Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Системы искусственного интеллекта**

**Лабораторная работа №2**

**«Нечеткие множества»**

**Вариант №14**

Выполнил

студент группы ИВТАПбд-41

Масловский Егор Артемович

Проверил

Преподаватель кафедры ВТ

Хайруллин И.Д.

Ульяновск

2025

**Цель работы:**

Необходимо на языке Python разработать скрипт, позволяющий выполнить операцию пересечения заданных пользователем нечетких множеств с трапециевидными функциями принадлежности. Входными данными будут параметры функций принадлежности и четкие объекты для каждого из множеств. Выходными – пересечение данных нечетких множеств.

Предметная область: Анализ социальных сетей

* Активность пользователей: низкая, средняя, высокая
* Вовлеченность (engagement): неактивная, малоактивная, активная, сильно активная

**Контрольные вопросы по лабораторной работе:**

1. Дайте определение нечеткому множеству.
2. Какие способы задания функций принадлежности вы знаете?
3. Какую операцию вы реализовали в своей лабораторной работе?

**Теоретический материал:**

Объекты – лингвистические переменные. Таким образом, операндами и результатом интеллектуальных операций являются значения особого вида – *нечеткие множества.*

*Нечёткие множества (fuzzy sets) в машинном обучении (ML) — это наборы элементов с разной степенью принадлежности.* В отличие от традиционных множеств с чёткими границами, нечёткие множества позволяют элементам принадлежать к группе в разной степени. Каждый элемент имеет значение принадлежности от 0 до 1, указывающее на его степень принадлежности к набору.

Основные интеллектуальные операции строятся с помощью *операций нечеткой логики.*

Алгоритмы вычисления нечетких значений предназначены для манипулирования со значениями, представленными нечеткими множествами на основе операций нечеткой логики, поэтому они классифицируются как нечеткие системы логического вывода. Часто используют сокращенную форму обозначенного класса моделей – ***нечеткие модели или нечеткие системы.***

Некоторые особенности теории:

* **Функции принадлежности** — определяют, каким образом каждому элементу нечёткого множества присваивается значение принадлежности. Распространённые типы — треугольные, трапециевидные и гауссовы функции.
* **Лингвистические переменные** — переменные, описываемые естественным языком, такие как «высокий», «средний» или «низкий».

**Способы задания функции принадлежности:**

* Структурный способ
* Функциональный способ

**Пересечение нечетких множеств** A и B определяется как операция, результатом которой является новое нечеткое множество C = A ∩ B с функцией принадлежности:



Эта операция соответствует логической операции "И" и находит минимальную степень принадлежности для каждого элемента.

**Описание реализации:**

Архитектура программы включает несколько ключевых компонентов: модуль математических вычислений для работы с нечеткими множествами, модуль взаимодействия с пользователем и модуль визуализации результатов.

Программа начинается с инициализации необходимых структур данных и параметров, затем переходит к взаимодействию с пользователем, после чего выполняет вычисления и завершается визуализацией результатов.

Для работы с трапециевидными функциями принадлежности была разработана специализированная функция trapezoidal\_mf(), которая принимает четыре параметра, определяющих форму трапеции. Особое внимание было уделено обработке граничных условий и предотвращению ошибок деления на ноль. В коде добавлена малая константа 1e-6 к знаменателям, что обеспечивает численную устойчивость при вычислениях.

Листинг 1. Трапециевидная функция принадлежности

|  |
| --- |
| def trapezoidal\_mf(x, a, b, c, d):result = np.zeros\_like(x)   # Условия для трапециевидной функции  condition1 = (x >= a) & (x < b)  condition2 = (x >= b) & (x <= c)  condition3 = (x > c) & (x <= d)   # Вычисляем значения для каждой части  result[condition1] = (x[condition1] - a) / (b - a) if b != a else 1.0  result[condition2] = 1.0  result[condition3] = (d - x[condition3]) / (d - c) if d != c else 1.0   return result |

Были определены два универсума: для активности (0-50) и вовлеченности (0-20). Для каждого универсума создано по три и четыре нечетких множества с перекрывающимися зонами, что позволяет обеспечить плавные переходы между лингвистическими категориями.

