7. Metoda Monte Carlo

Zadání:

Metoda Monte Carlo představuje rodinu metod a filozofický přístup k modelování jevů, který využívá vzorkování prostoru (například prostor čísel na herní kostce, které mohou padnout) pomocí pseudonáhodného generátoru čísel. Jelikož se jedná spíše o filozofii řešení problému, tak využití je téměř neomezené. Na hodinách jste viděli několik aplikací (optimalizace portfolia aktiv, řešení Monty Hall problému, integrace funkce, aj.). Nalezněte nějaký zajímavý problém, který nebyl na hodině řešen, a získejte o jeho řešení informace pomocí metody Monte Carlo. Můžete využít kódy ze sešitu z hodin, ale kontext úlohy se musí lišit.

Řešení:

Mým úkolem bylo zvolení problematiky, kterou jsem chtěl podle metody Monte Carlo následně zkoumal.

Zvolil jsem si průzkum pravděpodobnosti, když uhodnu nějaký určitý rozsah čísel ve sportce. Poté jsem pomocí této metody zjistil výslednou pravděpodobnost určité modelové situace. Jako první jsem si zvolil pravidla pro návrh sportky.

Princip sportky a vyhodnocení

Vždy vylosuji šestici náhodných čísel z rozsahu od 1 do 48. Následně po vylosování z pozice hry přijde řada na vylosování bonusového čísla, které je ze stejného rozsahu.

```python
#SPORTKA
class Sportka:
    def __init__(self):
        self.vylos_cisla = []
        self.bonus_cislo = []
        self.hrac_hl_losovani = []
        self.hrac_bonus_num = []

    def losovani(self, pocet_cisel):
        self.vylos_cisla.clear()
        for _ in range(pocet_cisel):
            pick = random.randint(1,48)
            self.vylos_cisla.append(pick)
        #print("Vylosovaná čísla:",self.vylos_cisla)
        return

    def bonus_num(self):
        self.bonus_cislo.clear()
        bonus_pick = random.randint(1,48)
        self.bonus_cislo.append(bonus_pick)
        print("----------------------------BONUSY:----------------------------
        print("Bonusové číslo:",self.bonus_cislo)
        return
```

Následně přijde řada pro losování z pozice hráče. Ten si také vylosuje svých základních 6 čísel, následně svoje bonusové číslo.

```python
#HRÁČ

    def losovani_hrace(self,pocet_cisel):
        self.hrac_hl_losovani.clear()
        for _ in range(pocet_cisel):
            pick = random.randint(1, 48)
            self.hrac_hl_losovani.append(pick)
        #print("Vylosovaná čísla HRÁČE:",self.hrac_hl_losovani)
        return

    def bonus_hrac(self):
        self.hrac_bonus_num.clear()
        bonus_pick = random.randint(1,48)
        self.hrac_bonus_num.append(bonus_pick)
        print("Bonusové číslo HRÁČ:",self.hrac_bonus_num)
        return
```

Po vylosování čísel hráče a sportky přijde řada na vyhodnocení losu. Jako první jsem definoval dvě globální proměnné: výhra a status. Výhra mi vyhodnocuje vyhranou částku v Kč a status je pro mě logická proměnná, která určitě výhru, nebo prohru.

Následně jsem porovnával shodu vylosovaných hodnot pomocí funkce intersection(), kterou jsem získal průniky (shodná čísla), které se objevovaly v listech losovaných čísel hráče a sportky. Pomocí rozměru výsledného průniku množin jsem zjistil počet shodných čísel pomocí funkce len(), která mi vrátila počet prvků v průniku.

```
#vyhodnoceni kola
spravne_cisla = set(self.hrac_hl_losovani).intersection(self.vylos_cisla)
pocet_spravnych_cisel = len(spravne_cisla)

pocet_bonus = set(self.bonus_cislo).intersection(self.hrac_bonus_num)
bonus_inter = len(pocet_bonus)
```

Po získání počtu shodných čísel přišel čas na vyhodnocení celkového losování pomocí podmínek. Pokud je počet správných čísel roven určitému intu, tak nastala výhra losu a k logické hodnotě status byla přiřazena jednička (ta symbolizuje výhru tiketu). Na konci se také vyhodnocuje výherní bonus. Pokud nám vyšel i bonus, tak se k statusu přičte jednička.

Zároveň kromě statusu přiřazuji hodnoty k proměnné výhra, která nám určuje peněžní odměnu na daném losu.

