Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра математики

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: «**СТОХАСТИЧНИЙ ВАРІАНТ ЗАДАЧІ ПРО НАРЕЧЕНУ**»

Виконав: студент 4-го року навчання,
Освітньої програми «Прикладна математика», 113

Авдєєнко Іван Максимович

Керівник Щестюк Н. Ю.,_____
кандидат фіз.-мат. наук, доцент

Рецензент _____
(прізвище та ініціали)

Кваліфікаційна робота захищена
з оцінкою _____

Секретар ЕК _____
«____» _____ 20____ р.

Київ – 2023

ЗМІСТ

ВСТУП

В даній роботі мова буде йти про задачу, яка у свій час дала початок новому розділу теорії ймовірностей, а саме, теорії оптимальної зупинки випадкових процесів. Мова буде йти про задачу про «перебірливу наречену», або як її називають в літературі англійською мовою «проблема секретаря». Ця задача також стосується галузей прикладної теорії ймовірностей, статистики та теорії прийняття рішень. Вперше ця задача була опублікована Мартіном Гарднером в журналі «Scientific American» в лютому 1960. Класичний варіант задачі про перебірливу наречену формулюється так:

1. Існує фіксоване та відоме число n претендентів на одну позицію, яких можна розташувати в порядку якості.
2. Претенденти розглядаються послідовно у випадковому порядку.
3. Для кожного претендента j наречена може дізнатися лише відносний ранг претендента, тобто, наскільки цінним він є у порівнянні з j - 1 раніше розглянутими претендентами.
4. Відхиленого претендента не можна повернути. Якщо досягнуто n-го претендента, його необхідно прийняти.
5. Наречена перемагає (тобто отримує винагороду в 1) за вибір претендента з абсолютним рангом 1 (тобто, найкращого претендента в загальному списку з n претендентів) та програє (тобто отримує винагороду в 0) в будь-якому іншому випадку.

Проблему можна назвати актуальною, оскільки у житті ми часто зіштовхуємося з проблемами прийняття рішень: при виборі житла на обмеженому ринку взяти перший прийнятний варіант чи почекати кращого? При продажу будинку прийняти пропозицію чи почекати кращої пропозиції, якої може і не бути? При пошуку АЗС на автомагістралі наскільки довго шукати дешевше пальне? При торгівлі на фондовому ринку в який момент прийняти рішення про купівлю чи продаж активу?

У даній роботі більш детально буде розглянуто стохастичний варіант задачі. Стохастичність передбачає наявність невизначеності та елементів випадковості. Це означає, що майбутні результати не можуть бути точно передбачені, але можуть бути описані ймовірнісними розподілами. Стохастичні моделі дозволяють більш реалістично відобразити людську поведінку в умовах невизначеності. Стохастичні моделі вибору в задачі про наречену дозволяють дослідникам вивчати, як особи, які приймають рішення, зважують компроміс між потенційним виграшем від вибору кандидата з вищим рейтингом і ризиком, пов’язаним із таким вибором. Ці моделі можуть надати розуміння факторів, які впливають на прийняття рішень, таких як переконання особи, яка приймає рішення, уподобання та терпимість до ризику.

Поєднуючи ідеї застосування оптимальної стратегії та стохастичних моделей, можна отримати глибше розуміння процесу прийняття рішень у задачі про наречену та вивчити взаємодію між раціональним прийняттям рішень і поведінкою людини в умовах невизначеності.

РОЗДІЛ 1. ІСТОРІЯ ЗАДАЧІ

Вперше проблема секретаря була представлена в 1949 році американським математиком Меррілом М. Флудом, який назвав її проблемою нареченої в лекції, яку він прочитав того року. Він кілька разів посилався на цю лекцію протягом 1950-х років, наприклад, у виступі на конференції в Перд'ю 9 травня 1958 року, і згодом задача стала широко відомою, хоча тоді ще не було опубліковано жодних досліджень на цю тему. У 1958 році він надіслав листа Леонарду Гіллману з копіями багатьом друзям, в якому виклав доказ оптимальної стратегії з додатком від Р. Палермо, який довів, що в усіх стратегіях домінує стратегія «за будь-яких умов відхилити перших р претендентів, а потім прийняти наступного кандидата, який кращий»

Першу публікацію зробив популяризатор математики Мартін Гарднер у журналі Scientific American в лютому 1960 року. Він почув про цю задачу від Джона Х. Фокса молодшого та Л. Джеральда Марні, які незалежно один від одного придумали еквівалентну проблему в 1958 році; вони назвали це «грою в гугол». Фокс і Марні не знали оптимального рішення; Гарднер звернувся за порадою до Лео Мозера, який надав правильний аналіз для публікації в журналі. Незабаром після цього кілька математиків написали Гарднеру, щоб розповісти йому про еквівалентну проблему, і все це, швидше за все, було відображено в оригінальній роботі Флуда.

Фергюсон має велику бібліографію і вказує на те, що подібна (але інша) проблема розглядалася Артуром Кейлі ще в 1875 році і навіть Йоганном Кеплером задовго до цього.

РОЗДІЛ 2. КЛАСИЧНИЙ ВАРІАНТ ЗАДАЧІ ПРО НАРЕЧЕНУ ТА ЇЇ ОПТИМАЛЬНА СТРАТЕГІЯ

Формулювання задачі наведено у вступній частині. Оптимальна стратегія для задачі полягає в максимізації ймовірності вибору найкращого кандидата. Оптимальна стратегія формулюється наступним чином. Для деякого цілого числа $r \geq 1$ наречена відхиляє перших $r-1$ кандидатів і приймає наступного кандидата, який є кращим за всіх попередніх розглянутих. Для такої стратегії ймовірність $\Phi_n(r)$ вибору найкращого кандидата дорівнює $1/n$ для r = 1 і для $r > 1$,

$$\Phi_n(r) = \sum_{j=r}^{n} P(j - \text{й кандидат найкращий і був обраний}) =$$

$$= \sum_{j=r}^{n} \left(\frac{1}{n}\right)\left(\frac{r-1}{j-1}\right) = \left(\frac{r-1}{n}\right) \sum_{j=r}^{n} \frac{1}{j-1}$$