Операция пересечения реализована через функцию fuzzy\_intersection(), которая использует поэлементный минимум между двумя массивами значений принадлежности. Для работы с конкретными значениями, вводимыми пользователем, применяется встроенная функция min() для пар значений принадлежности.

Листинг 2. Пересечение множеств

|  |
| --- |
| def fuzzy\_intersection(set1, set2):  return np.minimum(set1, set2) |

Интерфейс вывода результатов организован в виде структурированных таблиц, где последовательно отображаются степени принадлежности для каждого нечеткого множества и результаты операции пересечения для всех комбинаций.

Модуль визуализации создает комплексное представление результатов через систему из двух графиков. На двух графиках отображаются трапециевидные функции принадлежности для активности и вовлеченности с отметками введенных пользователем значений.

Вычислительный модуль программы использует интерполяцию для определения степеней принадлежности конкретных значений. Для каждого введенного пользователем значения программа находит соответствующие степени принадлежности ко всем определенным нечетким множествам.

Операция пересечения выполняется как для визуализации на универсумах, так и для конкретных пар значений. Это позволяет одновременно получить как общее представление о взаимосвязи множеств, так и конкретные числовые результаты для анализируемого случая.

**Тестирование:**

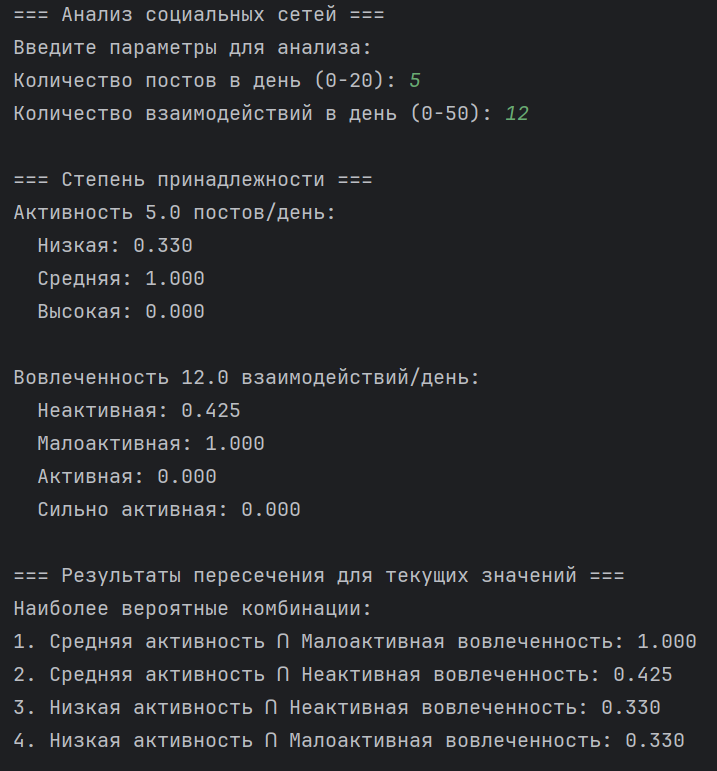


