrează cu date etichetate, iar aici modelele de date de ieșire sunt cunoscute sistemului. Însă, învățarea nesupravegheată se ocupă de date neetichetate, unde ieșirea se bazează pe colecția de percepții. În timp ce în procesul decizional al învățării prin consolidare al lui Markov, agentul interacționează cu mediul în etape discrete.
3. Numele în sine spune că învățarea supravegheată este extrem de supravegheată. Iar învățarea nesupravegheată nu este supravegheată. Spre deosebire de învățarea prin consolidare, aceasta este mai puțin supravegheată, ceea ce depinde de agent în determinarea ieșirii.
4. Datele de intrare în învățarea supravegheată sunt date etichetate. În timp ce în

învățarea nesupravegheată datele sunt neetichetate. Datele nu sunt predefinite în învățarea prin consolidare.

5. Învățarea supravegheată prezice pe baza unui tip de clasă. Învățarea nesupravegheată descoperă modele subiacente. Iar în Învățarea prin Reîntărire, agentul de învățare funcționează ca un sistem de recompensă și acțiune.
6. Învățarea supravegheată mapează datele etichetate la ieșiri cunoscute. Pe de altă parte, învățarea nesupravegheată explorează tipare și prezice ieșirile. Învățarea prin Reîntărire urmează o metodă de încercare și eroare.
7. Pe scurt, în Învățarea Supravegheată, scopul este de a genera formule bazate pe valorile de intrare și ieșire. În Învățarea Nesupravegheată, găsim o asociere între valorile de intrare și le grupăm. În Învățarea prin Reîntărire, un agent învață prin feedback întârziat, interacționând cu mediul. [18]

Criteria	**Supervised ML**	**Unsupervised ML**	**Reinforcement ML**
Definition	Learns by using labelled data	Trained using unlabelled data without any guidance.	Works on interacting with the environment
Type of data	Labelled data	Unlabelled data	No – predefined data
Type of problems	Regression and classification	Association and Clustering	Exploitation or Exploration
Supervision	Extra supervision	No supervision	No supervision
Algorithms	Linear Regression, Logistic Regression, SVM, KNN etc.	K – Means, C – Means, Apriori	Q – Learning, SARSA
Aim	Calculate outcomes	Discover underlying patterns	Learn a series of action
Application	Risk Evaluation, Forecast Sales	Recommendation System, Anomaly Detection	Self Driving Cars, Gaming, Healthcare

Figura 3.2: Diferențele dintre Supervised-Unsupervised-Reinforcement Learning

3.4.3. Algoritmul Random Forest

Algoritmii de tip Random Forest au trei hiperparametri principali, care trebuie setați înainte de antrenament. Aceștia includ dimensiunea nodului, numărul de arbori și numărul de caracteristici eșantionate [19]. De acolo, clasificatorul de tip Random Forest poate fi utilizat pentru a rezolva probleme de regresie sau de clasificare.

Algoritmul de tip Random Forest este alcătuit dintr-o colecție de arbori de decizie, prin imaginea 3.3, iar fiecare arbore din ansamblu este compus dintr-un eșantion de date extras dintr-un set de antrenament cu înlocuire, numit eșantion bootstrap. Din acel eșantion de antrenament, o treime este pusă deoparte ca date de testare, cunoscute sub numele de eșantion out-of-bag (oob), la care vom reveni mai târziu. O altă instanță de aleatorie este apoi injectată prin feature bagging, adăugând mai multă diversitate setului de date și reducând corelația dintre arborii de decizie. În funcție de tipul de problemă, determinarea predicției va varia. Pentru o sarcină de regresie, arborii de decizie individuali vor fi mediați, iar pentru o sarcină de clasificare, un vot majoritar - adică cea mai frecventă variabilă categorică - va produce clasa prezisă. În cele din urmă, eșantionul oob este apoi utilizat pentru validare încrucișată, finalizând predicția respectivă. [20]

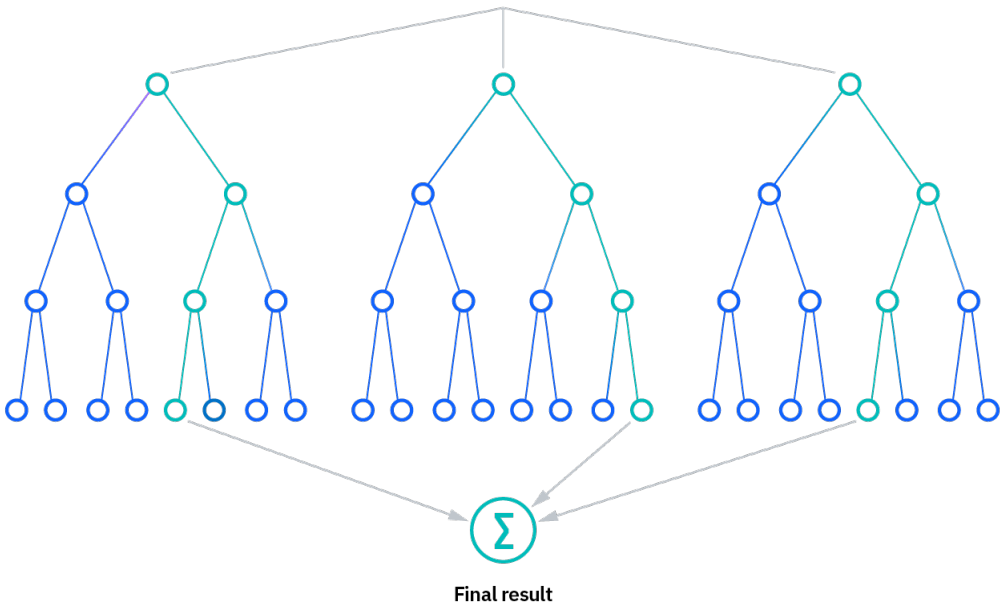

Figura 3.3: Random Forest – Învățare prin agregarea arborilor

Capitolul 4. Analiză și fundamentare teoretică

4.1. Algoritmul de Reinforcement Learning

4.1.1. Q-Learning explicat și cum va fi implementat

Q-Learning este un algoritm de învățare automată prin RL. Acest algoritm se folosește de o variabilă numită Q-Value asociată fiecărei variante de stare-acțiune.

$$Q(s,a) = Q(s,a) + \alpha \left[R + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a) \right] \tag{4.1}$$

În contextul nostru, Starea(s) este formată din valoarea totală a cărților din mâna individului, de prezența unui As și de valoarea cărții vizibile a dealer-ului.

Acțiunea(a) sunt variantele pe care le poate lua individul: Hit pentru a extrage o carte și Stand pentru a se opri.

4.1.2. Sistemul de recompensă

Sistemul de recompense care a fost implementat în algoritm este structurat în două părți:

1. Recompense intermediare. Acest tip de recompensă este acordat în timpul în care se ia o decizie și are scopul de a îmbunătăți procesul individului de învățare. Astfel, AI-ul primește +0.5 puncte pentru blackjack, -0.5 pentru bust și +0.3 bonus pentru o mână puternică (>=17 și <= 20).
2. Recompense finale. Acest tip de recompense este acordat la finalul rundei și reflectă deznodământul rundei. AI-ul primește ca recompensă +1.0 pentru victorie normală, +1.5 pentru Blackjack(victorie normală + recompensa intermediară), -1.0 pentru pierdere (inclusiv bust) și 0.0 pentru egalitate.

4.1.3. Procesul de Învățare

Procesul de învățare se bazează pe formula 4.1 care este împărțită astfel:

- α (alpha) este rata de învățare și influențează cât de mult informația noua înlocuiește informația veche in Q-Table.
- γ (gamma) este factorul de discount, și determină cât de mult sunt valorificate recompensele din viitor față de cele imediate
- r este recompensa imediată
- s' este starea următoare
- a' este acțiunea următoare

4.1.4. Explorare vs. Exploatare

Pentru a optimiza explorarea unor noi strategii, cu exploatarea cunoștințelor acumulate, s-a implementat o strategie epsilon-greedy. Aceasta funcționează prin selectarea acțiunii considerate a fi cea mai bună (exploatare) de cele mai multe ori, dar cu o probabilitate mică (epsilon), alege o acțiune aleatorie (explorare). Astfel, cu probabilitatea ϵ se alege o acțiune aleatorie(explorare), iar cu probabilitatea 1-ϵ se alege acțiunea cu cea

mai mare Q-Value (exploatare). Această strategie ajută ca individul să exploreze eficient pentru descoperirea unor noi strategii, cât și să exploateze cunoștințele acumulate pentru a maximiza recompensele pe termen lung.

4.1.5. Optimizarea strategiei

Strategia de joc a individului este actualizată permanent prin actualizarea Q-Value bazată pe experiența acumulată, ajustarea parametrilor de învățare(alpha, gamma și epsilon) pentru optimizare și evaluarea strategiei prin calcularea ratei de câștig.

4.2. Modelul de Decizie Markov (MDP)

Modelul de decizie Markov (MDP) este un framework folosit pentru modelarea proceselor de a lua o decizie în condiții de incertitudine. Acesta reprezintă un sistem care evoluează în timp, unde deciziile luate în prezent influențează atât recompensele imediate cât și stările viitoare ale algoritmului. MDP este util în situațiile în care deciziile trebuie luate secvențial, rezultatele acțiunilor sunt într-o măsură aleatorii și există un motiv pentru optimizare.