Оптимальним значенням r є таке, що максимізує цю ймовірність. У випадку $n \to \infty$ запишемо x як ліміт r/n, тоді використовуючи t для j/n і dt для $1/n$ сума перетворюється на інтеграл

$$\Phi_n(r) = \left(\frac{r-1}{n}\right) \sum_{j=r}^{n} \left(\frac{n}{j-1}\right)\left(\frac{1}{n}\right) \to x \int_x^1 \left(\frac{1}{t}\right) dt = -x \ln x$$

Значення x, яке максимізує цю величину, легко знайти, прирівнявши похідну від x до нуля, а потім розв'язавши для x. Коли це зроблено, ми знаходимо, що оптимальне $x = 1/e \approx 0.3678$ і
максимальна ймовірність успіху для оптимальної стратегії $= 1/e$
Це означає, що для для великого n потрібно пропустити приблизно 37% претендентів і після цього прийняти першого кандидата кращого за всіх попередніх. Дотримуючись такої стратегії ймовірність успіху буде складати також приблизно 37% [1]

РОЗДІЛ 3. ЗАГАЛЬНИЙ ВАРІАНТ ЗАДАЧІ ПРО НАРЕЧЕНУ ТА ЇЇ ОПТИМАЛЬНА СТРАТЕГІЯ

Формулювання загального варіанту

Розглянемо варіант задачі у якому наречена отримує деяку додатню винагороду $\pi(a)$ за вибір претендента з абсолютним рангом a і припустимо, що

$$\pi(1) \geq \cdots \geq \pi(n)$$

Оптимальна стратегія зупинки пошуку має подібну форму порогу як для класичної задачі. Відмінність полягає в тому, що наречена повинна провести зустріч з першими $t_1 - 1$ претендентами та відмовити їм усім, потім між претендентом t_1 та претендентом $t_2 - 1$ вона повинна прийняти лише найкращого претендента (тобто претендента з відносним рангом 1) ; між претендентом t_2 та претендентом $t_3 - 1$ вона повинна прийняти одного з двох найкращих претендентів, тобто кандидатів з відносними рангами 1 або 2; і так далі. Чим далі вона відкладає момент прийняття рішення, тим нижчі вона дозволяє собі стандарти і тим більш ймовірно, що вона приймає претендентів нижчої якості. Але в будь-якому разі, не зважаючи на ризик обрати не найкращого кандидата, ймовірність отримати винагороду $\pi(a) > 0$ зростає в порівнянні з класичним варіантом задачі де лише найкращий претендент дозволяє отримати додатну винагороду. Таким чином, при умові, що наречену влаштовує один з двох найкращих претендентів, ймовірність отримати ненульову винагороду зільшується до 79%, що було доведенно М. Тамакі у 1979 році. Таким чином ми отримуємо те, що називається Загальним варіантом задачі про наречену, замінивши пункт 5 з початкової умови на

5’. Наречена отримує винагороду π(a) за вибір претендента з абсолютним рангом a де π(1)≥⋯≥π(n)

Насправді класичний варіант задачі є особливим випадком загального, в якому $\pi(1) = 1$ та $\pi(a) = 0$, для всіх $a > 1$.

Знаходження оптимальної стратегії для загального варіанту задачі

Почнемо зі введення деякого уточнення. Порядок абсолютних рангів n претендентів представляється вектором $a = (a^1, \dots, a^n)$, який є випадковою перестановкою цілих чисел від 1 до n. Відносний ранг претендента j позначається як r^j, це число претендентів від $1, \dots, j$, чий абсолютний ранг менше або дорівнює a^j. Позначимо стратегію як вектор $s = (s^1, \dots, s^n)$ невід’ємних цілих чисел для якого $s^j \leq s^{j+1}$для будь яких $1 \leq j < n$. Стратегія вказує на те, що наречена зупиняється на першому кандидаті для якого $r^j \leq s^j$. Отже, ймовірність того, що наречена зупинить свій вибір на претенденті під номером j, за умови досягнення цього претендента дорівнює s^j/j. Позначимо цю ймовірність як Q^j. Поріг вибору нареченої для обрання кандидата з відносним рангом r позначається як t_r, - це найменше значення j для якого $r \leq s^j$. Таким чином стратегія s також може бути представлена вектором $t = (t_1, \dots, t_n)$. Представлення за допомогою порогових значень іноді буде зручнішим.

Оптимальні порогові значення можуть бути обчислені простим поєднанням числових методів пошуку з методами динамічного програмування. Далі ми розглянемо процедуру для виконання цього. Подібний метод раніше був описаний у роботі Ліндлі (1961) та коротко описаний Yeo and Yeo (1994).
Ймовірність того, що $j-$ий претендент з загальної кількості n, чий відносний ранг r^j має абсолютний ранг a, була запропонована Лінделі в 1961 р.:

$$\Pr(A = a|\, R = r^j) = \frac{\binom{a-1}{r-1}\binom{n-a}{j-r}}{\binom{n}{j}},$$

Де $r^j \le a \le r^j + (n-j)$; в іншому випадку $\Pr(A = a|\, R = r^j) = 0$.
Таким чином очікувана винагорода за вибір кандидата з відносним рангом r^j дорівнює:

$$E\left(\pi^j \middle| r^j\right) = \sum_{a=r^j}^{n} \Pr(A = a|\, R = r^j)\, \pi(a).$$

Очікувана винагорода за вибір на етапі j для етапу j стратегії $s^j > 0$ дорівнює:

$$E\left(\pi^j \middle| s^j\right) = \left(s^j\right)^{-1} \sum_{i=1}^{s^j} E\left(\pi^j \middle| r^j = i\right);$$

