Ultimate Texas Hold'em

Poročilo projekta pri predmetu

Izbrane teme iz analize podatkov

Neo Mistral
Oskar Težak

Fakulteta za matematiko in fiziko
Univerza v Ljubljani

2. september 2025

1 Uvod

Cilj projekta je razviti model z uporabo **nevronskih mrež**, ki bi znal poiskati optimalno strategijo za igranje igre *Texas Hold'em* pokra.

Najboljši in končni rezultati so vključeni v dveh priloženih Jupyter zvezkih:

- `NN-Optim_Mean.ipynb` – končna konfiguracija in rezultati za **enega igralca** (optimizacija pragov po krogih).
- `NN-Optim_Mean_n.ipynb` – končna konfiguracija in rezultati za n **igralcev** (posplošitev na večigralni scenarij).

Potek in logika **generiranja podatkov** sta podrobno dokumentirana v zvezku:

- `data_generation.ipynb` – opis načina vzorčenja, strukturiranja vhodov za posamezne kroge (*pre-flop*, *flop*, *turn*, *river*) in priprava ciljnih spremenljivk (povprečenje, binarni izid, izkupiček posamezne igre).

Dodatne pomožne skripte, vmesni poskusi ter spremljevalne datoteke so dostopni v javnem repozitoriju: `github.com/oskartezak/Ultimate_Texas_Holdem`

Za potrebe projekta sva pravila igre in logiko popolnoma sama implementirala (ultimate.py), s čimer sva pridobila popoln nadzor nad generiranjem podatkov in načinom prikazovanja igre v simulacijah. V datotetki testiranje.ipynb pa se lahko preiskusi in dobi vpogled v porazdelitev kombinacij in izid iger pri najini začetni stregiji, katero je cilj izboljšati z uporabo nevronskih mrež.

2 Opis igre Ultimate Texas Hold'em

Ultimate Texas Hold'em je različica pokra, ki se igra proti delilcu, ne pa proti drugim igralcem. Cilj igre je premagati delilca z boljšo poker kombinacijo iz dveh lastnih in petih skupnih kart.

Potek igre

1. **Začetna stava** – igralec postavi enaka vložka na polji *Ante* in *Blind*. Po želji lahko postavi tudi stransko stavo *Trips*, ki se izplača na osnovi končne kombinacije.
2. **Deljenje kart** – igralec in delilec prejmeta po dve skriti karti. Igralec lahko zdaj izbere, da stavi dodatno stavo *Play* v višini 3× ali 4× začetne stave, ali pa počaka.
3. **Flop** – razkrijejo se prve tri skupne karte. Če igralec prej ni stavil, lahko zdaj postavi stavo *Play* v višini 2× začetne stave, ali pa še vedno počaka.
4. **Turn in River** – razkrijeta se še četrta in peta skupna karta. Če igralec do zdaj ni stavil, mora zdaj postaviti stavo *Play* v višini 1× začetne stave.
5. **Razkritje** – delilec razkrije svoji dve karti. Za kvalifikacijo mora imeti vsaj par. Če se delilec ne kvalificira, se stava *Ante* vrne igralcu, stave *Blind* in *Play* pa se obravnavajo glede na primerjavo kombinacij.
6. **Izplačilo** –

- Stava *Play* se vedno izplača $1:1$, če je igralčeva kombinacija boljša.
- Stava *Ante* se izplača $1:1$, če se delilec kvalificira in igralec zmaga.
- Stava *Blind* se izplača po posebni lestvici, odvisno od moči igralčeve kombinacije (npr. *Straight* $1:1$, *Flush* $3:2$, *Full House* $3:1$, ipd.).
- Stranska stava *Trips* se izplača posebej glede na moč kombinacije, neodvisno od izida proti delilcu.

3 Generiranje in oblike podatkov

Podatke sva generirala naključno. Igra temelji na 52 kartah, ki sva jih označila s številkami od 1 do 52. Vsak igralec, vključno z delilcem, prejme dve karti, na mizi pa je skupaj pet kart: najprej trije t. i. *flop* in nato še dve karti, ki skupaj s prvimi tremi tvorita *river*.