În contextul studiului, MDP oferă un framework matematic pentru a modela starea curentă a jocului (State), acșiunile posibile ale individului (Action) și tranzițiile dintre stări.

4.2.1. Componentele MDP

Stările, (s') în contextul studiului, reprezintă starea sistemului și este împărțită astfel:

- Valoarea totală a mâinii jucătorului (2-21)
- Prezența unui As în mână (boolean)
- Valoarea cărții vizibile a dealer-ului (2-11)
- Numărul total de cărți prezente în mână
- Dacă există o pereche de cărți cu aceeași valoare (boolean)
- Dacă tipul cărții vizibile a dealer-ului este As. (boolean)

Setul de acțiuni posibile este binar: Hit pentru a cere o carte și Stand pentru a se opri. Funcția de tranziție este împărțită în distribuția cărților din pachet (după amestecarea acestora), respectarea regulilor jocului și extragerea cărților. Tranzițiile dintre stări sunt determinate de acțiunea individului, probabilitățile extragerii unei anumite cărți și regulile fixate ale dealer-ului.

Astfel, logica de actualizare a stării este actualizată permanent prin colectarea datelor și se folosește de această metodă pentru a optimiza datele pe care individul le folosește în optimizarea sa.

4.2.2. Proprietățiile Markov

Modelul implementat trebuie să respecte următoarele reguli Markov:

1. Proprietatea Markov de Ordinul 1. Starea curentă conține toate informațiile necesare pentru a prezice starea viitoare, istoricul anterior al jocului nu influențează deciziile viitoare (Această proprietate nu exclude învățarea din experiență) și deciziile se bazează doar pe starea curentă
2. Staționaritate. Probabilitățile de tranziție rămân constante în timp, regulile jocului nu se schimbă și distribuția cărților este staționară (se resetează la fiecare rundă).

4.2.3. Optimizarea Politicii

Optimizarea politicii se folosește de Reinforcement Learning, prin următorii parametrii:

- Epsilon: crează o balanță între explorare și exploatare.
- Rata de Învățare(alpha): determină cât de mult se actualizează Q-Value.
- Rata de Discount(gamma): determină importanța deciziilor viitoare în comparație cu cele anterioare.

Această politică este optimizată continuu și în permanență prin actualizarea Q-Table la fiecare acțiune intermediară a individului și încă o dată la sfârșitul rundei și prin folosirea epsilon-greedy pentru selecția acțiunilor. Această abordare permite individului să învețe din experientă și să-și actualizeze și optimizeze strategiile de joc.

4.3. Sistemul de Învățare

4.3.1. Parametrii de învățare

Algoritmul de învățare utilizează trei parametrii pentru a controla procesul de învățare:

1. Rata de Învățare (alpha) Determină cât de mult se actualizează Q-Value după fiecare experiență, o valoare mai mare înseamnă învățare mai rapidă dar mai instabilă iar o valoare mai mică înseamnă învățare mai lentă dar mai stabilă.
2. Factorul de Discount (gamma) determină importanța recompenselor viitoare în raport cu cele imediate, o valoare mai mare înseamnă că agentul ia în considerare mai mult consecințele pe termen lung iar o valoare mai mică înseamnă că agentul se concentrează mai mult pe recompensele imediate.
3. Epsilon controllează balanța între explorare și exploatare, Cu probabilitatea ϵ, agentul alege o acțiune aleatorie (explorare) iar cu probabilitatea 1-ϵ, agentul alege cea mai bună acțiune cunoscută (exploatare)

   Valoriile optime ale parametrilor o să fie afișate la capitolul de testare și validare, astfel, vom crea o clasă separată care va testa fiecare combinație a celor trei parametrii, de la 0.1 până la 0.9, care va antrena separat individul pentru a observa cea mai bună combinație de parametrii, în acest fel vom hyperparametriza și vom salva datele într-o tabelă pentru a putea fi comparate. Vom simula astfel 200.000 de runde de joc pentru fiecare combinație.

4.3.2. Procesul de Actualizare

Sistemul își actualizează propriile cunoștințe în cele două momente: actualizări intermediare și actualizări finale.

4.3.3. Procesul de Învățare

Mecanismul de învățare funcționează pe următoarele principii:

1. Învățare din experiență. Prin intermediul Q-Table cunoștințele sunt acumulate gradual, fiecare rundă oferind informații pentru îmbunătățirea individului, acesta învățând din propriile interacțiuni cu mediul(starea).
2. Adaptare continuă, strategia de joc fiind actualizată în constant pe baza rezultatelor.

3. Evaluarea performanței, prin afișarea ratei de câștig a totalului de runde și a ultimelor 100.000 de runde.

4.3.4. Optimizare

Sistemul include optimizări pentru îmbunătățirea individului în procesul de învățare, recompense intermediare, actualizări ale Q-Table și neîntreruperea procesului incremental și strategia epsilon-greedy pentru explorare-exploatare.

4.4. Modelul de Evaluare a Performanței

4.4.1. Sistemul de Tracking

Sliding window (Sw) care pe parcursul a 100.000 de runde permite evaluarea algoritmului pe termen lung și oferă o perspectivă asupra evoluției strategiei. La fiecare 1.000 de runde, se calculează rata de câștig, de egalitate și de înfrângere pe baza Sw. Aceste valori sunt afișate în consolă și pe graficul WinRateChart, pentru a urmări progresul AI-ului în timp real.

Structura de date este codificată simplu: 1 pentru victorie, 0 pentru egalitate, -1 pentru înfrângere.

4.4.2. Metrici de Performanță

1. Rata de câștig(Win Rate), oferă procentajul de victorie pentru toate rundele, pentru ultimele 100.000 de runde și oferă o măsură directă a eficienței strategiei.
2. Dealer Win Rate, oferă procentajul de victorie a dealer-ului, ajută la observarea eficientă a rezultatelor și overă context pentru evaluarea winrate-ului AI-ului.
3. Tie Rate reprezintă procentajul de egalitate, este important pentru observarea metricilor și reflectă strategia de evitare a riscului.

4.4.3. Vizualiarea Performanței

Graficul de winrate afișează în timp real rata evoluției individului, cât și rata de câștig a dealer-ului și rata de egalitate. Acesta monitorizarea vizuală a progresului și ajută la indentificarea tendințelor.

4.4.4. Analiză Detaliată

Înregistrarea în detaliu se face în structura GameRecord, fiecare rundă din ultimele 100.000 este stocată și capturează următoarele date:

- Valoarea inițială a mâinii (initialHandValue)
- Prezența Asului inițial (initialHasAce)
- Valoarea primei cărți (initialCard1Value)
- Valoarea celei de-a doua cărți (initialCard2Value)
- Valoarea cărții vizibile a dealerului (dealerVisibleCardValue)
- Dacă cartea vizibilă este As (dealerVisibleCardIsAce)
- 6 acțiuni posibile (actions[6])
- 6 valori ale cărților extrase (cardValues[6])
- Numărul de pachete folosite (numDecks)
- Numărul de hit-uri efectuate (numHits)
- Valoarea finală a mâinii (finalHandValue)
- Numărul total de cărți (finalNumCards)

- Numărul de Ași (numberOfAces)
- Prezența Asului în mâna finală (finalHasAce)
- Valoarea finală a mâinii dealerului (dealerFinalValue)
- Numărul total de cărți ale dealerului (dealerFinalNumCards)
- Prezența Asului în mâna finală a dealerului (dealerHasAceFinal)
- Rezultatul rundei (result: 1 pentru victorie, 0 pentru egalitate, -1 pentru pierdere)
- Tipul rezultatului (resultType: ”normal”, ”blackjack”, ”bust”, ”tie”)
- Dacă AI-ul a avut Blackjack (isBlackjackAI)
- Dacă AI-ul a dat bust (bustAI)
- Dacă dealerul a dat bust (bustDealer)

Aceste date ulterior sunt exportate într-un fișier .csv, care o să ofere un mediu optim pentru dezvoltarea statisticilor avansate care o să ne permită să analizăm rezultatele.

4.4.5. Optimizări teoretice

Am ales să implementăm un număr de pachete customizabil, deoarece în concluzia acestui studiu se vor compara rezultatele altor studii care se bazează pe diferite numere de pachete, de la 1 la 6. Acest factor influențează impactul strategiei și complexitatea crescută a individului, dar, de asemenea, oferă un rezultat mai realist.

4.5. Compararea rezultatelor

Pentru a valida rezultatul individului antrenat prin Reinforcement Learning, am comparat rezultatele generate de AI din fișierele .csv cu rezultatele studiului inițial. Într-un fișier de tip .ipynb folosind Jupyter Notebook am încărcat fișierele folosind biblioteca Pandas, am analizat distribuția rezultatelor și am calculat metrici de performanță, în comparație alăturată cu rezultatele studiului și am generat grafice pentru a putea vizualiza evoluția performanței individului.

4.6. Antrenarea AI-ului prin Random Forest: implementare și funcționabilitate

Pentru a construi un model de predicție al deciziei optime pentru fiecare acțiune din jocul Blackjack, am folosit algoritmul Random Forest Regressor, iar procesul de antrenare a fost împărțit astfel:

4.6.1. Prelucrarea datelor

Toate cele 6 fișiere .csv care conțineau 100.000 de date fiecare au fost concatenate într-un singur DataFrame. Am selectat coloanele relevante antrenării algoritmului și am afișat cele mai relevante coloane.

Datele au fost ulterior împărțite într-un set de antrenament și unul de testare(80%/20%) pentru a evalua performanța modelului.