В іншому випадку, коли $s^j = 0,\ E\left(\pi^j \middle| s^j\right) = 0$. Тепер, позначаючи очікувану винагороду починаючи з етапу $j+1$ і потім дотримуючись стратегії фіксованих порогів $(s^{j+1}, \dots, s^n)$ як v^{j+1}, значення v^j для будь-яких $s^j \le j$ буде дорівнювати:

$$v^j = Q^j\, E\left(\pi^j \middle| s^j\right) + \left(1 - Q^j\right) v^{j+1}$$

Оскільки очікуваний прибуток від оптимальної стратегії на етапі n дорівнює

$$v^n = n^{-1} \sum_{a=1}^{n} \pi(a),$$

Ми можемо знайти s^j для кожного j $(j = n-1, \dots, 1)$ яке максимізує v^j шляхом пошуку через допустимі s^j; очікувані винагороди оптимального порогу s^{j*} позначаємо як v^{j*}. За обмеженням монотонності на s^j пошук може бути обмежений до $0 \le s^j < s^{j+1}$. Отже, виходячи з v^{n*}та починаючи з етапу $n-1$ і рухаючись у зворотному напрямку процедура динамічного програмування для знаходження оптимальної стратегії для загального варіанту задачі може бути узагальнена таким чином:

$$s^{j*} = \arg \max_{s \in \{0, \dots, s^{j+1*}\}} v^j$$

Очікувана винагорода за дотримання стратегії s буде дорівнювати:

$$E(\pi|s) = \sum_{j=1}^{n} \left[\prod_{i=0}^{j-1} (1 - Q^i) \right] Q^j E\left(\pi^j \middle| s^j = v\right), \text{ де } Q^0 = 0 \qquad (2.1)$$

Оптимальна стратегія s^*це така стратегія s, яка максимізує рівняння 2.1.
Позначаючи позицію кандидата на якій пошук припиняється як m, ймовірність того, що наречена зупиниться на $(j < n) -$ му претенденті дорівнює:

$$\Pr(m = j) = \left[\prod_{i=0}^{j-1}(1 - Q^i)\right] Q^j$$

І очікування позиція зупинки пошуку буде дорівнювати (за Морігуті, 1993):

$$E(m) = 1 + \sum_{j=1}^{n-1}\left[\prod_{i=1}^{j}(1 - Q^i)\right] \quad (2.2)$$

РОЗДІЛ 4. СТОХАСТИЧНА МОДЕЛЬ ВИБОРУ В ЗАДАЧІ ПРО НАРЕЧЕНУ

В попередньому розділі ми визначили, що наречена припиняє пошук на кандидаті j тоді і тільки тоді, коли відносний ранг претендента не перевищує значення порогу для даного етапу, тобто коли $r^j < s^{j*}$.

Використовуючи стохастичну модель ми припускаємо, що порогові значення зупинки на певному етапі можна моделювати як випадкові величини. Кожного разу коли наречена зустрічає кандидата з відносним рангом r, вважається, що вона вибирає поріг з її розподілу порогів для кандидатів з відносним рангом r. Після цього вона приймає рішення про зупинку опираючись на порогове значення цієї вибірки. Самі пороги не спостерігаються, оскільки вони задані не строгим значенням, а певним розподілом. Модель, яку ми розглядаємо така: ми припускаємо, що спостережувана поведінка нареченої відповідає її поведінці так наче вона випадково вибирає порогові значення з певного розподілу, враховуючи обмеження моделі.

Позначаючи порогове значення вибірки як σ_r, наречена зупиняється на претенденті з відносним рангом r^j тоді і тільки тоді, коли $r^j \leq \sigma_r$(тобто коли відносний ранг претендента не перевищує порогове значення вибірки). Також варто зауважити, що на кожному етапі j наречена робить вибірку з розподілу, який залежить від відносного рангу заявника, який спостерігається на цьому етапі. Також розподіл не є умовним лише на етап вибору, він є умовним лише на відносний ранг кандидата який розглядається на цьому етапі.

Ми припускаємо, що функція щільності ймовірності для порогу вибірки задається так:

$$f(\sigma_r) = \frac{e^{-\frac{\sigma_r - \mu_r}{\beta_r}}}{\beta_r \left[1 + e^{-\frac{\sigma_r - \mu_r}{\beta_r}}\right]^2} \quad (4.3)$$

Отже, при умові досягнення цього порогу, ймовірність того, що буде вибрано кандидата з відносним рангом r^jдорівнює:

$$Pr(r^j \leq \sigma_r) = \frac{1}{1 + e^{-\frac{j - \mu_r}{\beta_r}}} \quad (4.4)$$

В формулах (4.3) та (4.4) і надалі β_r – це дисперсія порогового розподілу, μ_r – відповідає середньому значенню порогу для кандидата з відносним рангом r. Це параметр який характеризує центр розподілу порогів.

Ми припускаємо, що $\mu_1 \leq \cdots \leq \mu_n$ та $\beta_1 \geq \cdots \geq \beta_n$. Це припущення ґрунтується на обмеженні з умови задачі, що винагороди не зростають зі збільшенням відносного рангу претендента. Також слушно буде зазначити, що $Pr(r^j \leq \sigma_r) \geq Pr(r'^j \leq \sigma_{r'})$ при будь-яких $r \leq r'$Тому ймовірність того, що наречена зупиниться на певному етапі буде більшою, якщо відносний ранг спостережуваного претендента зменшується

Також важливо зауважити, що стохастична модель наближається до детерміністської (визначеної) коли $\beta_r \to 0$ для всіх r. Оптимальна стратегія також досягається коли β_r є малим (чим ближче до 0 тим краще) і $t_r^* - 1 \leq \mu_r < t_r^*$ для кожного r (тобто коли μ_r максимально наближено до оптимального порогу). Приклади розподілу порогів та ймовірностей зупинки пошуку будуть наведені в практичній частині роботи.

Згідно з моделлю, ймовірність $\hat{Q}^j$ того, що наречена закінчить пошук на претенденті $j < n$ за умови, що вона його досягла дорівнює:

$$\hat{Q}^j = \sum_{r^j=1}^{j} \frac{1}{j} Pr\left(r^j \leq \sigma_r\right)$$

Підставивши $\hat{Q}^j$ в рівняння 2.2 маємо можливість обрахувати очікувані позиції зупинки пошуку. Приклади очікуваних позицій зупинки для різних значень μ_1 та β_1 також детальніше буде розлянуто в аналізі практичної частини.

РОЗДІЛ 5. РЕАЛІЗАЦІЯ ПРАКТИЧНОЇ ЧАСТИНИ

В практичній частині роботи було реалізовано методи пошуку найкращого претендента для класичного та стохастичного варіантів задачі про наречену. Робота виконувалася в середовищі Google Colaboratory.