Рис. 1. Результат 1 теста в консоли

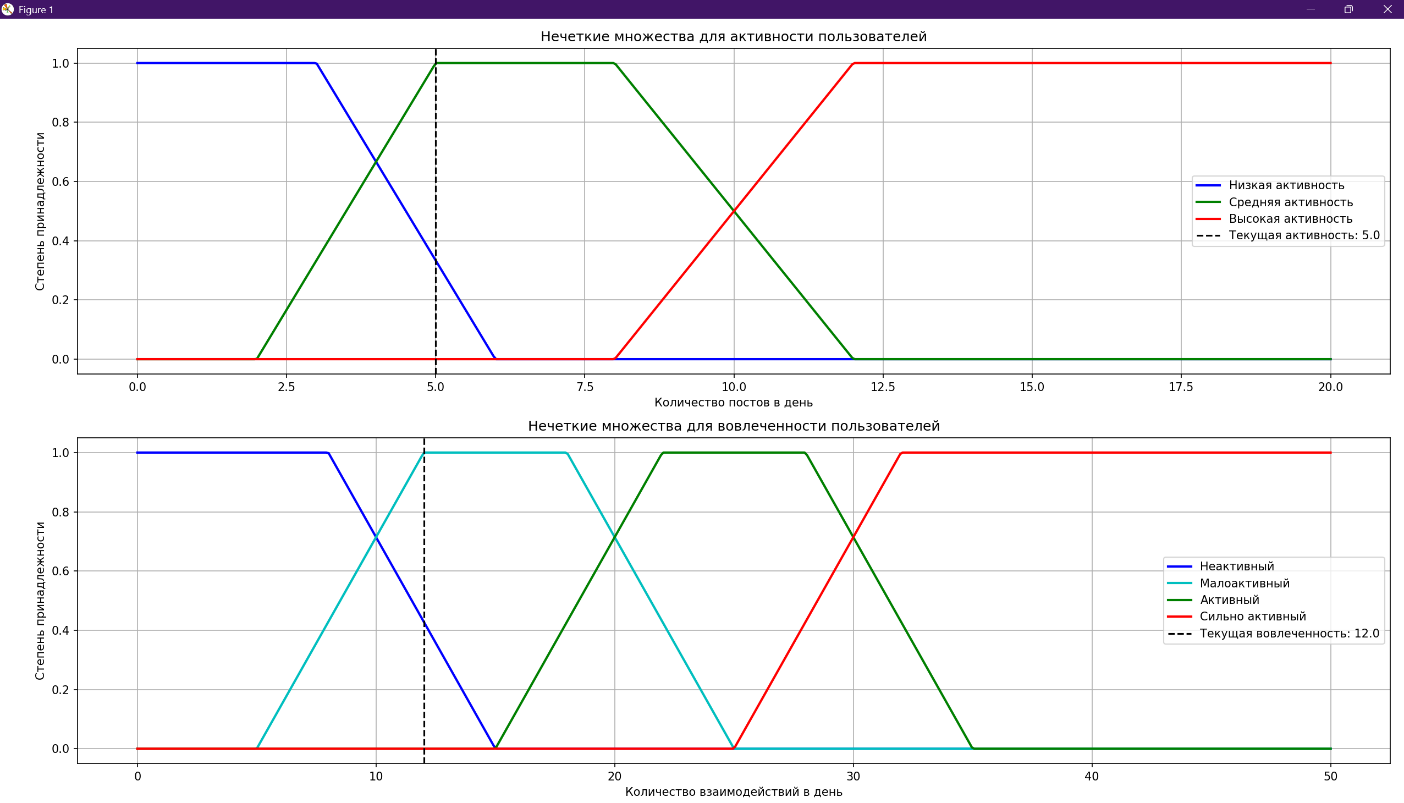


Рис. 2. Графики 1 теста

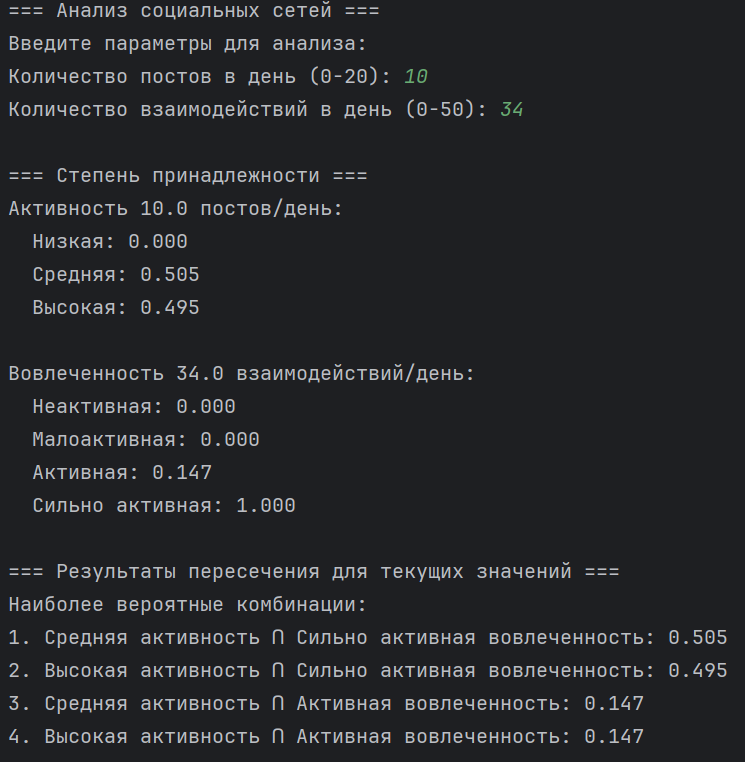


Рис. 3. Результат 2 теста в консоли

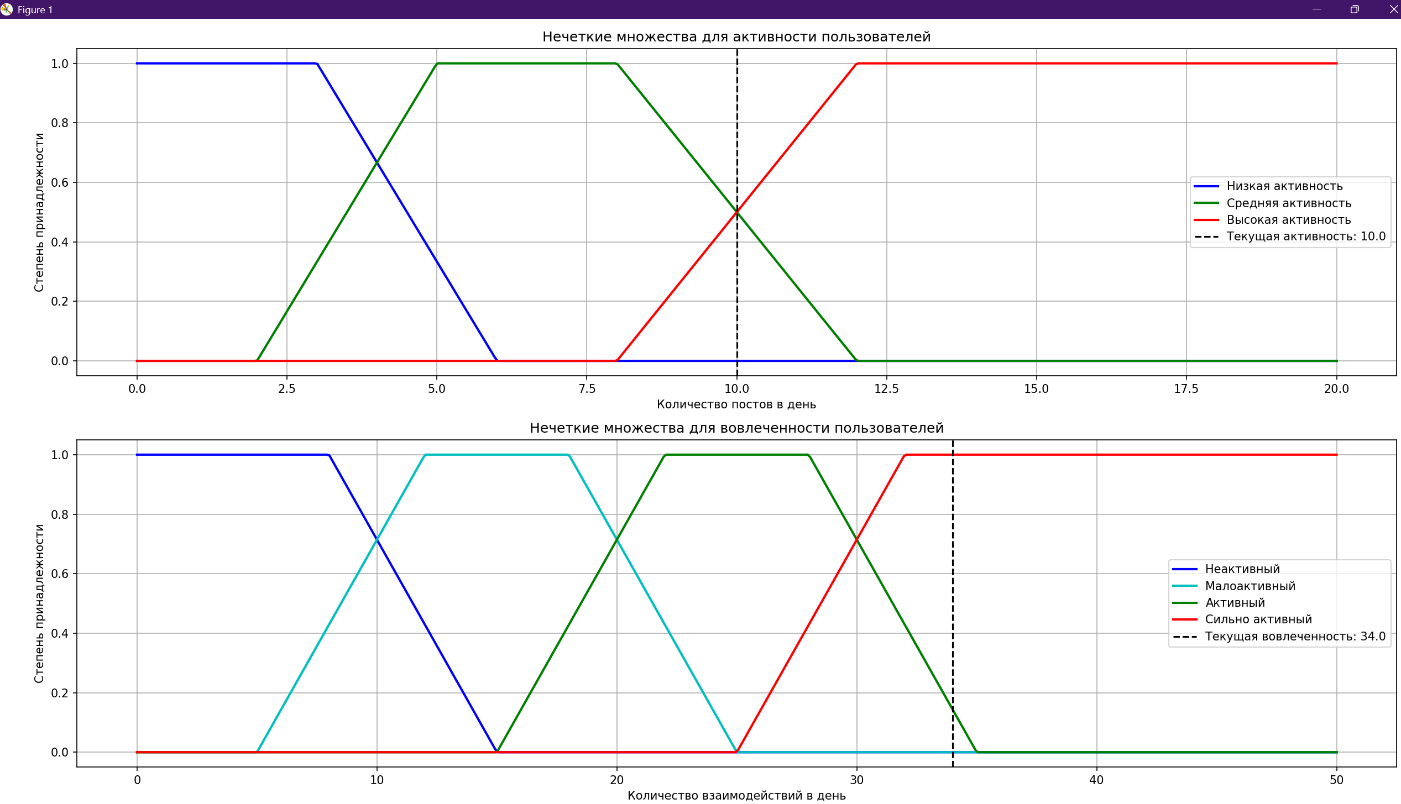


Рис. 4. Графики 2 теста

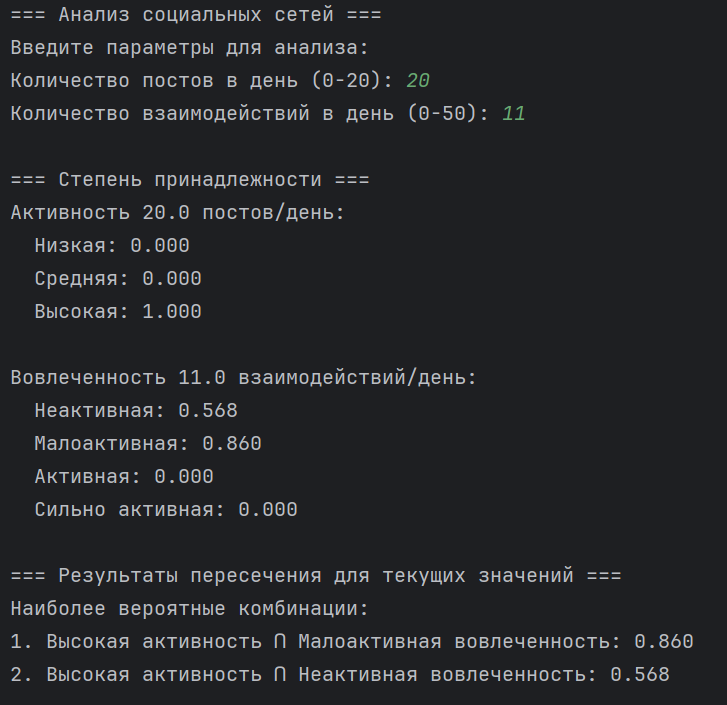


Рис. 5. Результат 3 теста в консоли

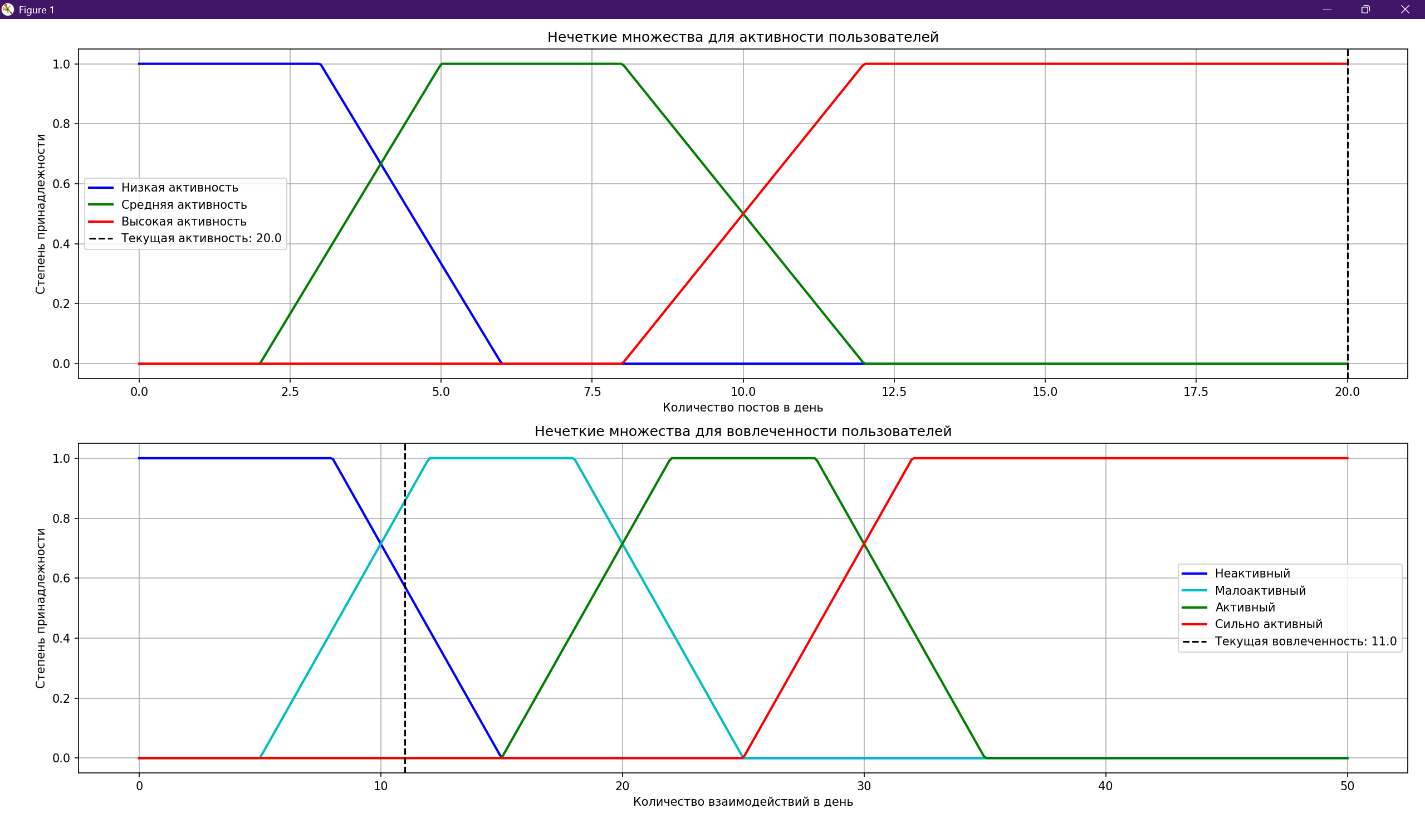


Рис. 6. Графики 3 теста

**Контрольные вопросы:**

*1. Дайте определение нечеткому множеству*

**Нечеткое множество** — это расширение понятия классического множества, в котором элементы могут иметь **степень принадлежности** к множеству, принимающую значения из интервала [0,1], а не только бинарные значения 0 или 1.

Формально, нечеткое множество A на универсуме X определяется как множество упорядоченных пар:

A = {(x, μₐ(x)) | x ∈ X}

где:

* X - универсум (область определения)
* μₐ: X → [0,1] - функция принадлежности
* μₐ(x) - степень принадлежности элемента x к нечеткому множеству A

**Отличия от классического множества:**

* В классической теории: элемент либо принадлежит (1), либо не принадлежит (0) множеству
* В нечеткой логике: элемент может принадлежать множеству частично (0.3, 0.7 и т.д.)

*2. Какие способы задания функций принадлежности вы знаете?*

Существует несколько основных способов задания функций принадлежности:

* 1. Параметрические функции
     + Треугольная: задается тремя параметрами (a, b, c)
     + Трапециевидная: задается четырьмя параметрами (a, b, c, d)
     + Гауссова
     + Сигмоидальная
  2. Непараметрические методы
     + Статистические: на основе гистограмм и распределений
     + Экспертные оценки: опрос специалистов в предметной области
     + Адаптивные алгоритмы: нейронные сети, генетические алгоритмы
  3. Эвристические методы
     + Относительные предпочтения: попарные сравнения элементов
     + Интервальные оценки: задание функций через α-срезы
  4. Аналитические методы
     + На основе регрессионного анализа
     + Метод главных компонент

*3. Какую операцию вы реализовали в своей лабораторной работе?*

В лабораторной работе была реализована **операция пересечения нечетких множеств**.