Am folosit RandomForestRegressor cu 100 de arbori de decizie. Modelul a fost antrenat pe setul de antrenament și evaluat pe setul de testare, calculând scorul R^2 și scorul OOB(out-of-bag). Scorul R^2 măsoară cât de bine reușește modelul să explice variația valorilor reale din datele de test, scorul fiind cât mai aproape de valoarea 1 pentru a fi cât mai precis. Scorul OOB reprezintă o metodă de validare internă specifică algoritmului Random Forest. Astfel, scorul OOB arată cât de bine generalizează modelul pe date nevăzute, folosind doar datele ”lăsate pe dinafară” la fiecare arbore. Un scor OOB ridicat indică faptul că modelul nu supraînvață.

4.6.2. Salvarea și vizualizarea datelor

Modelul antrenat este salvat pentru o utilizare ulterioară și pentru a nu fi nevoită rularea scriptului de mai multe ori, astfel, o dată antrenat modelul rămâne salvat într-un fișier .joblib.
S-au generat grafice pentru comparația valorilor reale cu cele prezise. Acest model permite AI-ului să estimeze valoarea așteptată a fiecărei acțiuni posibile pe baza stării curente a jocului și să ia decizii optime pentru maximizarea șanselor de câștig.

4.7. Interfața grafică

Pentru a oferi un intermediar între modelul AI și experiența utilizatorului, am dezvoltat o interfață grafică folosind Python și librăriile Tkinter sau PyQt. Interfața grafică având următoarele funcționalități:

- Introducerea cărților
- Introducerea numerelor de pachete
- Afișarea rezultatelor celei mai bune decizii ca procentaj
- Afișarea celei mai bune decizii: HIT/STAND
- Un grafic tehnic pentru a afișa valoarea exactă a rezultatului

Capitolul 5. Proiectare în detaliu și implementare

Acest capitol împarte proiectul în două părți, proiectarea AI-ului care invață să joace folosing Reinforcement Learning și predicția folosind Random Forest cu interfața grafică. Capitolul explică funcțiile principale ale aplicației.

5.1. Proiectarea în detaliu al AI-ului cu Reinforcement Learning

5.1.1. Schema generală a aplicației

Aplicația este construită modular, fiecare clasă având un scop clar și concis în cadrul simulării jocului de Blackjack folosind AI. Structura claselor este următoarea:

- Deck: gestionează pachetul de cărți.
- Card: reprezintă cărțile individuale și valorile lor folosite în pachet.
- Hand: gestionează cărțile din mâinile individului și a dealer-ului.
- Dealer: implementează logica si strategia de joc a dealer-ului.
- AIPlayer: implementează logica de joc a individului.
- HyperparameterTuning: optimizează parametrii AI-ului.
- WinrateChart: generează un grafic pentru vizualizarea rezultatelor în timp real.
- Game: este clasa principală și coordonează jocul complet.

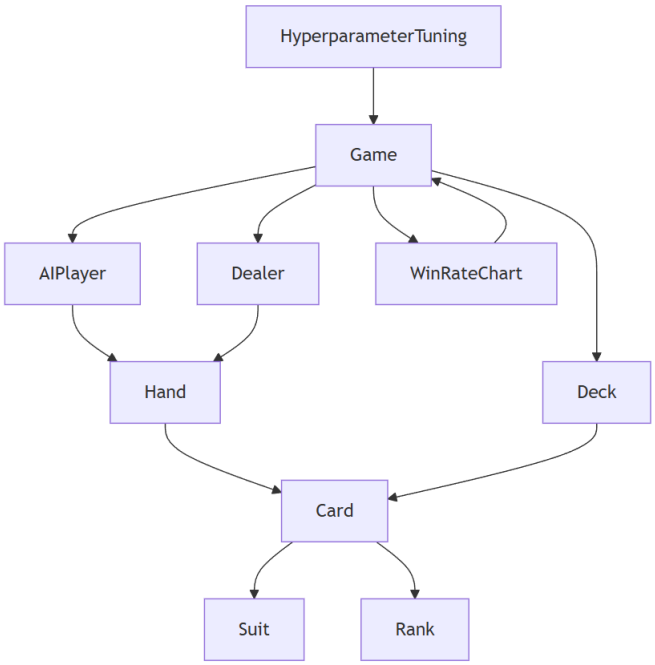

Figura 5.1: Diagrama Generală a Claselor

În imaginea 5.1 este expusă diagrama generală a claselor și cum sunt conectate acestea. De asemenea, clasele Suit si Rank sunt clase de tip enum și reprezintă tipul si culoarea cărților.

5.1.2. Clasa `Card`

Clasa `Card` reprezintă o carte individuală din pachet. Fiecare carte are două atribute: o valoare (2-10, K, Q, K, A) și un tip(Hearts, Diamonds, Clubs, Spades). Această clasă este fundamentală pentru un joc de cărți deoarece regulile și acțiunile jocului se bazează pe clasificarea cărților.

Această clasă este folosită pentru a distribui fiecare carte jucătorului și dealerului. Ea permite identificarea valorii și tipul unei cărți, facilitând astfel scorul, verificarea regulilor și afișarea stării jocului.

5.1.3. Clasa `Deck`

Clasa `Deck` reprezintă pachetul de cărți folosit în joc. Un pachet conține toate cele 52 de cărți, fiecare având o valoare și un tip. Clasa `Deck` este responsabilă de gestionarea cărților: inițializare, amestecare și distribuire.

Clasa `Deck` are rolul de a simula comportamentul unui pachet real de cărți, conținând și permițând:

- Crearea unui pachet complet cu toate cărțile posibile.
- Amestecarea cărților
- Distribuirea cărților către individ și dealer.
- Resetarea pachetului atunci când runda s-a încheiat.

5.1.4. Clasa `Hand`

1. Adăugarea cărților în mână
   Clasa `Hand` permite adăugarea de noi cărți extrase din pachet în mâna jucătorului sau a dealer-ului. Clasa reține intern o listă cu toate cărțile deținute, iar fiecare nouă carte este adăugată la această listă.
2. Calcularea valorii totale a mâinii
   Clasa conține o metodă care calculeaza valoarea totală a cărților din mână, conform regulilor din Blackjack. Fiecare carte are o valoare numerică, de la 2 la 10, cărțile speciale au valoarea 10 iar asul are valoarea 1 sau 11 în funcție de avantajul jucătorului, dacă suma cărților depășește sau nu 21 de puncte. La calcul, se adună valorile tuturor cărților, iar dacă există unul sau mai mulți ași, se verifică dacă este posibil ca unul dintre ei să fie considerat 11 fără a depăși limita de 21. Dacă da, valoarea totală crește cu 10 pentru fiecare as care poate fi considerat 11.
3. Obținerea cărților în mână
   Această metodă oferă posibilitatea de a accesa lista de cărți din mână, pentru a afișa sau a analiza conținutul acesteia în momentul jocului.
4. Resetarea mâinii
   La începutul fiecărei runde, mâna este golită de cărțile vechi pregătind jucatorii pentru o rundă noua.
5. Verificarea stării
   Clasa permite verificarea numarului de cărți din mână(pentru a verifica, de exemplu, dacă jocătorul are blackjack), dacă mâna a depășit valoarea de 21 și dacă mâna conține un as.
   Clasa oferă într-un final o metodă de a transforma mâna intr-un text mai ușor de citit, pentru a fi afișată în consolă sau în rapoarte.

5.1.5. Clasa `Dealer`

Clasa `Dealer` simulează comportamentul și starea dealerului din joc. Dealerul este un participant special, cu reguli proprii diferite față de jucător, astfel:

1. Gestionarea mâinii dealerului
   Dealerul deține o mână proprie, gestionată cu ajutorul unei instanțe din clasa Hand. Toate cărțile primite de dealer sunt agăugate în această mână, iar calculul valorii sau de resetare sunt gestionate în clasa Hand.
2. Primirea de cărți
   Dealerul poate primi cărți noi, două cărți inițiale la începutul rundei și apoi pe parcurs în funcție de joc. Fiecare carte primită este adăugată în mâna dealer-ului.
3. Regulile dealer-ului
   Dealer-ul are o strategie fixă, folosită în majoritatea casino-urilor. Dealer-ul extrage câte o carte până atinge valoarea minimă sau egală cu 17 puncte, după ce a atins această valoare, acțiunea este automat de stand.
4. Afișarea mâinii dealer-ului
   Clasa conține două metode de afișare a mâinii dealerului, o metodă afișează doar prima carte iar a doua este ascunsă, iar o metodă afișează amândouă cărți, metodele fiind folosite în funcție de momentul jocului.
5. Resetarea mâinii
   La începutul fiecărei runde, dealer-ul își resetează mâna de cărțile din runda trecută, pregătind astfel o rundă noua.
6. Accesarea valorii mâinii
   Această metodă furnizează valoarea totală a mâinii dealer-ului, necesară pentru determinarea rezultatului rundei.
7. Metodă de a prelua cartea vizibilă
   Această metodă este folosită ca dată de antrenare a individului, fiind esențială pentru învățarea eficientă.

Clasa `Dealer` gestionează toate aspectele legate de dealer: primirea și gestionarea cărților, aplicarea regulilor specifice de joc, afișarea mâinii și interacțiunea cu celelalte componente ale aplicației. Dealerul acționează ca un arbitru automat, asigurând desfășurarea corectă a jocului conform regulilor standard de Blackjack.