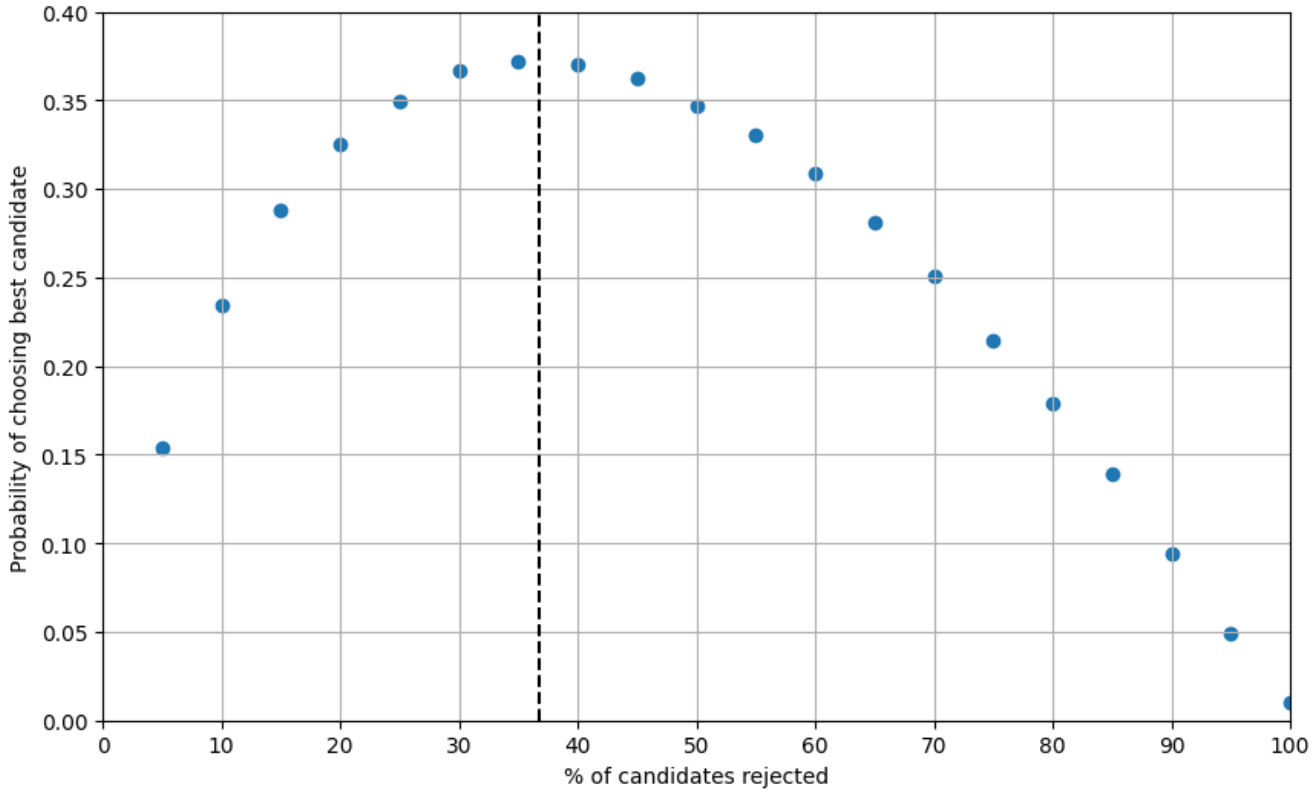
```
def choose_candidate(n, reject=np.e):
  '''Choose a candidate from a list of n candidates using a specified
strategy.

  reject: percentage of candidates to initially reject (optimal strategy by
default)'''

  candidates = np.arange(1, n+1)
  np.random.shuffle(candidates)

  if reject == np.e:
      stop = int(round(n/reject))
  else:
      stop = int(round(reject*n/100))

  best_from_rejected = np.min(candidates[:stop])
  rest = candidates[stop:]

  try:
    return rest[rest < best_from_rejected][0]
  except IndexError:
    return candidates[-1]
```

Для реалізації класичної задачі про наречену було створено функцію `choose_candidate` яка приймає два параметри: n – кількість претендентів на наречену та $reject$ – відсоток претендентів, які потрібно відхилити. За замовчування це значення встановлено як e, що відповідає оптимальній стратегії. Потім всередині функції створюється масив кандидатів $candidates$, де кожному кандидату присвоюється унікальний номер від 1 до n. Після цього кандидати перемішуються у випадковому порядку. Вибір найкращого претендента відбувається за допомогою відхилення заданого числа перших кандидатів. Після цього приймається перший кандидат, чиє значення менше за найкращого з відхилених. Якщо найкращого кандидата не було знайдено, то IndexError виведе останнього кандидата, який залишився

```
best_candidate = []
for r in range(5, 101, 5):
    sim = np.array([choose_candidate(n=100, reject=r) for i in range(100000)])
    # np.histogram counts frequency of each candidate
    best_candidate.append(np.histogram(sim, bins=100)[0][0]/100000)
```