Podatke sva vedno generirala za celotno igro, da sva lahko izračunala njen izid. V model pa sva nato vključila le tiste karte, ki so bile na voljo v določenem krogu igre. Te karte so predstavljale vhodne spremenljivke:

- v prvem krogu je model videl samo dve karti igralca (ostala mesta so bila nastavljena na 0),
- v drugem krogu pet kart,
- v tretjem krogu vseh sedem kart.

Model nikoli ni imel vpogleda v delilčeve karte. V primeru več igralcev pa je model upošteval tudi njihove karte.

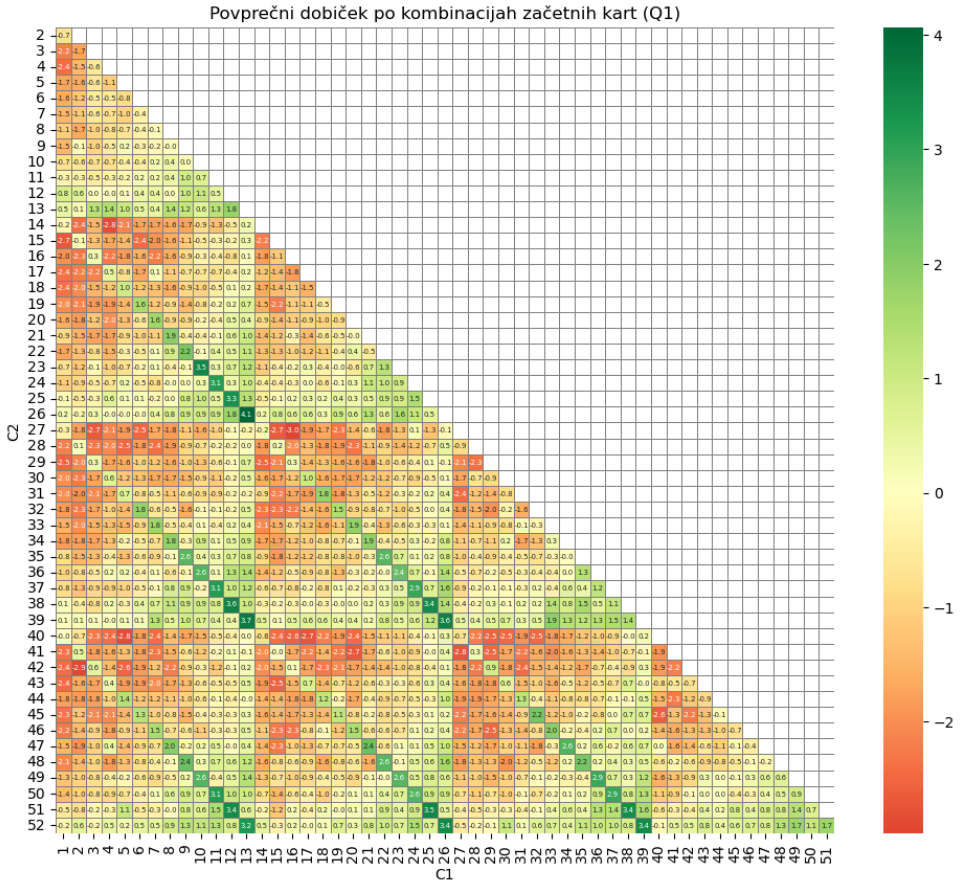
Drugi del podatkov so predstavljale **ciljne vrednosti**. Preizkusila sva več načinov njihovega določanja:

1. **Povprečenje iger** – za vsak par igralčevih kart sva generirala več možnih kombinacij preostalih kart in izračunala povprečen izid na za krog igre.
2. **Binaren izid (zmaga/poraz)** – kot ciljno spremenljivko sva uporabila enostaven pokazatelj zmage ali poraza (0 ali 1). Tudi te vrednosti sva generirala na enak način kot pri povprečenju, le da sva namesto povprečnega izida upoštevala zmago ali poraz.
3. **Izkupiček posamezne igre** – kot ciljno spremenljivko sva uporabila dejanski izid posamezne igre, brez povprečenja. Tudi ti podatki so bili generirani enako kot pri prvem pristopu, pri čemer sva ohranila rezultat posamezne igre.