5.1.6. Clasa `AIPlayer`

Clasa AIPlayer simulează un jucător automatizat, controlat prin algoritmi de inteligența artificială. Individul își creează strategii de joc pe care și le optimizează singur.

1. Gestionarea mâinii
   AI-ul deține o mână proprie, gestionată cu ajutorul clasei `Hand`. Cărțile primite de individ sunt adăugate în această mână, iar deciziile de calcul al valorii sau de resetare a mâinii sunt gestionate de catre clasa `Hand`.

2. Q-Table
   Q-Table, sau tabela de valori Q, este structura esențială pentru algoritmii de învățare automată și este folosită pentru a stoca și actualiza experiența acumulată de individ în timpul jocului.
   Această tabelă asociază fiecărei stări a jocului o valoare numerică numită valoare Q. Această valoare reprezintă cât de avantajoasă este acea acțiune în acea stare, pe

baza experienței anterioare a AI-ului. Ea funcționează astfel:

- Fiecare stare a jocului este codificată sub forma unui String în metoda getState.
- Pentru fiecare stare, Q-Table păstrează două valori, una pentru hit, iar una pentru stand.
- La fiecare decizie, AI-ul consultă Q-Table pentru a vedea care acțiune are valoarea Q mai mare.
- După fiecare rundă, valorile Q sunt actualizate în funcție de rezultatul obținut, astfel AI-ul învață din experiență.

Acest Q-Table permite AI-ului să învețe și să-și amintească deciziile din trecut pentru fiecare situație posibilă. Astfel, cu cât individul joacă mai mult, cu atât învață mai eficient.

3. Metoda `getState`
   Metoda `getState` are rolul de a descrie starea curentă a jocului din perspectiva AI-ului. Metoda este folosită pentru a actualiza strategia de joc din Q-Table. Starea este construită pe baza unor informații:
   - Valoarea totală a mâinii AI-ului.
   - Valoarea cărții vizibile a dealer-ului.
   - Daca AI-ul are As.
   - Numărul de cărți din mână (pentru a distinge mâna inițială de valoarea totală).
   - Dacă AI-ul are două cărți pereche (de aceeași valoare).
   - Dacă dealer-ul are cartea vizibila de tip As.

   Aceste informații sunt exportate ulterior într-un șir de caractere de tip String, care este folosită drept cheie pentru Q-Table. Astfel, pentru fiecare stare posibilă, individul poate învăța ce acțiune este mai avantajoasă. Definirea corectă a stării este esențială pentru ca AI-ul să poată distinge între situații diferite și să învețe strategii optime pentru fiecare context de joc.

4. Metoda `decideToHit`
   Această metodă este responsabilă pentru luarea deciziei dacă AI-ul extrage o nouă carte sau ia decizia de stand. Pentru starea curentă obținută prin metoda `getState`, AI-ul verifică Q-Table, unde sunt stocate valorile asociate fiecărei acțiuni(hit/stand). Astfel, decizia se ia folosind strategia de tip epsilon-greedy. Această constantă reprezintă explorarea AI-ului, cu cât valoarea acestei constante este mai mare cu atât AI-ul explorează mai mult, dar și ignoră mai mult Q-Table. Astfel, cu o probabilitatea constantei epsilon AI-ul alege la întâmplare pentru a descoperi strategii noi, în restul cazurilor, AI-ul alege acțiunea cu cea mai mare valoare Q, adică care acțiune i-a adus cele mai bune rezultate în trecut.

5. Metoda `updateQTable`
   Această metodă actualizează valorile Q din Q-Table pe baza experienței recente a AI-ului, astfel:
   - După fiecare acțiune și după fiecare rundă, AI-ul primește o recompensă în funcție de rezultat.
   - Această metodă folosește recompensa pentru a ajusta Q-Value.
   - Actualizarea se face ținând cont de valoarea Q veche pentru acea stare și acțiune, cea mai bună valoare Q pentru starea următoare, rata de învățare și factorul de discount.

$$updatedQValue = oldQValue + learningRate\Big[reward + discountFactor \cdot nextMaxQ - oldQValue\Big] \tag{5.1}$$

Astfel, formula inițială 3.17 va lua forma formulei 5.1 care oferă o lizibilitate mai bună și o înțelegere mai ușoară.

6. Recompense intermediare
   Pe lângă rezultatul final al rundei, AI-ul primește recompense intermediare pentru acțiuni anume, de exemplu, pentru obținerea unui blackjack, recompensă negativă pentru bust și un bonus pentru o mână puternică.
7. Resetarea mâinii
8. Afișarea mâinii

5.1.7. Clasa `Game` - clasa principală

Această clasă este clasa principală a aplicației, ea este responsabilă de gestionarea întregului flux al unei runde de Blackjack. Astfel, clasa conține următoarele funcționalități:

1. Constructorii `Game` - Inițializează toate componentele: pachetul de cărți (Deck), AIPlayer, Dealer, structurile pentru statistici și, opțional, graficul pentru winrate. Permite configurarea parametrilor AI-ului și a numărului de pachete.

2. Metoda `playMultipleGames`
   Metoda principală pentru rularea simulărilor. Parcurge un număr specificat de runde, apelând pentru fiecare rundă metoda playRound(). La intervale regulate, calculează și afișează statisticile de performanță.

3. Metoda `playRound` Gestionează o rundă de Blackjack în urmatoarea ordine:
   - Resetează mâinile AI-ului și dealerului, precum și pachetul de cărți.
   - Apelează dealInitialCards() pentru a distribui cărțile inițiale.
   - Apelează aiTurn() pentru ca AI-ul să ia decizii și să joace.
   - Dacă AI-ul face bust sau are blackjack, se trece direct la determinarea rezultatului.
   - Altfel, apelează dealerTurn() pentru ca dealerul să joace.
   - La final, apelează determineOutcome() pentru a stabili rezultatul rundei și a actualiza Q-table-ul AI-ului.

4. metoda `dealInitialCards`
   Distribuie două cărți AI-ului și două dealerului și afișează mâinile inițiale, cu cartea ascunsă a dealerului.

5. Metoda `aiTurn`
   - Permite AI-ului să joace, luând decizii succesive (hit/stand) pe baza strategiei sale.
   - Pentru fiecare acțiune: Stochează starea și acțiunea. Dacă AI-ul alege „hit”, extrage o carte, actualizează scorul și Q-table-ul cu o recompensă intermediară.

Dacă AI-ul face bust sau alege „stand”, turul se oprește.
- Toate acțiunile și valorile cărților sunt înregistrate pentru analiză.

6. Metoda `dealerTurn` Dealer-ul joacă automat după regula proprie, extrage cărți până ajunge la valoarea minimă sau egală cu 17 puncte. Metoda afișează progresul dealerului și starea mâinii sale.

7. Metoda `determineOutcome`

   - Compară valorile finale ale mâinilor AI-ului și dealerului.
   - Stabilește rezultatul rundei: victorie AI, victorie dealer, egalitate, blackjack sau bust.
   - Acordă recompensele finale AI-ului și actualizează Q-table.
   - Actualizează statistici (număr de victorii, egaluri, etc.)
   - Apelează metoda updateRecentResults() și recordGame() pentru a salva rezultatul si datele rundei.

8. Metoda `updateRecentResults`
   Această metodă actualizează vectorul circular cu rezultatele recente, pentru a stoca ultimele N runde(în cazul nostru 100.000) în care AI-ul a fost cel mai antrenat.

9. Metoda `recordGame`
   Creează o înregistrare detaliată a rundei curente (GameRecord) și salvează aceste înregistrări în istoricul global, incluzând:

   - Valorile inițiale ale mâinilor.
   - Acțiunile AI-ului și cărțile extrase.
   - Numărul de pachete, numărul de hit-uri, prezența asului, etc.
   - Valorile finale și rezultatul rundei.

10. Metoda `exportToCSV`
    Această metodă exporta istoricul complet al ultimelor N runde cu datele din metoda recordGame() într-un fișier cu formatul .CSV, folosit ulterior pentru antrenarea algoritmului de Machine Learning.

11. Metoda `getRecentRates` Returnează statisticile recente (winrate AI, dealer și egalitate) sub formă de vector.

12. Metoda `isBlackjack` Verifică dacă jucătorul are Blackjack din primele două cărți.

13. Metoda `main`
    Această metodă este metoda principală care rulează tot sistemul. În această metodă poți configura parametrii (epsilon, learningRate, discountFactor) și poti adăuga câte pachete să fie utilizate(numberOfDecks). De asemenea, este apelată o metoda interna playMultipleGames care va rula un număr configurabil de runde de joc.
    Astfel, diagrama de flux care explică cum combină clasa toate aceste metode arată astfel:

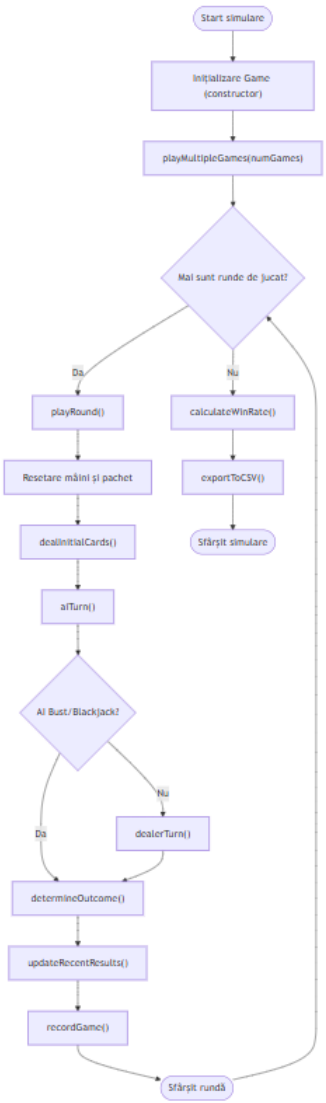

Figura 5.2: Diagramă de flux a clasei Game

5.1.8. Clasa `WinRateChart`

Clasa `WinRateChart` are rolul de a vizualiza grafic evoluția performanței AI-ului și a dealerului pe parcursul simulărilor de Blackjack. Ea oferă o reprezentare vizuală a ratelor de câștig, de pierdere și de egalitate, ajutând la analiza rapidă a progresului și eficienței AI-ului.