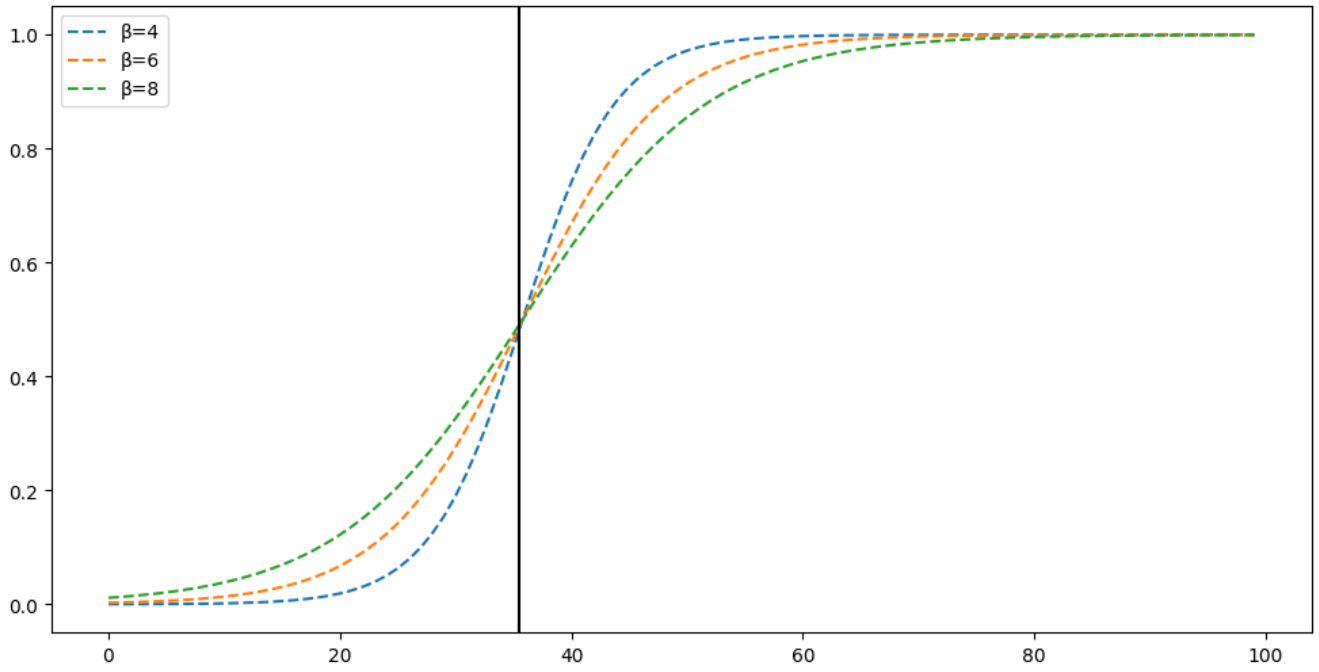
```
plt.figure(figsize=(10, 6))
plt.scatter(range(5, 101, 5), best_candidate)
plt.xlim(0, 100)
plt.xticks(np.arange(0, 101, 10))
plt.ylim(0, 0.4)
plt.xlabel('% of candidates rejected')
plt.ylabel('Probability of choosing best candidate')
plt.grid(True)
plt.axvline(100/np.e, ls='--', c='black')
plt.show()
```

Після цього побудуємо графік ймовірності вибору найкращого кандидата за кількістю відхилених кандидатів. Для цього використаємо бібліотеку matplotlib.pyplot. Графік зображений на рис.1. Як і очікувалось, оптимальна стратегія знаходиться десь між 35% і 40%, тобто 1/e. Імовірність вибору кращого кандидата, коли ми відкидаємо перших n/e кандидатів, також становить 1/e.



Для побудови подальших графіків були застосовані формули з розділу 4, які описують розподіл та залежність ймовірностей вибору залежно від параметрів μ_1 та β. Зокрема наступний графік використовує формулу 4.4 для задання ймовірності вибору найкращого кандидата.

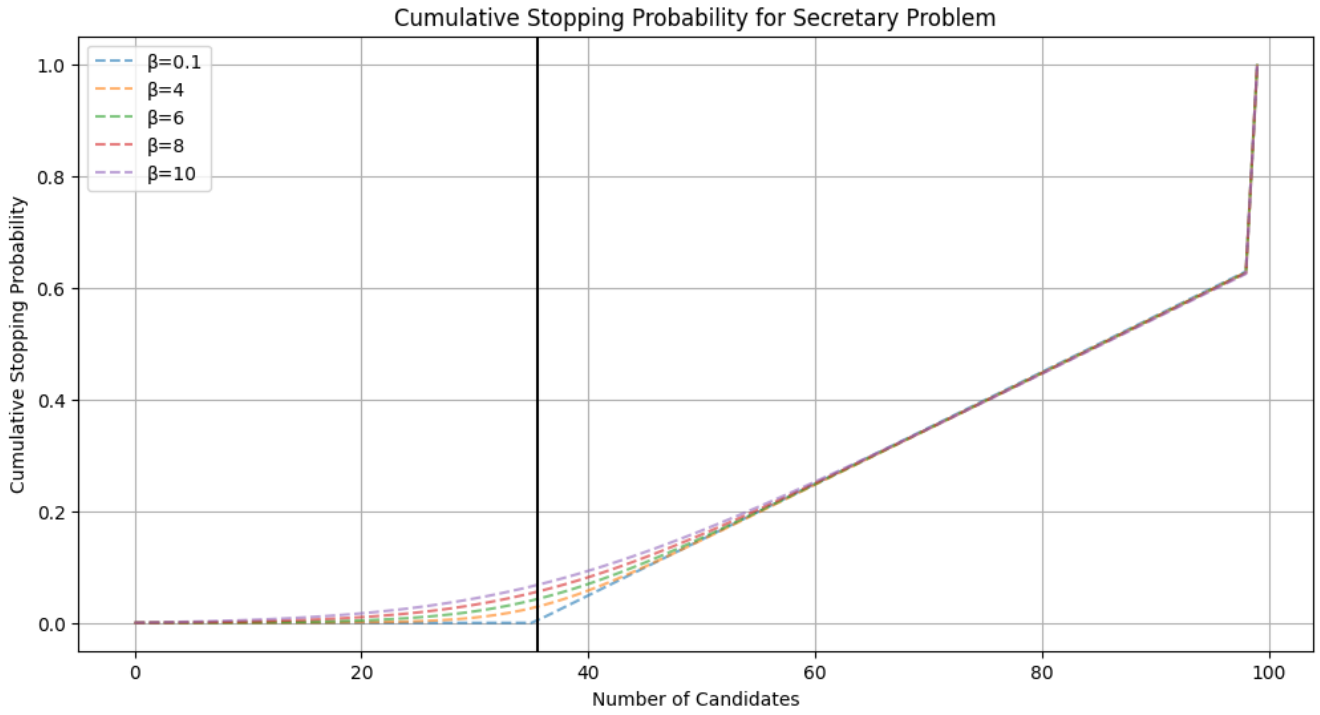
```
# Перший графік

plt.figure(figsize=(12,6))
# Ймовірність вибору кандидата
def selection_probability(j, mu, beta):
    return 1 / (1 + math.exp(-(j - mu) / beta))

num_candidates = 100
mu = num_candidates/np.e #оптимальний поріг
for beta in [4,6,8]:
    stopping_prob = [selection_probability(n, mu, beta) for n in range(1,101)]
    plt.plot(stopping_prob, linestyle='dashed', label=f"β={beta}")

plt.axvline(x = 35.5, color = 'black') #оптимальний поріг
plt.legend(loc="upper left")
plt.show()
```