Vyhodnocení tiketu

```
if pocet_spravnych_cisel == 6:
    vyhra = bank
    status = 1
    print("Jackpot! Všechna čísla uhodnuta!", vyhra ,"Kč!")

elif pocet_spravnych_cisel == 5:
    vyhra = 50000
    status = 1
    print("Gratulujeme, máte výhru!", vyhra ,"Kč!")

elif pocet_spravnych_cisel == 4:
    vyhra = 1200
    status = 1
    print("Gratulujeme, máte výhru!", vyhra ,"Kč!")

elif pocet_spravnych_cisel == 3:
    vyhra = 100
    status = 1
    print("Gratulujeme, máte výhru!", vyhra ,"Kč!")

elif pocet_spravnych_cisel == 2:
    vyhra = 40
    status = 0
    print("Gratulujeme, máte výhru!", vyhra ,"Kč!")

else:
    print("Bohužel, nemáte výhru!", vyhra ,"Kč!")
    status = 0

if bonus_inter > 0:
    bonus_vyhra = bonus_inter * 5000
    vyhra += bonus_vyhra
    status += 0
    print(f"Vyšel Vám bonus: {vyhra} Kč")
return status
```

Z důvodu počítání pravděpodobnosti vracím z funkce hodnotu status.

Implementace Monte Carlo

Vytvořím si prázdný seznam, do kterého budu zaznamenávat okamžitou pravděpodobnost a proměnnou součet hodnot, do které budu přičítat výsledek iterace.

Potom v cyklu s počtem iteraci volám funkce z classy Sportka, které slouží k zahájení losování a k následnému výsledku. U konce každého cyklu ukládám výsledek iterace (výhru/prohru) a tu přičítám k součtu hodnot. Následně vypočítá okamžitou pravděpodobnost v procentech a přidám jí do seznamu.

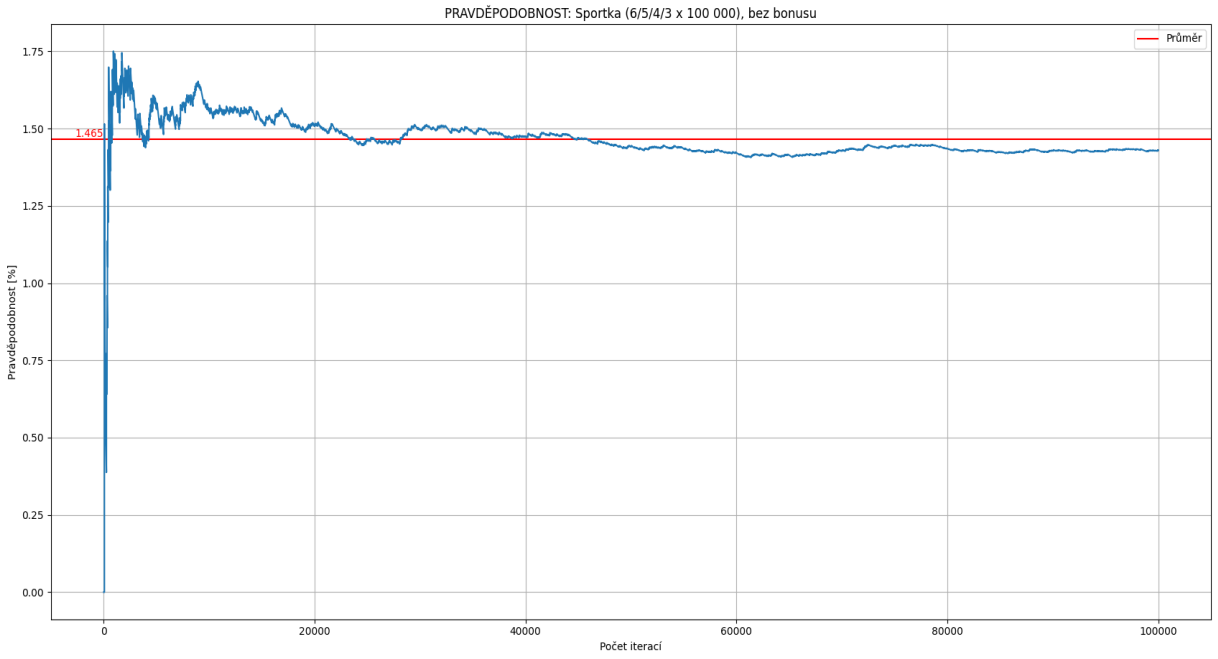
```
def monte_carlo(self, pocet_iteraci):
    okamzite_pravdepodobnost = []
    soucet_hodnot = 0

    for pokus in range(pocet_iteraci):
        self.losovani(6)
        self.bonus_num()
        self.losovani_hrace(6)
        self.bonus_hrac()


        vysledek_iterace = self.vyhodnotit_vysledky()


        soucet_hodnot += vysledek_iterace
        print("SOUČET HODNOT:", soucet_hodnot)

        okamzite_pravdepodobnost.append(soucet_hodnot / (pokus + 1)*100)
```