Această clasă conține următoarele componente principale:

- Serii de date (XYSeries): rata de câștig a AI-ului, a dealer-ului și rata de egalitate
- Dataset (XYSeriesCollection): Toate seriile de date sunt grupate într-un dataset, care va fi folosit pentru a genera graficul
- Graficul (JFreeChart): Graficul propriu-zis este generat folosind biblioteca JFreeChart, sub forma unui grafic de tip linie (XYLineChart).
- Fereastra grafică (JFrame): Graficul este afișat într-o fereastră separată, care poate fi redimensionată și poziționată pe ecran.

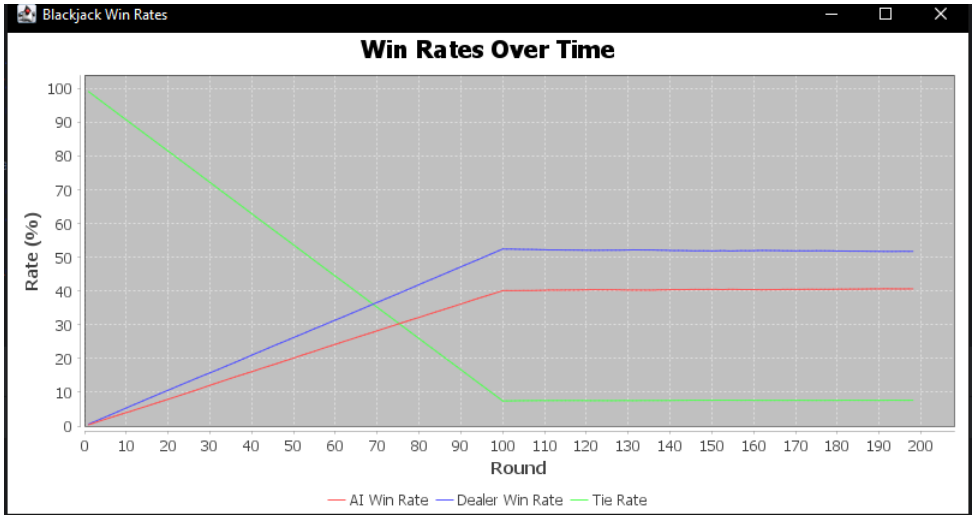

Figura 5.3: Grafic de afișare a ratei de câștig în timp real

Figura 5.3 reprezintă graficul generat în urma rulării programului, care reprezintă rata de câștig afișată în timp real pe parcursul învățării individului.

5.1.9. Clasa `HyperparametersTuning`

Clasa `HyperparameterTuning` are rolul de a automatiza procesul de testare și optimizare a parametrilor de învățare ai AI-ului (cum ar fi rata de explorare, rata de învățare și factorul de discount). Scopul este de a identifica combinația de hiperparametri care duce la cea mai bună performanță a AI-ului în jocul de Blackjack.

Ea conține următoarele componente principale:

- Clasa definește liste de valori posibile pentru fiecare hiperparametru: epsilon, learningRate și discountFactor.
- Număr de simulări per combinație: Pentru fiecare combinație de hiperparametri, se rulează un număr mare de simulări pentru a evalua performanța AI-ului cu acea configurație.
- Structură pentru rezultate: Rezultatele fiecărei combinații (de exemplu, rata de câștig obținută) sunt stocate într-o structură specială, pentru a putea fi comparate ulterior.

5.2. Proiectarea în detaliu al algoritmului Random Forest

5.2.1. Compararea rezultatelor cu studiul matematic

Fișierul `sample.ipynb` servește ca un instrument de analiză și comparare statistică pentru simulările de Blackjack din capitolul trecut. Acest fișier este structurat pe mai multe celule, fiecare având ca scop compararea rezultatelor.

Astfel, prima celulă din cod se ocupă de încărcarea datelor din fișierele .CSV generate anterior. Se folosește biblioteca pandas pentru a citi datele și a le uni într-un singur DataFrame. Această celulă asigură datele necesare pentru analiza statistică din următoarele celule.

A doua celulă definește o funcție care calculează probabilitatea de a obține o mână puternică pentru un pachet și două pachete. Rezultatele sunt comparate cu valorile teoretice din studiul matematic, iar rezultatele sunt afișate într-un tabel comparativ.

A treia celulă reprezintă rezultatele dealerului în funcție de cartea vizibilă. Se construiesc două tabele, primul în care sunt introduse rezultatele din studiu, iar al doilea care este generat din datele preluate din rezultatul AI-ului, tabelele fiind comparate ulterior.

A 4-a celulă calculează profitul așteptat de jucător pentru acțiunea Stand. Celula calculează profitul așteptat de jucător, în funcție de valoarea mâinii sale și de cartea vizibilă a dealerului. De asemenea, se afișează două tabele pentru compararea rezultatelor.

A 5-a celulă calculează profitul așteptat de jucător pentru acțiunea de Hit. Această celulă este antonimă celulei anterioare, ea făcând același lucru pentru acțiunea de Hit, comparând rezultatele.

A 6-a celulă definește analiza strategiilor optime(hard vs soft hands). Se calculează pe baza valorilor de profit așteptat, cea mai bună decizie pentru fiecare valoare a mâinii, împărțit în două tabele, Soft Hands și Hard Hands.

Într-un final, acest fișier este esențial pentru validarea datelor din fișierele CSV. Oferă o vizibilitate și o legibilitate optimă pentru vizualizarea și compararea datelor, oferind o relevanță AI-ului cu Reinforcement Learning.

5.2.2. Antrenarea modelului Random Forest

Fișierul `trainModel.py` este responsabil pentru antrenarea modelului Random Forest, care să prezică valoarea așteptată (EV) pentru unele scenarii din jocul Blackjack, pe baza datelor simulate în fișierele CSV. Acest fișier rulează întregul sistem: încărcarea datelor, antrenarea modelului, evaluarea performanței și salvarea modelului și a rapoartelor vizuale.

Astfel, fișierul începe prin definirea și crearea directoarelor pentru model și rapoarte. Funcția de preparare a datelor încarcă toate fișierele CSV anterioare, le concatenează și le preprocesează pentru a le putea folosi în antrenarea modelului. Datele sunt procesate astfel:

- Se crează o coloană suplimentară care codifică acțiunea de Hit/Stand.
- Redenumește coloanele.
- Transformă valorile duale în valori numerice unde este necesar.
- Selectează doar coloanele relevante pentru antrenare.
- Elimină sau completează valorile lipsă.

Următoarele două funcții(`plot_feature_importance` și `plot_actual_vs_predicted`) generează două imagini. Prima imagine reprezintă cele mai relevante coloane(GiniIndex) folosite de algoritm, iar a doua imagine reprezintă un grafic care compară valorile reale din datele de test și valorile prezise de model.

Funcția de antrenare și salvare a modelului este funcția principală care organizează întregul proces. Ea apelează funcția de pregătire a datelor, împarte datele în seturi de antrenament și test(80-20%), antrenează modelul Random Forest, oferă date de performanță modelului de antrenament și de testare (OOB score și R^2 score), generează graficele și salvează modelul antrenat pentru o utilizare ulterioară.

Ultima funcție, fiind funcția care apelează întregul proces și care permite rularea scriptului. Astfel, acest fișier se ocupă de partea de Machine Learning a proiectului.

5.2.3. Obținerea predicților

În fișierul `blackjack_predictor.py` este centralizată logica de predicție pentru Blackjack, folosindu-se de modelul antrenat anterior.

Prima funcție reprezintă încărcarea modelului antrenat cu Random Forest salvat anterior. Verifică dacă modelul există, afișează un mesaj dacă acest fișier nu este găsit. Dacă fișierul există, încarcă modelul folosind joblib și confirmă printr-un mesaj.

A doua funcție(`parse_cards`) procesează un șir de caractere care reprezintă cărțile (King, As, etc.) și returnează valoarea mâinii și dacă există un as. Acesta transformă șirul în majuscule și separă cărțile individuale, calculează suma cărților; dacă în prezența unui As suma depășește 21 de puncte, As-ul se transformă din 11 puncte într-un punct și, în final, returnează totalul și un boolean dacă mâna conține o carte de tip As.

A treia funcție(`get_model_predictions`) folosește modelul de Machine Learning pentru a prezice valoarea așteptată (EV) pentru acțiunile de Hit sau Stand, în funcție de starea curentă a jocului. Acest fișier creează două seturi de date de intrare: 0 pentru Stand, și 1 pentru Hit, folosește modelul pentru a prezice EV pentru fiecare acțiune și returnează cele două valori(EV pentru Hit și EV pentru Stand).