Цей графік відображає ймовірність вибору найкращого кандидата з врахуванням різних значень параметру β. З графіку можна зробити висновок, що для етапу вибору j, який є меншим за порогове значення μ_1 ймовірність вибору кандидата з відносним рангом 1 збільшується зі зростанням β, і навпаки, після проходження порогу ймовірність зменшується зі зростанням β.

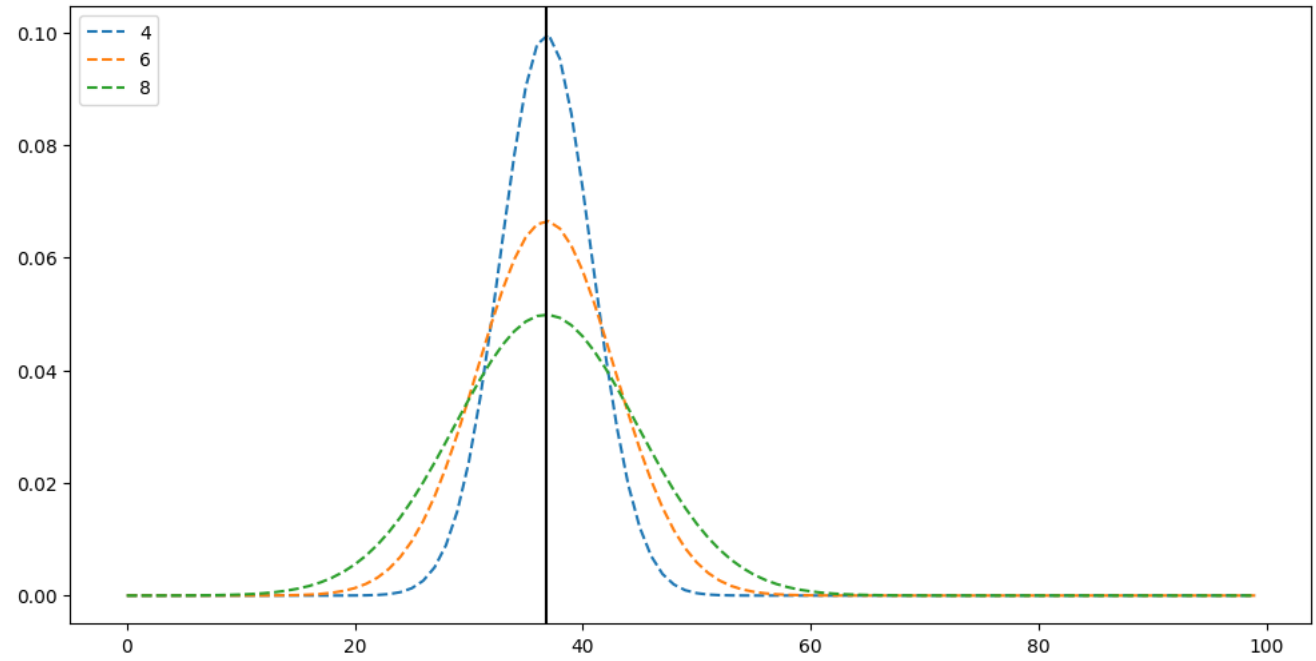
```
# Другий графік
plt.figure(figsize=(12,6))

def cumulative_stopping_probability(num_candidates, mu, beta):
    cumulative_prob = 0.0
    stopping_probs = []

    for j in range(1, num_candidates):
```

```
        conditional_prob = 1 / (1 + math.exp(-(j - mu) / beta))
        cumulative_prob += conditional_prob/num_candidates
        stopping_probs.append(cumulative_prob)
    return stopping_probs

num_candidates = 100  # Number of candidates

mu = num_candidates/np.e
for beta in [0.1,4,6,8,10]:

    stopping_probs = cumulative_stopping_probability(num_candidates, mu, beta)
    stopping_probs.append(1)
    plt.plot(range(0, num_candidates), stopping_probs, linestyle='dashed',
label=f"β={beta}", alpha=0.6)

plt.axvline(x = 35.5, color = 'black')
plt.legend(loc="upper left")
plt.xlabel('Number of Candidates')
plt.ylabel('Cumulative Stopping Probability')
plt.title('Cumulative Stopping Probability for Secretary Problem')
plt.grid(True)
plt.show()
```



Цей графік відображає накопичувальні ймовірності припинення пошуку для різних значень β. Найнижчий графік, який відповідає β=0.1 максимально наближений до оптимальної стратегії. Це означає, що наречена не буде приймати жодного кандидата допоки не пройде порогове значення. Зі збільшенням дисперсії розподілу порогів (β) можемо стостерігати більш раннє припинення пошуку, причому чим більше β, тим раніше наречена зупиняє пошук.

```
plt.figure(figsize=(12,6))


def calculate_probability_sigma(x, mu, sigma):
    # Assuming sigma follows a normal distribution
    prob = stats.norm.pdf(x, loc=mu, scale=sigma)
    return prob

mu = num_candidates/np.e  # Mean of the normal distribution
sigma = 4  # Standard deviation of the normal distribution

for sigma in [4,6,8]:
    probability =[calculate_probability_sigma(x, mu, sigma) for x in
range(0,100)]
    # print("Pr(σ = {}):".format(x))
    plt.plot(probability, linestyle='dashed', label=f"{sigma}")
plt.axvline(x = num_candidates/np.e, color = 'black')
plt.legend(loc="upper left")

plt.show()
```