Для двух нечетких множеств A и B с функциями принадлежности μₐ(x) и μ\_B(x) пересечение C = A ∩ B определяется как:

μ\_C(x) = min(μₐ(x), μ\_B(x)) для всех x ∈ X

Было выполнено пересечение множеств:

* **Активность** = { низкая, средняя, высокая }
* **Вовлеченность** = { неактивная, малоактивная, активная, сильно активная }

Результат: 12 комбинаций

Операция пересечения соответствует логической операции **"И"**:

* Результирующая степень принадлежности показывает **наихудший** (минимальный) сценарий
* Обеспечивает консервативную оценку

Операция пересечения была выбрана как наиболее соответствующая требованиям варианта задания и обеспечивающая надежную оценку.

**Вывод:**

В ходе выполнения лабораторной работы была успешно разработана и реализована программа для операции пересечения нечетких множеств с трапециевидными функциями принадлежности в предметной области анализа социальных сетей. Программа позволяет анализировать взаимосвязь между кредитным рейтингом и уровнем риска, вычисляя степени принадлежности для различных лингвистических категорий и определяя их пересечение через операцию минимума.

**Приложение: код программы**

|  |
| --- |
| import numpy as np import matplotlib.pyplot as plt   # Функция для определения трапециевидной функции принадлежности def trapezoidal\_mf(x, a, b, c, d):  *"""  Трапециевидная функция принадлежности.  :param x: Точки, для которых вычисляется функция принадлежности.  :param a: Левая граница начала возрастания функции.  :param b: Левая верхняя граница (где функция равна 1).  :param c: Правая верхняя граница (где функция равна 1).  :param d: Правая граница окончания убывания функции.  :return: Значение функции принадлежности в точках x.  """* result = np.zeros\_like(x)   # Условия для трапециевидной функции  condition1 = (x >= a) & (x < b)  condition2 = (x >= b) & (x <= c)  condition3 = (x > c) & (x <= d)   # Вычисляем значения для каждой части  result[condition1] = (x[condition1] - a) / (b - a) if b != a else 1.0  result[condition2] = 1.0  result[condition3] = (d - x[condition3]) / (d - c) if d != c else 1.0   return result   # Операция пересечения нечетких множеств (минимум) def fuzzy\_intersection(set1, set2):  return np.minimum(set1, set2)   # Универсум для активности пользователей (посты в день) x\_activity = np.linspace(0, 20, 500)  # Универсум для вовлеченности (лайки, комментарии, репосты в день) x\_engagement = np.linspace(0, 50, 500)  # Определение трапециевидных функций для активности пользователей # Исправленные параметры для лучшего покрытия low\_activity = trapezoidal\_mf(x\_activity, 0, 0, 3, 6) medium\_activity = trapezoidal\_mf(x\_activity, 2, 5, 8, 12) high\_activity = trapezoidal\_mf(x\_activity, 8, 12, 20, 20)  # Определение трапециевидных функций для вовлеченности # Исправленные параметры для лучшего покрытия inactive = trapezoidal\_mf(x\_engagement, 0, 0, 8, 15) low\_engagement = trapezoidal\_mf(x\_engagement, 5, 12, 18, 25) active = trapezoidal\_mf(x\_engagement, 15, 22, 28, 35) highly\_active = trapezoidal\_mf(x\_engagement, 25, 32, 50, 50)  # Ввод данных от пользователя print("=== Анализ социальных сетей ===") print("Введите параметры для анализа:")  try:  # Активность пользователя  activity\_value = float(input("Количество постов в день (0-20): "))  engagement\_value = float(input("Количество взаимодействий в день (0-50): "))   # Проверка на выход за границы универсума  if activity\_value < 0 or activity\_value > 20:  print("Ошибка: активность должна быть в диапазоне 0-20")  exit()  if engagement\_value < 0 or engagement\_value > 50:  print("Ошибка: вовлеченность должна быть в диапазоне 0-50")  exit()  except ValueError:  print("Ошибка: введите числовые значения")  exit()   # Функция для вычисления принадлежности для конкретного значения def calculate\_membership(value, x\_universe, mf):  # Находим ближайший индекс в универсуме  