Po vyhodnocení přišel čas na grafický výstup celé problematiky. Pomocí knihovny matplotlib vytvořím následně spojnicový graf, na kterém zobrazuji okamžitou pravděpodobnost vůči počtu iterací. Před celkovým zobrazení jsem spočítal výsledný průměr pomocí knihovny numpy, který zobrazuji pro představu průměrné pravděpodobnosti.

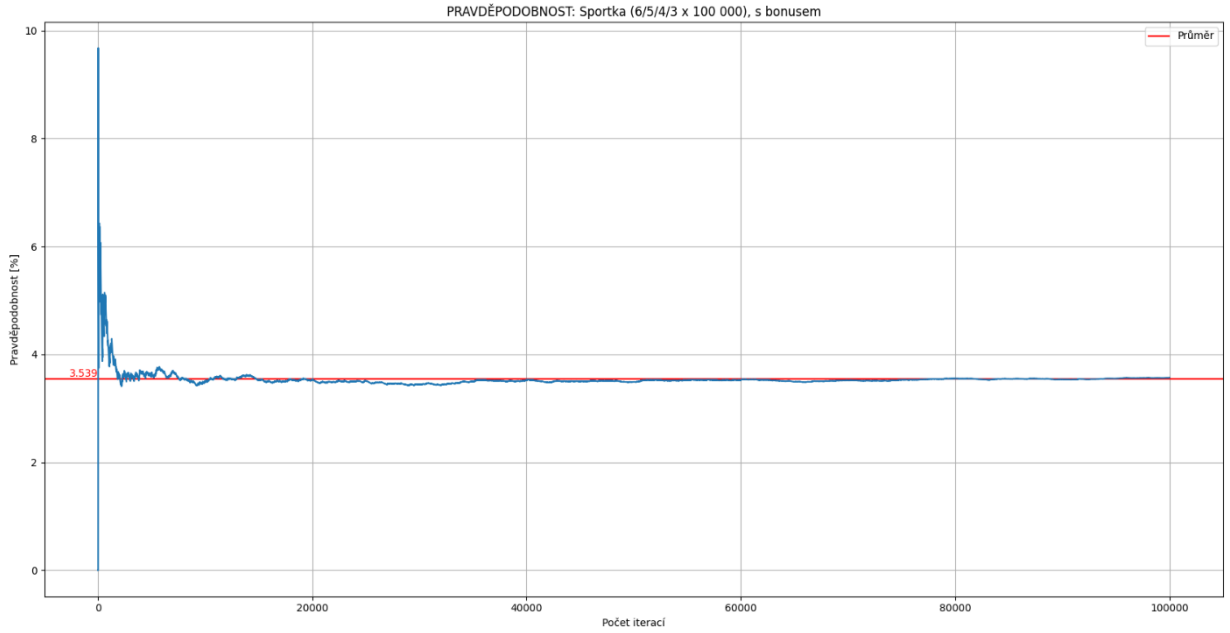
Při formátování grafu jsem přidal na každou osu labely (popisky) a legendu pro průměrnou hodnotu. Také jsem použil do grafu implementoval mřížku pro lepší orientaci a také jsem definoval velikost znaků na každé ose. A úplně na konec jsem přidal funkci tight_layout pro optimalizaci zobrazení grafu.

```python
#VÝPOČET PRŮMĚRU
avg = np.average(okamzite_pravdepodobnost)
print("AVARAGE JE:", avg)

#FORMÁT GRAFU - VÝSTUP
plt.title("PRAVDĚPODOBNOST: Sportka (6/5/4/3 x 100 000), s bonusem")
plt.axhline(y=avg, color="r", linestyle="-")
plt.xlabel("Počet iterací")
plt.ylabel("Pravděpodobnost [%]")
plt.legend(['Průměr'], loc='upper right')

# Přidání popisku s hodnotou avg
plt.text(0.5, avg, f"{round(avg,3)}", ha='right', va='bottom', color='r')

plt.xticks(fontsize=10)  # Velikost čísel na ose x
plt.yticks(fontsize=10)  # Velikost čísel na ose y
plt.grid(True)  # Zobrazení mřížky

plt.tight_layout()  # Optimalizace umístění os a popisků

plt.plot(okamzite_pravdepodobnost)
plt.show()
```