Funcția principală oferă o interfață în consolă pentru utilizator, permitând introducerea manuală a cărților și oferind o predicție pe baza modelului. Acest lucru este simplificat ulterior printr-o interfată grafică.

5.2.4. Interfața grafică

Acest fișier este responsabil pentru crearea unei interfețe grafice (GUI) care să ofere o interacțiune între sistem și utilizator. Ea folosește biblioteca '`tkinter`' pentru elementele vizuale și '`matloblib`' pentru elementele grafice. Acest script oferă posibilitatea utilizatorului de a introduce cărți, numărul de pachete, scriptul oferind ulterior o predicție a celei mai bune decizii.

În acest sens, întreaga aplicație este încapsulată în clasa '`BlackjackPredictorApp`', care gestionează starea și comportamentul interfeței.

Constructorul care inițializează fereastra principală și componentele aplicației face următoarele lucruri:

- Setează titlul si dimensiunea ferestrei.
- Încarcă modelul de ML din `blackjack_predictor.py`.
- Inițializează variabile de stare (cărtile jucătorului și cartea dealerului) și un grafic.
- Definește constante
- Apelează funcții pentru a construi interfața

Funcția de creare a elementelor vizuale(`crate_widgets`) oferă Frame-uri pentru dealer și pachete care organizează secțiunile pentru cartea dealerului și selectorul de pachete, Frame pentru jucător care afișează cărțile și valoarea lor, Frame pentru acțiuni care conțin butoane pentru adăugare carte, calculează și resetează jocul, Frame pentru rezultate care afișează predicția și recomandarea și, în final, un Frame pentru grafic care arată evoluția predicțiilor pe măsură ce se adaugă cărți.

Funcția de selectare a cărții dintr-un pachet(`open_card_selector`) deschide o fereastră nouă care permite utilizatorului să selecteze o carte dintr-un pachet de 52 de cărți. Ea este folosită atât pentru jucător, cât și pentru selectarea cărții vizibile a dealer-ului.

Funcția de selectare a cărții (`select_card`) este funcția apelată în momentul în care jucătorul apasă pe o carte din pachet. Funcția adaugă cartea la starea internă (dealer sau jucător), reîmprospătează fereastra pentru a apărea în fereastra principală și închide

automat fereastra după selectarea unei cărți.

Funcția de reîmprospătare a ferestrei (`update_display`) actualizează interfața pentru a reflecta starea curentă a jocului. Această funcție afișează imaginea cărților jucătorului și cartea dealerului și recalculează și afișează totalul mâinii jucătorului, incluzând valoarea variabilă a asului.

Funcția de calculare a predicției este logica principală care se execută la apăsarea butonului 'Calculează'. Această funcție verifică dacă au fost selectate cărțile necesare, colectează datele, apelează funcția de predicție din fișierul `blackjack_predictor.py` pentru a obține EV, adaugă rezultatele în grafic și afișează valorile EV și recomandarea finală.

Funcția `update_plot` redesenează graficul cu datele actualizate din istoric. Șterge graficul anterior, desenează valorile așteptate pentru acțiuni și adaugă un titlu și o legendă pentru claritate.

Funcția de resetare (`reset_all`) resetează jocul la starea inițială în momentul apăsării butonului 'Resetează Jocul'. Golește cărțile, resetează graficul și revine la starea de la început.

Într-un final, funcția principală '`main`' folosește instanțele necesare clasei BlackJackPredictorApp și pornește aplicația.

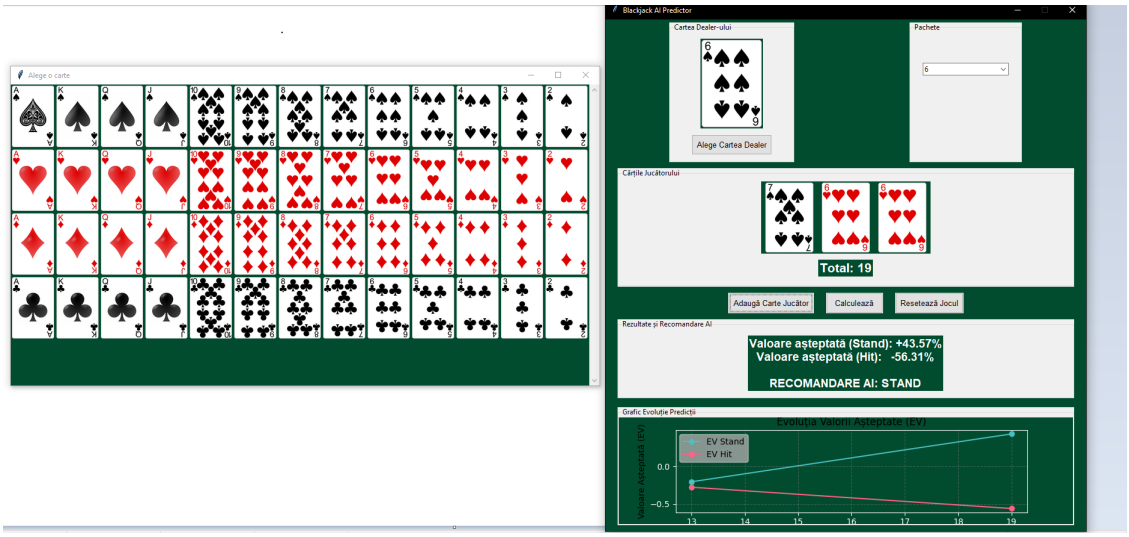

Figura 5.4: Interfața grafică a aplicației

Figura 5.4 reprezintă interfața grafică a aplicației și componentele sale. De asemenea, este afișată și fereastra de selectare a cărților, șansele de câștig pentru fiecare decizie și graficul predicției.

Capitolul 6. Testare și validare

6.1. Testarea și validarea aplicației cu Reinforcement Learning

Testarea aplicației este fundamentală pentru ca aplicația să funcționeze corect, că regulile jocului sunt respectate și ca algoritmul de inteligență artificială reușește să-și îmbunătățească performanța.

Astfel, am verificat corectitudinea implementării regulilor de Blackjack, incluzand distribuirea cărților, detectarea situațiilor speciale (blackjack, egalitate sau bust), respectarea regulilor dealerului, distribuirea corectă a cărților, calculul mâinilor și exportul corect al datelor în fișierele CSV.

Am rulat un număr diferit de runde pentru a observa comportamentul AI-ului și dacă își îmbunătățește rata de câștig. Am analizat evoluția Q-Table-ului, comportamentul AI-ului în unele situații (mână puternică, dealer-ul are As, etc.) și capacitatea AI-ului de a evita bust-ul.

Parametrii AI-ului (epsilon, learningRate și discountFactor) au fost testați automat prin clasa HyperparameterTuning. Au fost rulate 20.000 de runde pentru fiecare combinație de parametrii, astfel valorile cu rata de câștig cea mai mare conform 6.1 au fost epsilon = 0.1, learningRate = 0.1 și discountFactor = 0.1.

Epsilon	LearningRate	DiscountFactor	WinRate	DealerWinRate	TieRate	TotalGames
0.1	0.1	0.1	0.4077	0.5194	0.0737	200000
0.1	0.1	0.2	0.4075	0.5179	0.0746	200000
0.1	0.1	0.3	0.4074	0.5158	0.0765	200000
0.1	0.1	0.4	0.407	0.5198	0.0748	200000
0.1	0.1	0.5	0.4069	0.5173	0.0757	200000
0.1	0.1	0.6	0.4062	0.5223	0.0736	200000
0.1	0.1	0.7	0.4055	0.5198	0.0746	200000
0.1	0.1	0.8	0.4054	0.5176	0.075	200000
0.1	0.1	0.9	0.4042	0.518	0.0758	200000
0.1	0.2	0.1	0.3986	0.5309	0.0729	200000
0.1	0.2	0.2	0.3983	0.5288	0.0727	200000
0.1	0.2	0.3	0.3983	0.5276	0.0741	200000
0.1	0.2	0.4	0.3977	0.531	0.0722	200000
0.1	0.2	0.5	0.3976	0.5299	0.0724	200000

Figura 6.1: Rezultatele parametrilor în ordine descrescătoare dupa W

Testarea rezultatelor AI-ului, în urma deciziei alegerii parametrilor, a fost făcută pe un număr de 1.000.000 de runde cu 6 pachete, având rezultatele:

- Pentru 1.000.000 runde: 40.54% winrate, 7.58% egalitate și 51.88% loserate.
- Pentru ultimele 100.000 de runde: 40.85% winrate, 7.62% egalitate și 51.53% loserate.

Acest rezultat oferă o validare algoritmului, având o marjă de eroare de doar 1.5%, dar în schimb șansa de câștig a dealerului a scăzut de la un standard de 52% la 51.53%, ceea ce oferă o șansă mai ridicată de a nu pierde o rundă. De asemenea, analiza Q-table a

arătat că AI-ul a învățat să evite acțiunile riscante, de exemplu, să dea hit la o mână de 20 de puncte.

6.2. Testarea și Validarea AI-ului din Machine Learning

Testarea a constat în verificarea fiecărui modul și a fiecărei funcții principale din proiect.

Funcțiile prezente în fișierele train_model.py și blackjack_predictor.py au fost testate manual pentru a asigura corectitudinea încărcării și procesării datelor. S-au verificat dacă datele din fișierele CSV sunt citite corect, dacă sunt valori lipsă și dacă variabilele de intrare pentru model sunt generate în mod așteptat.