idx = np.argmin(np.abs(x\_universe - value))  return mf[idx]   # Вычисление степени принадлежности для активности activity\_membership = [  calculate\_membership(activity\_value, x\_activity, low\_activity),  calculate\_membership(activity\_value, x\_activity, medium\_activity),  calculate\_membership(activity\_value, x\_activity, high\_activity) ]  # Вычисление степени принадлежности для вовлеченности engagement\_membership = [  calculate\_membership(engagement\_value, x\_engagement, inactive),  calculate\_membership(engagement\_value, x\_engagement, low\_engagement),  calculate\_membership(engagement\_value, x\_engagement, active),  calculate\_membership(engagement\_value, x\_engagement, highly\_active) ]  # Вывод результатов принадлежности print("\n=== Степень принадлежности ===") print(f"Активность {activity\_value} постов/день:") print(f" Низкая: {activity\_membership[0]:.3f}") print(f" Средняя: {activity\_membership[1]:.3f}") print(f" Высокая: {activity\_membership[2]:.3f}")  print(f"\nВовлеченность {engagement\_value} взаимодействий/день:") print(f" Неактивная: {engagement\_membership[0]:.3f}") print(f" Малоактивная: {engagement\_membership[1]:.3f}") print(f" Активная: {engagement\_membership[2]:.3f}") print(f" Сильно активная: {engagement\_membership[3]:.3f}") # Визуализация результатов fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 10))  # График 1: Функции принадлежности для активности ax1.plot(x\_activity, low\_activity, 'b-', label='Низкая активность', linewidth=2) ax1.plot(x\_activity, medium\_activity, 'g-', label='Средняя активность', linewidth=2) ax1.plot(x\_activity, high\_activity, 'r-', label='Высокая активность', linewidth=2) ax1.axvline(x=activity\_value, color='k', linestyle='--', label=f'Текущая активность: {activity\_value}') ax1.set\_title('Нечеткие множества для активности пользователей') ax1.set\_xlabel('Количество постов в день') ax1.set\_ylabel('Степень принадлежности') ax1.legend() ax1.grid(True)  # График 2: Функции принадлежности для вовлеченности ax2.plot(x\_engagement, inactive, 'b-', label='Неактивный', linewidth=2) ax2.plot(x\_engagement, low\_engagement, 'c-', label='Малоактивный', linewidth=2) ax2.plot(x\_engagement, active, 'g-', label='Активный', linewidth=2) ax2.plot(x\_engagement, highly\_active, 'r-', label='Сильно активный', linewidth=2) ax2.axvline(x=engagement\_value, color='k', linestyle='--', label=f'Текущая вовлеченность: {engagement\_value}') ax2.set\_title('Нечеткие множества для вовлеченности пользователей') ax2.set\_xlabel('Количество взаимодействий в день') ax2.set\_ylabel('Степень принадлежности') ax2.legend() ax2.grid(True)  plt.tight\_layout() plt.show()  # Вывод результатов пересечения для текущих значений print("\n=== Результаты пересечения для текущих значений ===") current\_intersections = []  activity\_labels = ["Низкая", "Средняя", "Высокая"] engagement\_labels = ["Неактивная", "Малоактивная", "Активная", "Сильно активная"]  for i, activity\_mf in enumerate(activity\_membership):  for j, engagement\_mf in enumerate(engagement\_membership):  intersection\_value = min(activity\_mf, engagement\_mf)  current\_intersections.append((intersection\_value, i, j))  # Сортировка по убыванию степени принадлежности current\_intersections.sort(reverse=True, key=lambda x: x[0])  print("Наиболее вероятные комбинации:") found\_valid = False for i, (value, act\_idx, eng\_idx) in enumerate(current\_intersections):  if value > 0.01: # Небольшой порог для учета числовых погрешностей  print(f"{i+1}. {activity\_labels[act\_idx]} активность ∩ {engagement\_labels[eng\_idx]} вовлеченность: {value:.3f}")  found\_valid = True  if not found\_valid:  print("Нет значимых пересечений. Попробуйте другие значения.")  print("Рекомендуемые тестовые значения:")  print(" Активность: 2, 6, 10 постов/день")  print(" Вовлеченность: 10, 20, 30 взаимодействий/день") |