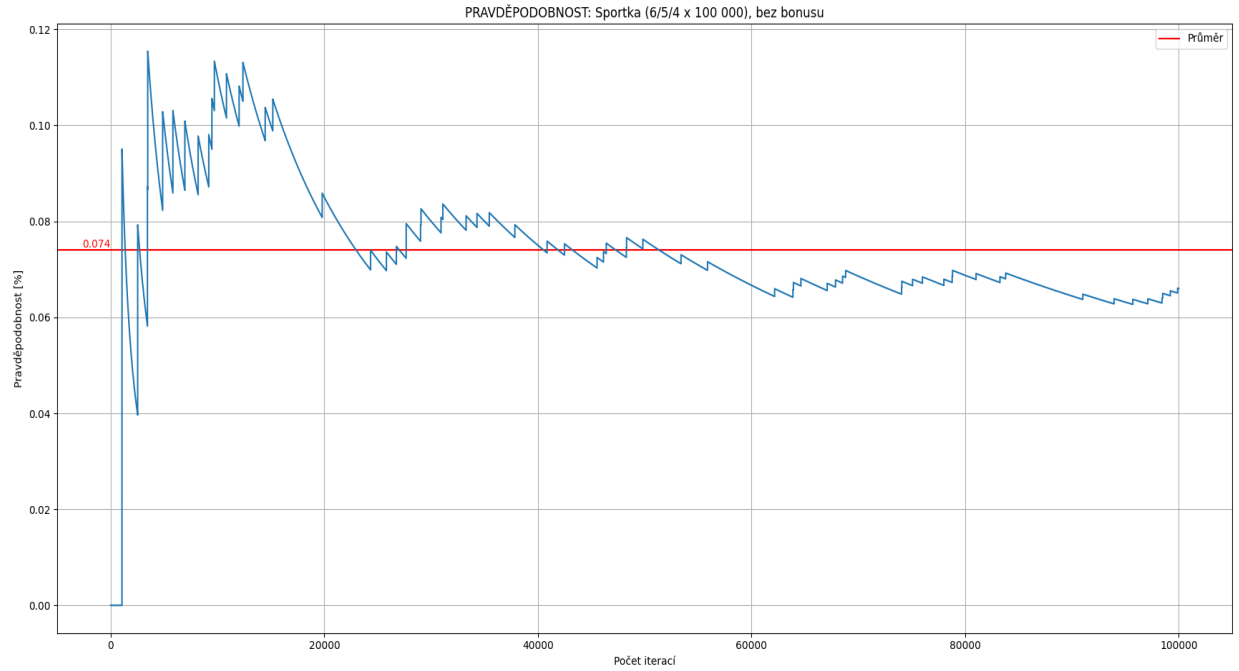
Výstupy



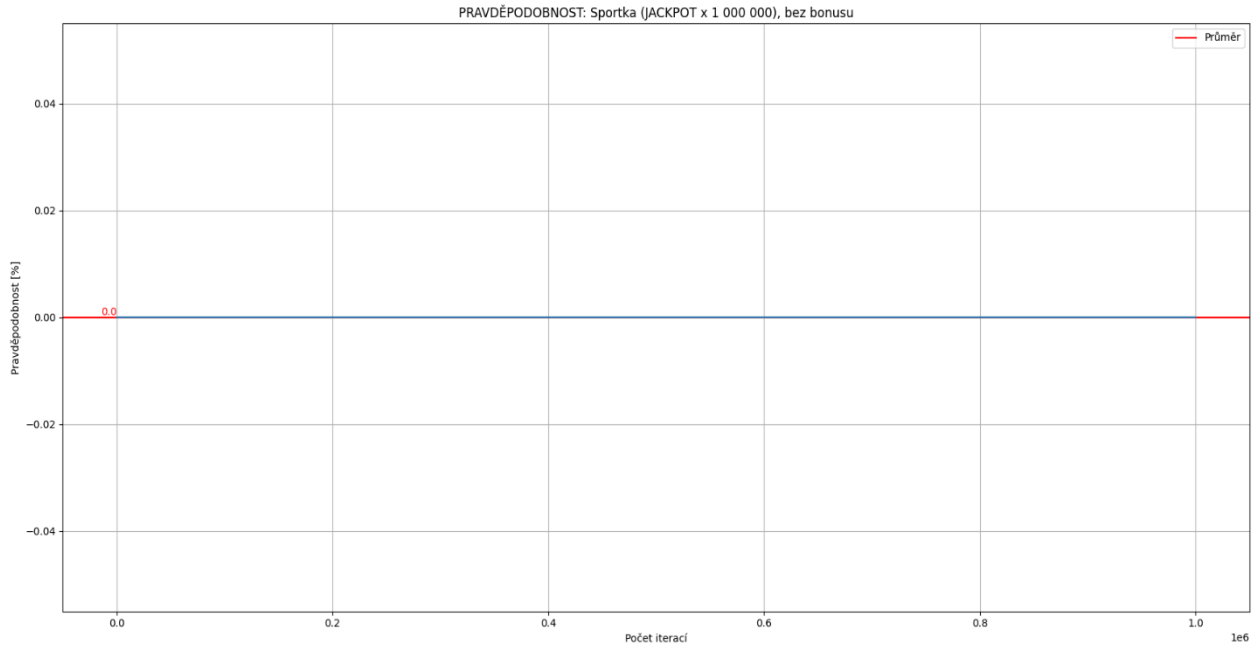
*Obr. 7 Výsledná pravděpodobnost **pro 3 až 6 shodných čísel bez bonusu** na 100 000 iterací vyšel kolem **1,465%.***