S-a verificat dacă algoritmul Random Forest a fost antrenat corect, poate fi salvat separat și încărcat corespunzător, iar funcțiile de predicție returnează valori așteptate pentru scenariile date. Funcțiile precum parse_cards au fost testate manual cu diverse inputuri pentru a asigura buna desfășurare a mâinilor, inclusiv a Asului.

Testarea interfeței grafice a fost testată manual pentru a verifica funcționabilitatea butoanelor și selecților, afișarea corectă a imaginilor cărților, dacă mesajele sunt afișate corect și decizii unice, de exemplu, dacă jucătorul alege doar o carte, sau dacă jucătorul nu a ales cartea vizibilă a dealer-ului. De asemenea, s-a testat funcționabilitatea graficului, dacă arată corect valorile așteptate pentru Hit și Stand.

Validarea modelului de Machine Learning s-a realizat pe baza împărțirii datelor în setul de antrenament și de test, datele fiind împărțite 80% pentru antrenare și 20% pentru testare. Performanța a fost măsurată cu ajutorul metricilor R^2 score și Out-Of-Bag score (OOB). Valorile obținute sunt: OOB Score: 0.2117 și R^2 Score: 0.2101.

De asemenea, au fost generate suplimentar grafice de importanță a coloanelor și un grafic de afișare a valorilor prezise care au confirmat că modelul învață corect, precum sunt afișate în imaginile 6.2 și 6.3.

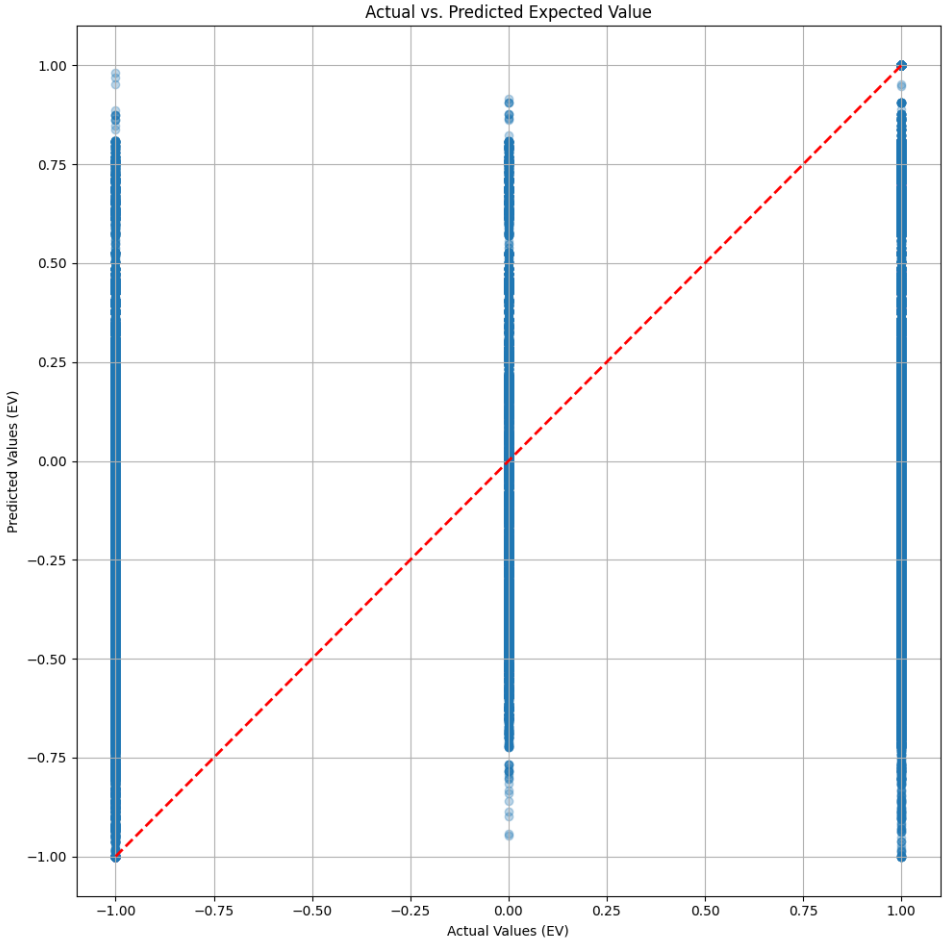

Figura 6.2: Valorile Actuale vs. Valorile Prezise

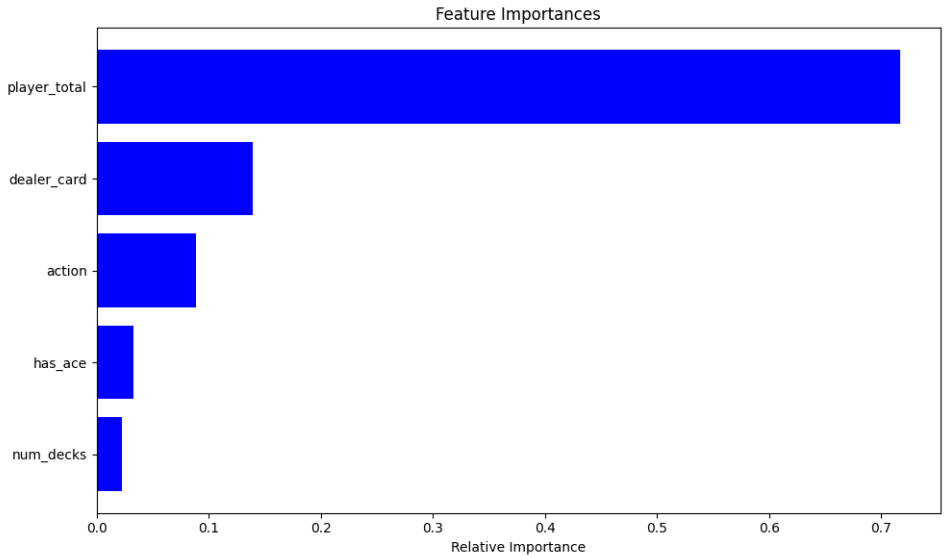

Figura 6.3: Imporanța datelor de intrare

Capitolul 7. Manual de instalare și utilizare

Proiectul, fiind împărțit pe două structuri, a fost organizat în două foldere diferite: "AI_RL_BJ" și "JupyterProject". Cele două structuri funcționează independent. Folderele se pot dezarhiva din arhiva Licenta.rar.

7.1. Manual de instalare și utilizare pentru "AI_RL_BJ"

Pentru rularea programului este necesară instalarea Java Development Kit (JDK) versiunea 17+ (recomandat), un editor de text sau un mediu de dezvoltare (IDE), precum și un program de vizualizare a fișierelor CSV (de exemplu, Microsoft Excel).

7.1.1. Instalare Java

În cazul în care Java nu este deja instalat pe sistem, se recomandă descărcarea și instalarea JDK de la `https://adoptopenjdk.net/` sau `https://www.oracle.com/java/technologies/downloads/`. După instalare, se poate verifica în terminal/cmd folosind comanda:

```
java -version
```

Ar trebui să apară versiunea instalată.

7.1.2. Compilarea proiectului

Se deschide terminalul sau Command Prompt și se navighează în folderul proiectului:

```
cd C:\BlackjackAI\proiect_licenta1\src
```

Apoi se compilează toate fișierele Java cu comanda:

```
javac *.java
```

(sau se poate utiliza funcția de build din IDE-ul preferat)

7.1.3. Rularea aplicației

În același terminal, aplicația se rulează cu comanda:

```
java Game
```

sau, dacă punctul de intrare se află în altă clasă (de exemplu, HyperparameterTuning):

```
java HyperparameterTuning
```

7.1.4. Utilizarea aplicației

- La rularea aplicației, simularea va începe automat, iar AI-ul va juca un număr mare de runde împotriva dealerului. Pe ecran vor apărea mesaje cu progresul simulării și, opțional, o fereastră cu graficul evoluției ratelor de câștig.
- După finalizarea simulării, în folderul proiectului vor fi generate fișiere CSV (ex: `blackjack_stats.csv`) care conțin date detaliate despre fiecare rundă și despre performanța AI-ului. Aceste fișiere pot fi deschise cu Excel sau orice alt program de calcul tabelar pentru analizarea rezultatelor.
- Pentru testarea diferitelor strategii sau optimizarea AI-ului, se poate rula clasa `HyperparameterTuning`, care va încerca automat mai multe combinații de parametri și va salva rezultatele în fișierul `hyperparameter_results.csv`.

7.1.5. Recomandări de utilizare

- Pentru obținerea unor rezultate relevante, se recomandă rularea simulării cu cel puțin 100.000 de runde.
- Parametrii AI-ului pot fi modificați direct în cod sau la inițializarea clasei `Game` / `HyperparameterTuning`.
- Simularea poate fi oprită manual înainte de final folosind combinația de taste `Ctrl+C` în terminal.

7.2. Manual de instalare și utilizare pentru folderul "JupyterProject"

Acest folder conține aplicația care utilizează algoritmul Random Forest din Machine Learning, precum și interfața grafică. Cerințele software includ instalarea Python versiunea 3.12 sau mai recentă și pachetul `pip` pentru gestionarea dependențelor.

7.2.1. Instalarea Python

1. Se accesează site-ul `https://www.python.org/downloads/` și se descarcă cea mai recentă versiune de Python 3.x compatibilă cu sistemul de operare.
2. În timpul instalării, trebuie bifată opțiunea *Add Python to PATH*.