Даний графік відображає гіпотетичні розподілу порогів.

```
num_candidates = 100  # Number of candidates
mu = 37  # Mean for the conditional probability
beta = 8
def selection_probability(j, mu, beta):
    return 1 / (1 + math.exp(-(j - mu) / beta))
def calculate_eq(mu, beta, n, j):
    summary = 0
    for r in range(1, j+1):
        # summary += Pr[r]/j
        summary += selection_probability(r, mu, beta)
```

```
    return summary/j

def calc_em(mu, beta, n):

    sum_res = 0
    for j in range(1, n):
        mult = []
        for i in range(1,j+1):
            mult.append(1 - calculate_eq(mu, beta, n, i))
            # print(1 - Q_values[i])
            # print(mult)
        sum_res += np.prod(mult)

    return 1 + sum_res

beta_s = [0.1,1,2,4,8,10,12,16]
mu_s = [25, 29.5, 30, 35]


data=[]
preds = []
for beta in beta_s:
    row=[]
    for mu in mu_s:
        row.append(calc_em(mu, beta, 100))
    data.append(row)


df = pd.DataFrame(data)

print(df)

#define figure and axes
fig, ax = plt.subplots(figsize=(12,6))

#hide the axes
fig.patch.set_visible(False)
ax.axis('off')
ax.axis('tight')
```

Очікувані позиції зупинки наведені в таблиці:

	25	29.5	30	35
0.1	31.025872283353316	36.101436936585095	36.630538535178076	42.18563611028305
1	30.603745797541336	35.68737749220716	36.24894005016547	41.83479621675095
2	29.263298906062914	34.474469798100245	35.04792546712418	40.73536553535694
4	23.777968468236697	29.50913259008033	30.136841995315468	36.30661679248397
8	11.747853421230605	15.827676198554213	16.323316947229774	21.66738580798352
10	8.683469404767214	11.546842489686819	11.90430045532912	15.903479713762586
12	6.86032195816074	8.88094076154593	9.135320407521398	12.027491862836404
16	4.976070634693923	6.0950631584861705	6.235020575668201	7.826424012436975

Результати показують, що використання стохастичної моделі призводить до ранньої зупинки пошуку у порівнянні зі оптимальною стратегією для загального

варіанту задачі. Причина ранньої зупинки за стохастичною моделлю може бути сформульована досить просто. По-перше, існує ненульова ймовірність того, що наречена зупиниться раніше, ніж оптимально; в такому випадку вона не матиме можливості зупинитися вчасно або зупинитися занадто пізно. По-друге, хоча розподіл порогів сам по собі є симетричним, безумовні ймовірності зупинки не є такими. Ймовірність спостерігання відносного рангу $r^j \leq j$ зменшується зі збільшенням j. Розглянемо випадок коли $r^j = 1$. Якщо $j = 1$, то ймовірність зустріти претендента з відносним рангом 1 завжди 1. Для $j = 2$ ця ймовірність складає 0,5. У загальному випадку ця ймовірність буде дорівнювати $1/j$. Таким чином, для заданого значення σ_r, ймовірність зупинки на претенденті j є строго спадною змінною. Тому властивості самої задачі можуть приводити до ранньої зупинки за моделлю.

ВИСНОВКИ

В дипломній роботі на тему "Стохастичний варіант задачі про наречену" було проаналізовано різні варіанти задачі про наречену та їх оптимальні стратегії. Задача стосується галузей теорії оптимальної зупинки випадкових процесів, теорії прийняття рішень та прикладної теорії ймовірностей. Мова йшла про класичний, загальний та стохастичний варіанти задач. Вивчивши загальний варіант задачі про наречену та його оптимальну стратегію, було перейдено до розгляду загального варіанту, який дозволяє краще поглибитися в проблему оптимальної зупинки випадкового процесу. Після цього було детально розглянуто стохастичний варіант, у якому порогові значення при виконанні стратегії вибору задаються не строгими значеннями, а розподілом значень.

Під час дослідження було реалізовано програму з використанням мови програмування python для візуалізації та аналізу отриманих результатів. Зокрема, були представлені результати накопичувальних ймовірностей, що виникають при вирішенні задачі, гіпотетичний розподіл порогових значень та значення очікуваних позицій зупинки пошуку для різних параметрів.

Отримані результати показують, що використання стохастичних моделей призводить до ранньої зупинки пошуку. Ефективність стохастичної моделі вибору в задачі про наречену полягає в тому, що вона більш наближено до реальних умов описує процес прийняття рішення. Відхилення в параметрах моделі можна інтерпретувати як вплив ставлення до ризику та бажання не лише максимізації винагороди, а й мінімізації затрачених ресурсів, що може бути в пріоритеті при прийнятті важливого рішення

Отже, результати цієї дипломної роботи відкривають нові перспективи для дослідження та вдосконалення стратегій та стохастичних моделей в задачі про наречену. Вони можуть бути корисними для дизайну та оптимізації алгоритмів пошуку в різних областях, де виникають подібні задачі. Подальше вдосконалення стохастичного варіанту задачі про наречену може привести до покращення результатів та забезпечити більш ефективні стратегії в пошуку та його оптимальному моменті припинення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Who Solved the Secretary Problem? Thomas S. Ferguson Statistical Science 1989, Vol. 4, No. 3, 282-296
2. Gardner, Martin (1966). "3". New Mathematical Diversions from Scientific American. Simon and Schuster.
3. On Generalized Secretary Problems, J. Neil Bearden and Ryan O. Murphy, The University of Arizona, 2000/04/29
4. BASIC THEORY OF SELECTION BY RELATIVE RANK WITH COST, Sigeiti Moriguti, Journal of the Operations Research Society of Japan, Vol. 36, No. 1, March 1993
5. Secretary problem
6. The Secretary Problem and Its Extensions: A Review, P. R. Freeman, International Statistical Review / Revue Internationale de Statistique, Vol. 51, No. 2 (Aug., 1983)
7. https://changyaochen.github.io/secretary-problem/
8. https://towardsdatascience.com/optimal-stopping-algorithm-with-googles-colab-5b7f9f217e51
9. Sequential Decision Making with Relative Ranks: An Experimental Investigation of the “Secretary Problem”, DARRYL A. SEALE, AND AMNON RAPOPORT, ORGANIZATIONAL BEHAVIOR AND HUMAN DECISION PROCESSES Vol. 69, No. 3, March, pp. 221–236, 1997 ARTICLE NO. OB972683
10. Optimal Stopping Rules Subhash Suri, January 9, 2020
11. A SECRETARY PROBLEM WITH DOUBLE CHOICES, Mitsushi Tamaki, Journal of the Operations Research Society of Japan Vol. 22, No.4,December 1979
12. С. М. Гусейн-Заде РАЗБОРЧИВАЯ НЕВЕСТА Библиотека Математическое просвещение Выпуск 25
13. A Multiple Secretary Problem with Switch Costs by Jiachuan Ding