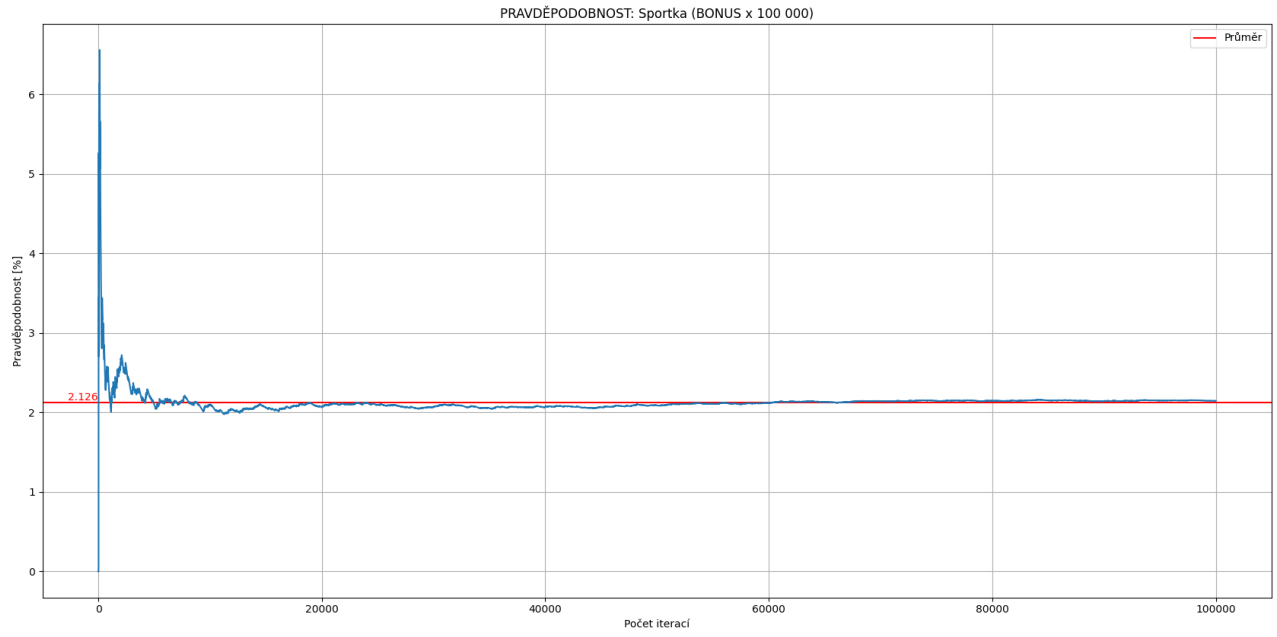
*Obr. 8 Výsledná pravděpodobnost **pro 3 až 6 shodných čísel s bonusem** na 100 000 iterací vyšel kolem **3,539%.***



*Obr. 9 Výsledná pravděpodobnost **pro 4 až 6 shodných čísel bez bonusu** na 100 000 iterací vyšel kolem **0,074%.***



*Obr. 10 Výsledná pravděpodobnost **pro JACKPOT bez bonusu** na 1 000 000 iterací vyšel **0% (po 30 000 000 iterací mi JACKPOT padl pouze 1x).***



*Obr. 11 Výsledná pravděpodobnost **pro BONUS** na 100 000 iterací vyšel **2,124%.***

ODKAZ NA GITHUB:

https://github.com/Marty808s/Monte-Carlo

- Ve složce MONTE CARLO jsou také grafy.