Velikost podatkovne zbirke. Za igralca je velikost podatkovne zbirke znašala 1326 primerov (vseh možnih začetnih kombinacij igralčevih kart). Za preostale karte sva uporabila večkratnike te številke: prvi večkratnik 15 (za "flop"), drugi 15 (za "river"), in pri zadnjem večkratnik 20 (za delilca).

Očitno za optimalno strategijo v prvem krogu (preflop) ni zares potreben model, saj lahko "profitabilnežačetne roke razberemo direktno iz vzorca, ampak v kasnejših krogih (flop, river) postane prostor možnih kombinacij prevelik, zato je uporaba modela smiselna za posploševanje in napovedovanje vrednosti tudi za redke ali še nevidene situacije. Kritične so tudi mejne sitiacije, kjer je pričakovan dobiček blizu 0, saj se lahko in se zgodi,

da so nekatere karte bolj "dovzetneža nekatere kombinacije in dobijo večji potencial v nadaljevanju. Okvirna strategija za preflop je prikazana na naskledlnji sliki:



Slika 1: Prikaz profitabilnosti začetnih kart v prvem krogu (preflop).

4 Metodologija

Pri večini poskusov sva uporabila **linearne** oziroma **konvolucijske nevronske mreže**. Za določanje stavnih pragov sva poleg različnih optimizacijskih pristopov uporabila tudi **lastne ocene**, s katerimi sva dodatno prilagodila strategijo in izboljšala delovanje modela. Povsod sva poskušala z razlčnimi velikostmi slojev.

4.1 Poskus 1

Vhodni podatki: povprečenje iger
Model: konvolucijski in linearni modeli z *one-hot* kodiranjem kart
Rezultati testiranja: rezultati so bili slabi, modeli niso zaznali uporabnih vzorcev in niso uspeli preseči naključnega ugibanja.
Komentar: preveliko število vhodov, ki nosijo premalo informacij. Učenje traja predolgo.

4.2 Poskus 2

Vhodni podatki: povprečenje iger
Model: konvolucijski in linearni modeli, kjer je bila vsaka karta predstavljena s parom

(rang, barva)
Rezultati testiranja: rezultati so bili nezadovoljivi, saj se prednost delilca ni zmanjšala.
Komentar: takšna predstavitev slabše prikazuje relacije med kartami.

4.3 Poskus 3

Vhodni podatki: povprečenje iger
Model: linearni model z *embedding* plastjo in sigmoidno aktivacijsko funkcijo $(0, 1)$
Rezultati testiranja: prag za stavo je bil nastavljen na 0.5. Delilec je v povprečju ohranil prednost približno 5%.
Komentar: težave pri nastavljanju pragov, model je bil zelo občutljiv na manjše spremembe.

4.4 Poskus 4

Vhodni podatki: povprečenje iger
Model: konvolucijski model z *embedding* plastjo in sigmoidno aktivacijsko funkcijo $(0, 1)$
Rezultati testiranja: prag za stavo 0.5, delilec je bil v profitu za približno 5%.
Komentar: enake težave kot pri linearnem modelu – težavno nastavljanje pragov in občutljivost na spremembe.

4.5 Poskus 5

Vhodni podatki: binarni izid igre
Model: linearni model z *embedding* plastjo in sigmoidno aktivacijsko funkcijo $(0, 1)$
Rezultati testiranja: prag za stavo 0.5. Delilec je ohranil prednost približno 7%.
Komentar: model ocenjuje le zmage/poraze, ne pa dobička, zato so razlike v izkupičku lahko velike.

4.6 Poskus 6

Vhodni podatki: binarni izid igre
Model: konvolucijski model z *embedding* plastjo in sigmoidno aktivacijsko funkcijo $(0, 1)$
Rezultati testiranja: prag za stavo 0.5. Delilec je bil v profitu približno 7%.
Komentar: enake omejitve kot pri linearnem modelu – ocenjevanje le zmage, brez upoštevanja velikosti dobička.

4.7 Poskus 7

Vhodni podatki: izkupiček posamezne igre
Model: konvolucijski model z *embedding* plastjo in sigmoidno aktivacijsko funkcijo $(0, 1)$
Rezultati testiranja: prag za stavo 0.5. Delilec je bil v prednosti približno 4%.
Komentar: za enako kakovost rezultatov je potrebna bistveno večja količina podatkov kot pri povprečenju.