7.2.2. Instalarea dependențelor

1. Se deschide o fereastră de Command Prompt (Windows) sau Terminal (Linux/Mac).
2. Se navighează în directorul proiectului:

   ```
   cd C:\Proiecte\BlackjackAI
   ```

3. Se instalează toate pachetele necesare folosind comanda:

   ```
   pip install -r requirements.txt
   ```

7.2.3. Verificarea instalării

1. Se verifică existența fișierului `blackjack_model.joblib` în directorul `models`.
2. Dacă fișierul nu există, se rulează scriptul de antrenare a modelului:

   ```
   python train_model.py
   ```

3. La final, ar trebui să apară mesajul *"Model saved to models/blackjack_model.joblib"*.

7.2.4. Rularea aplicației

1. În aceeași fereastră de Command Prompt/Terminal, se rulează:

   ```
   python blackjack_gui/gui_app.py
   ```

2. Se va deschide fereastra principală a aplicației **Blackjack AI Predictor**.

7.2.5. Utilizarea aplicației

1. În partea de sus a ferestrei, se apasă butonul **„Alege Cartea Dealer"**.
2. Se deschide o fereastră cu toate cărțile posibile. Se selectează cartea dorită pentru dealer (ex: „A" pentru As, „10" pentru zece etc.).
3. În secțiunea „Cărțile Jucătorului", se apasă pe **„Adaugă Carte Jucător"** pentru a selecta fiecare carte din mâna jucătorului. Pot fi adăugate până la 6 cărți.
4. În partea dreaptă sus, se selectează din lista derulantă numărul de pachete de cărți folosite în joc (între 1 și 6).

7.2.6. Calcularea predicției

1. După selectarea cărților și a numărului de pachete, se apasă butonul **„Calculează"**.
2. În secțiunea „Rezultate și Recomandare AI" vor fi afișate:
   - Valoarea așteptată (EV) pentru acțiunea „Stand" (dacă se stă)
   - Valoarea așteptată (EV) pentru acțiunea „Hit" (dacă se mai trage o carte)
   - Recomandarea AI (care acțiune este optimă în acel scenariu)
3. În partea de jos a ferestrei se va afișa un grafic care arată evoluția valorii așteptate pentru acțiunile „Stand" și „Hit", pe măsură ce se adaugă sau se modifică cărțile.
4. Pentru a începe o nouă simulare, se apasă butonul **„Resetează Jocul"**. Toate selecțiile vor fi șterse, iar utilizatorul poate introduce un nou scenariu.

7.2.7. Recomandări de utilizare

- Pentru obținerea unor rezultate relevante, este importantă selectarea corectă a cărților și a numărului de pachete.
- Dacă apar erori la încărcarea modelului, se recomandă rularea scriptului de antrenare și verificarea existenței fișierului `blackjack_model.joblib` în directorul `models`.

Capitolul 8. Concluzii

În concluzie, proiectul a reușit să îndeplinească ceea ce și-a propus. Am creat o aplicație care folosește algoritmi de învățare automată pentru a simula un joc de blackjack, și l-am apropiat cât mai mult de perfect, având o marjă de eroare de doar 1.5%. Am reușit să exportăm datele într-un Dataset pentru a avea o bază informatică pentru crearea unei predicții pe baza algoritmilor din Machine Learning și am creat o statistică afișată în timp real a ratei de câștig.

Am validat rezultatele AI-ului cu un studiu matematic, oferind date și tabele comparative, care au avut rezultat relativ egal, cu o marjă de eroare infimă, oferind astfel o relevanță algoritmului și datelor exportate.

Am reușit să creăm un model de Machine Learning folosind algoritmul Random Forest, care a reușit să prezică cea mai bună acțiune bazată pe datele exportate. Modelul a reușit un scor nu tocmai reușit, R^2 și OOB score 0.21 care oferă o marjă de eroare de aproape 80%, ceea ce nu este tocmai optim. Acest scor oferă un mare credit factorului de noroc, o predicție exactă este cu atât mai grea cu cât numărul de pachete este mai mare.

De asemenea, am reușit și să implementăm o interfață grafică, care permite utilizatorului să interacționeze cu aplicația, și să introducă cărțile prezente în joc, cât și numărul de pachete, interfața oferind o predicție în formă de procentaj și un grafic explicativ.

Proiectul poate implementa în versiunile viitoare decizia de split, o acțiune nouă, care permite jucătorului să împartă cărțile inițiale în două mâini diferite. De asemenea, o altă acțiune ar fi cea de double down, care permite jucătorului să dubleze suma pariată de pe primele două cărți. Acest lucru ar putea influența factorul de recompensă și ar putea dezvolta strategii noi de joc pentru AI.

Pe lângă rezultatele obținute, proiectul scoate în evidență potențialul inteligenței artificiale de a aborda probleme complexe, care tot timpul au fost în preocuparea oamenilor, precum jocurile de noroc. Procesul de dezvoltare a permis nu doar validarea conceptelor teoretice, ci și provocări practice, de ordinul optimizării, gestionării datelor și interpretarea rezultatelor. Limitările aplicației, cât și factorul de noroc reprezintă obstacole care pot duce la dezvoltarea aplicației.

Experiența acumulată a contribuit la consolidarea cunoștințelor în domeniul programării orientate pe obiecte, al algoritmilor de învățare automată și al analizei statisticilor. Consider că această lucrare reprezintă nu doar o realizare tehnică, ci și un punct de plecare pentru explorări viitoare, atât în domeniul simulărilor inteligente, cât și în cel al dezvoltării de soluții inovatoare bazate pe date și algoritmi adaptivi.

Bibliografie

[1] C. Barboianu, *Probability Guide to Gambling: The Mathematics of Dice, Slots, Roulette, Baccarat, Blackjack, Poker, Lottery and Sport Bets.* INFAROM Publishing, 2006.

[2] "The History of Blackjack," [Published in May 29,2019]. [Online]. Available: https://crescent.edu/post/the-history-of-blackjack

[3] "The History of Blackjack: How This Legendary Game Was Born," [publised on Jun 14, 2023]. [Online]. Available: https://casino.betmgm.com/en/blog/the-history-of-blackjack-how-this-legendary-game-was-born/

[4] "Basic Strategy," [Presented by an Official Casino House]. [Online]. Available: https://bicyclecards.com/how-to-play/blackjack

[5] Justin Berkman, "Perfect Blackjack Strategy: 15 Charts to Help You Master the Game." [Online]. Available: https://blog.prepscholar.com/blackjack-strategy

[6] "House Edge in Blackjack," [By Content Team in 14.06.2023]. [Online]. Available: https://sigma.world/play/blog/what-is-house-edge-in-blackjack/

[7] Jyotpal Singh Flora, "What Is House Edge in Blackjack?" [publised on February 24, 2023]. [Online]. Available: https://sigma.world/play/blog/what-is-house-edge-in-blackjack/

[8] Wikipedia, "Deep blue — wikipedia, enciclopedia liberă," 2025, [Online; accesat la 14-iunie-2025]. [Online]. Available: //ro.wikipedia.org/w/index.php?title=Deep_Blue&oldid=16874902

[9] Demis Hassabis || CEO and Co-Founder, Google DeepMind , "AlphaGo: using machine learning to master the ancient game of Go," [publised on Jan 27, 2016]. [Online]. Available: https://blog.google/technology/ai/alphago-machine-learning-game-go/

[10] Wikipedia contributors, "Monte carlo method — Wikipedia, the free encyclopedia," 2025, [Online; accessed 7-July-2025]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Monte_Carlo_method&oldid=1288027332

[11] ——, "Alphago — Wikipedia, the free encyclopedia," 2025, [Online; accessed 14-June-2025]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=AlphaGo&oldid=1294370965

[12] Jacob Murel Ph.D. and Eda Kavlakoglu, "What is reinforcement learning?" [Online]. Available: https://www.ibm.com/think/topics/reinforcement-learning

[13] Wikipedia contributors, "Reinforcement learning — Wikipedia, the free encyclopedia," 2025, [Online; accessed 14-June-2025]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Reinforcement_learning&oldid=1293560516

[14] "A Beginner's Guide to Q-Learning: Understanding with a Simple Gridworld Example ," [Written by Gregory Kovalchuk]. [Online]. Available: https://medium.com/@goldengrisha/a-beginners-guide-to-q-learning-understanding-with-a-simple-gridworld-example-2b6736e7e2c9

[15] D. Litman, "Reinforcement learning," https://people.cs.pitt.edu/~litman/courses/cs2710/lectures/rl.pdf, 2016, lecture slides, University of Pittsburgh.

[16] Wikipedia contributors, "Machine learning — Wikipedia, the free encyclopedia," 2025, [Online; accessed 14-June-2025]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Machine_learning&oldid=1294715931

[17] Vijay Kanade , "What Is Machine Learning? Definition, Types, Applications, and Trends," [publised on April 4, 2022]. [Online]. Available: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-ml/

[18] S. Kumar, "Supervised vs unsupervised vs reinforcement," https://www.aitude.com/supervised-vs-unsupervised-vs-reinforcement/, 2020.

[19] Sruthi , "Random Forest Algorithm in Machine Learning," [publised on 01 May, 2025]. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

[20] "What Is Random Forest? | IBM," [Published by IBM Co.]. [Online]. Available: https://www.ibm.com/think/topics/random-forest