ДОДАТОК

```
from __future__ import division
import numpy as np
import matplotlib.pyplot as plt
#n=1000


def choose_candidate(n, reject=np.e):
  '''Choose a candidate from a list of n candidates using a specified
strategy.

  reject: percentage of candidates to initially reject (optimal strategy by
default)'''

  candidates = np.arange(1, n+1)
  np.random.shuffle(candidates)

  if reject == np.e:
      stop = int(round(n/reject))
  else:
      stop = int(round(reject*n/100))

  best_from_rejected = np.min(candidates[:stop])
  rest = candidates[stop:]

  try:
    return rest[rest < best_from_rejected][0]
  except IndexError:
    return candidates[-1]

best_candidate = []
for r in range(5, 101, 5):
    sim = np.array([choose_candidate(n=100, reject=r) for i in range(100000)])
    # np.histogram counts frequency of each candidate
    best_candidate.append(np.histogram(sim, bins=100)[0][0]/100000)

plt.figure(figsize=(10, 6))
plt.scatter(range(5, 101, 5), best_candidate)
plt.xlim(0, 100)
plt.xticks(np.arange(0, 101, 10))
plt.ylim(0, 0.4)
plt.xlabel('% of candidates rejected')
plt.ylabel('Probability of choosing best candidate')
plt.grid(True)
plt.axvline(100/np.e, ls='--', c='black')
plt.show()

import random
import math
import numpy as np
```

```
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats

# Перший графік

plt.figure(figsize=(12,6))
# Ймовірність вибору кандидата
def selection_probability(j, mu, beta):
    return 1 / (1 + math.exp(-(j - mu) / beta))

num_candidates = 100
mu = num_candidates/np.e #оптимальний поріг
for beta in [4,6,8]:
    # conditional_probability = selection_probability(num_candidates, mu,
beta)
    # print(f"Conditional probability: {conditional_probability}")
    stopping_prob = [selection_probability(n, mu, beta) for n in range(1,101)]
    plt.plot(stopping_prob, linestyle='dashed', label=f"β={beta}")

plt.axvline(x = 35.5, color = 'black') #оптимальний поріг
plt.legend(loc="upper left")
plt.show()
# Другий графік
plt.figure(figsize=(12,6))

def cumulative_stopping_probability(num_candidates, mu, beta):
    cumulative_prob = 0.0
    stopping_probs = []

    for j in range(1, num_candidates):
        conditional_prob = 1 / (1 + math.exp(-(j - mu) / beta))
        cumulative_prob += conditional_prob/num_candidates
        stopping_probs.append(cumulative_prob)
    return stopping_probs

num_candidates = 100  # Number of candidates

mu = num_candidates/np.e
for beta in [0.1,4,6,8,10]:

    stopping_probs = cumulative_stopping_probability(num_candidates, mu, beta)
    stopping_probs.append(1)
    plt.plot(range(0, num_candidates), stopping_probs, linestyle='dashed',
label=f"β={beta}", alpha=0.6)

plt.axvline(x = 35.5, color = 'black')
plt.legend(loc="upper left")
plt.xlabel('Number of Candidates')
plt.ylabel('Cumulative Stopping Probability')
```

```
plt.title('Cumulative Stopping Probability for Secretary Problem')
plt.grid(True)
plt.show()
import random
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd
import scipy.stats as stats
import random
import math


# Третій графік
plt.figure(figsize=(12,6))


def calculate_probability_sigma(x, mu, sigma):
    # Assuming sigma follows a normal distribution
    prob = stats.norm.pdf(x, loc=mu, scale=sigma)
    return prob

mu = num_candidates/np.e  # Mean of the normal distribution
sigma = 4  # Standard deviation of the normal distribution

for sigma in [4,6,8]:
    probability =[calculate_probability_sigma(x, mu, sigma) for x in
range(0,100)]
    # print("Pr(σ = {}):".format(x))
    plt.plot(probability, linestyle='dashed', label=f"{sigma}")
plt.axvline(x = num_candidates/np.e, color = 'black')
plt.legend(loc="upper left")

plt.show()


# plt.figure(figsize=(12,6))

num_candidates = 100  # Number of candidates
mu = 37  # Mean for the conditional probability
beta = 8


def selection_probability(j, mu, beta):
    return 1 / (1 + math.exp(-(j - mu) / beta))


def calculate_eq(mu, beta, n, j):
    summary = 0
    for r in range(1, j+1):
        # summary += Pr[r]/j
        summary += selection_probability(r, mu, beta)
```

```
    return summary/j

def calc_em(mu, beta, n):

    sum_res = 0
    for j in range(1, n):
        mult = []
        for i in range(1,j+1):
            mult.append(1 - calculate_eq(mu, beta, n, i))
            # print(1 - Q_values[i])
            # print(mult)
        sum_res += np.prod(mult)

    return 1 + sum_res

beta_s = [0.1,1,2,4,8,10,12,16]
mu_s = [25, 29.5, 30, 35]


data=[]
preds = []
for beta in beta_s:
    row=[]
    for mu in mu_s:
        row.append(calc_em(mu, beta, 100))
    data.append(row)


df = pd.DataFrame(data)

print(df)

#define figure and axes
fig, ax = plt.subplots(figsize=(12,6))

#hide the axes
fig.patch.set_visible(False)
ax.axis('off')
ax.axis('tight')


#create table
table = ax.table(cellText=df.values, rowLabels=beta_s, colLabels=mu_s,
loc='center')

#display table
fig.tight_layout()
plt.show()
```