4.8 Poskus 8

Vhodni podatki: izkupiček posamezne igre
Model: linearni model z *embedding* plastjo in sigmoidno aktivacijsko funkcijo $(0, 1)$

Rezultati testiranja: prag za stavo 0.5. Delilec je bil v prednosti približno 5%.
Komentar: enake ugotovitve kot pri poskusu 7 – potrebna je večja količina podatkov.

4.9 Poskus 9

Vhodni podatki: povprečenje iger
Model: linearni model z *embedding* plastjo
Rezultati testiranja: prag za stavo ročno prilagojen okoli 0, fiksen za vse kroge. Delilec je bil v povprečju v prednosti približno 4%.
Komentar: ročno nastavljanje pragov je lažje, model pa se uči razmeroma hitro.

4.10 Poskus 10

Vhodni podatki: povprečenje iger
Model: konvolucijski model z *embedding* plastjo
Rezultati testiranja: prag za stavo ročno prilagojen okoli 0, fiksen za vse kroge. Delilec je bil v prednosti približno 4%.
Komentar: podobno kot pri linearnem modelu – enostavnejše nastavljanje pragov in hitrejše učenje.

4.11 Poskus 11

Vhodni podatki: binarni izid igre
Model: linearni model z *embedding* plastjo
Rezultati testiranja: prag za stavo ročno nastavljen okoli 0.5. Delilec je bil v prednosti približno 5%.
Komentar: ni velike razlike v primerjavi z uporabo sigmoidne funkcije. Model še vedno ne meri dejanskega dobička.

4.12 Poskus 12

Vhodni podatki: binarni izid igre
Model: konvolucijski model z *embedding* plastjo
Rezultati testiranja: prag za stavo ročno nastavljen okoli 0.5. Delilec je bil v prednosti približno 6%.
Komentar: podobno kot pri linearnem modelu – brez merjenja dejanskega dobička, rezultati ostajajo nezadostni.

4.13 Poskus 13

Vhodni podatki: izkupiček posamezne igre
Model: konvolucijski model z *embedding* plastjo
Rezultati testiranja: prag za stavo ročno nastavljen okoli 0, fiksen za vse kroge. Delilec je bil v prednosti približno 4%.
Komentar: zaradi velike količine podatkov je bilo učenje počasno.

4.14 Poskus 14

Vhodni podatki: izkupiček posamezne igre
Model: linearni model z *embedding* plastjo
Rezultati testiranja: prag za stavo ročno nastavljen okoli 0, fiksen za vse kroge. Delilec je bil v prednosti približno 4%.
Komentar: enake težave kot pri poskusu 13 – prevelik obseg podatkov in posledično počasno učenje.

4.15 Poskus 15

Vhodni podatki: povprečenje iger
Model: linearni model z *embedding* plastjo
Rezultati testiranja: pragovi za vsak krog optimizirani z `minimize` funkcijo. Delilec je bil v prednosti približno 3%.
Komentar: hitro učenje, optimizator se je izkazal kot učinkovit.

4.16 Poskus 16

Vhodni podatki: povprečenje iger
Model: konvolucijski model z *embedding* plastjo
Rezultati testiranja: pragovi za vsak krog optimizirani z `minimize` funkcijo. Delilec je bil v prednosti približno 3%.
Komentar: hitro učenje, optimizator se je izkazal kot učinkovit.

Vsi nadaljni poskusi so bili izvedeni z dvojnim outputom in sicer z pričakovao vrednostjo in standardnim odklonom. Nova ocenjevana spremnljivka je bil tako **Sharpe ratio**: $Sharpe = \frac{EV}{std}$. Izkaže se, da pri malem številu iger pride do zelo velike variance med posameznimi poskusi, zato je ta pristop (vsaj v teoriji) bolj robusten.

4.17 Poskus 17

Vhodni podatki: izkupiček posamezne igre
Model: linearni model z *embedding* plastjo
Rezultati testiranja: Prednost delilca okoli 6%.
Komentar: Zelo počasno učenje, težava z batch$_s izein posledino ocenostandardnega odklona$.

4.18 Poskus 18

Vhodni podatki: izkupiček posamezne igre
Model: linearni model z *embedding* plastjo
Rezultati testiranja: Pragovi optimizirani z isto mrezo preko soft praga (sigmoid). Prednost delilca okoli 5%.
Komentar: Soft pragovi se izkažejo kot neučinkoviti, a z hard mejami to ni izvedljivo (ni diferenciabilnosti).

4.19 Poskus 19

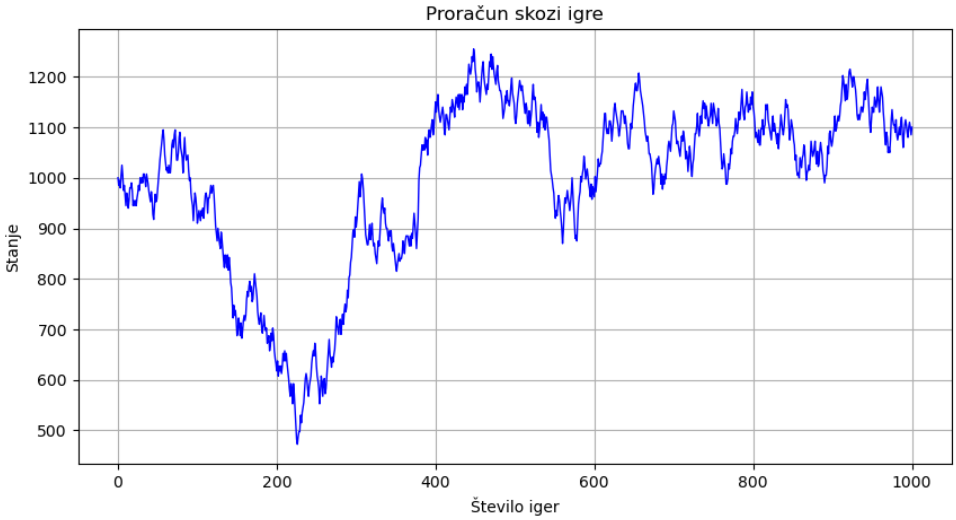
Vhodni podatki: izkupiček posamezne igre
Model: linearni model z *embedding* plastjo in poolingom
Rezultati testiranja: Prednost delilca okoli 6%.
Komentar: Premalo computing power-ja, za tako veliko količino podatkov, zato je bil za občutek pognan na manjšem vzorcu.

4.20 Poskus 20

Vhodni podatki: izkupiček posamezne igre
Model: linearni model z *embedding* plastjo in poolingom
Rezultati testiranja: Enako kot pri 18. Prednost delilca okoli 7%.
Komentar: Enako kot pri 19.

Primer igranja algoritma

```
Odigranih je bilo 1000 iger
Preflop: 371, Flop: 177, River: 276, Fold: 176
Betted: 20570, Končni proračun: 1100.0
Dobiček: 0.49%
Najvišje stanje: 1255.0 v igri 448
Three of a Kind: 43
Straight: 40
Flush: 27
Full House: 28
Four of a Kind: 1
Straight Flush: 0
Royal Flush: 0
```



Slika 2: Primer igranja algoritma.

5 Problemi

Med izvajanjem projekta sva naletela na več izzivov, ki so pomembno vplivali na kakovost rezultatov in hitrost eksperimentiranja:

- **Vhodna oblika podatkov:** izbira načina predstavitve kart (npr. *one-hot* kodiranje, $(rang, barva)$, *embedding* vektorji) se je izkazala za ključni dejavnik uspešnosti modela. Napačna reprezentacija močno oteži učenje relevantnih relacij.
- **Sigmoidna funkcija:** uporaba sigmoidne aktivacijske funkcije je povzročila, da so se vrednosti izhoda pogosto zgoščale v bližini 0 ali 1, kar je otežilo stabilno določanje pragov za stave.
- **Omejena računska moč:** zaradi omejenih zmogljivosti uporabljenih računalnikov je bilo treniranje modelov počasno, kar je zmanjšalo možnost izvedbe večjih eksperimentov in podrobnejšega iskanja optimalnih nastavitev.
- **Čas treniranja:** posamezni poskusi so zahtevali veliko časa za učenje, še posebej pri večjih podatkovnih zbirkah, kar je otežilo iterativno izboljševanje modelov.
- **Narava igre:** zasnova igre Texas Hold'em daje dolgoročno prednost delilcu. To pomeni, da model v resnici ne trenira za *zmago*, temveč za zmanjševanje izgub oziroma iskanje strategije, ki vodi v najmanjši možni poraz.

6 Za n igralcev

6.1 Podatki

Za ta sklop poskusov sva uporabila podatke, generirane s **povprečenjem iger**, pri čemer je bilo v simulaciji vključenih **štiri igralce**. Tako kot pri prejšnjih eksperimentih je bil cilj oceniti uspešnost modela pri napovedovanju optimalne strategije.

6.2 Metodologija

Metodološki pristop je bil zasnovan podobno kot pri poskusih z enim igralcem. Uporabila sva tako **linearne** kot tudi **konvolucijske modele**, pri čemer sva za določanje stavnih pragov izvedla dodatno **optimizacijo**.

Rezultati testiranja: Kljub uporabi optimizacije so bili rezultati nezadovoljivi, saj je delilec ohranil prednost približno 5.3%.

Komentar: Glavni razlog za slabše rezultate je bilo verjetno **premalo podatkov**, da bi model uspel pravilno razbrati vlogo vseh kart in interakcij med več igralci.

7 Rezultati

Rezultati eksperimentov kažejo, da uporaba različnih reprezentacij vhodnih podatkov (povprečenje iger, binarni izid, izkupiček posamezne igre) ter različnih arhitektur modelov (linearni in konvolucijski modeli z *embedding* plastmi) vpliva na končno uspešnost.

Povprečenje iger

Pri uporabi povprečenja iger so linearni in konvolucijski modeli z *one-hot* kodiranjem ali kodiranjem kart kot $(rang, barva)$ dosegali slabe rezultate. Modeli niso uspeli izluščiti uporabnih vzorcev, treniranje je bilo dolgotrajno, prednost delilca pa se je ohranila na ravni približno 5%.

Binarni izid

Pri prehodu na binaren izid (zmaga/poraz) so modeli dosegali nekoliko stabilnejše napovedi, vendar brez izboljšanja končnega rezultata. Delilec je v povprečju obdržal prednost med 6% in 7%, pri čemer se je izkazalo, da ocenjevanje zgolj zmage brez dejanskega izkupička poda popačeno sliko uspešnosti strategije.

Izkupiček posamezne igre

Pri uporabi dejanskega izkupička posamezne igre so bili rezultati primerljivi, a je bila za enako kakovost rezultatov potrebna bistveno večja količina podatkov. Delilec je tudi tukaj ostal v prednosti približno 4–5%.

Optimizacija pragov

Najboljše rezultate sva dosegla z **optimizacijo pragov** za posamezne kroge igre z uporabo funkcije `minimize`. V tem primeru je bilo mogoče zmanjšati prednost delilca na približno 3%. To predstavlja najbolj obetaven pristop, saj kaže, da je z ustrezno optimizacijo mogoče nekoliko zmanjšati naravno prednost igre.

Sklepne ugotovitve

Kljub različnim pristopom in arhitekturam so modeli le delno zmanjšali prednost delilca. Najboljši doseženi rezultat je bil približno 3% v prid delilca, kar ne doseže rezultata, kot sva ga dosegla z lastnimi strategijami.

Nadaljne delo

V okviru projekta sva poskušala z modeli, ki smo jih spoznali pri tem predmetu, a se je za konkreten problem izkazalo, da so potrebni bolj specializirani pristopi. Za nadaljnje delo bova v lastnem režimu razsikala še bolj napredne metode, kot so **reinforcement learning** in **Monte Carlo Tree Search**, ki so se v podobnih problemih izkazale kot zelo učinkovite. Zadnji korak pa je tudi implementacija igre v spletno aplikacijo, kjer bi lahko model uporabljali tudi drugi. Na koncu sva vseeno mnenja, da je bil projekt uspešen, saj sva se naučila veliko novega in pridobila izkušnje z delom na realnem problemu, spoznala veliko novih tehnik in orodij ter izboljšala svoje programerske sposobnosti. Naučila sva se, da je za dober model potrebno veliko časa in predvsem dobro razumevanje problema, da se lahko na koncu "izpili"vsaka malenkost za najboljši možni rezultat.