МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»
РТУ МИРЭА

Институт Информационных Технологий

Кафедра Вычислительной Техники

ПРАКТИЧЕСКАЯ РАБОТА №1

по дисциплине
«Проектирование интеллектуальных систем (часть 2/2)»

Студент группы:ИКБО-04-22 __Кликушин В.И._
(Ф. И.О. студента)

Преподаватель __Холмогоров В.В.__
(Ф.И.О. преподавателя)

Москва 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современные интеллектуальные системы все чаще сталкиваются с необходимостью обработки неточных, частично определенных или качественных данных. Во многих предметных областях, включая экономику, технические системы управления, медицину и анализ рисков, значения параметров не могут быть однозначно классифицированы. В таких ситуациях применение классической булевой логики оказывается недостаточным, поскольку она требует строгого разграничения между истинностью и ложностью.

Теория нечетких множеств, предложенная Лотфи Заде, предоставляет математический аппарат для формального описания размытых понятий и построения моделей на основе нечетких отношений.

Одним из ключевых преимуществ нечеткой логики является возможность моделировать рассуждения человека, опирающегося на приблизительные оценки, качественные характеристики и лингвистические переменные. Такой подход широко используется в экспертных системах, алгоритмах управления и аналитических моделях, где точные количественные зависимости установить невозможно.

В данной работе рассматривается применение методов нечетких множеств для анализа рынка виртуальных предметов игры Counter-Strike 2. Стоимость, ликвидность и инвестиционная привлекательность игровых скинов формируются под воздействием множества факторов, обладающих размытыми границами и качественными характеристиками.

1 ПОСТАНОВКА ЗАДАЧИ

Цель работы: приобрести навыки проектирования и реализации систем, основанных на нечетких отношениях.

Задачи: в соответствии с выбранной темой реализовать приложение, принимающее на вход N-ое количество нечетких множеств, реализовать в системе минимальный набор следующих операций с нечеткими множествами: построение продукционных отношений в виде матриц (которые в теории нечетких множеств формируются на основе операции пересечения множеств), построение на основе полученных матриц графиков, свертка матриц для получения новых правил на основе уже имеющихся.

2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Понятие нечеткого множества – эта попытка математической формализации нечеткой информации для построения математических моделей. В основе этого понятия лежит представление о том, что составляющие данное множество элементы, обладающие общим свойством, могут обладать этим свойством в различной степени и, следовательно принадлежать к данному множеству с различной степенью. При таком подходе высказывания типа «такой-то элемент принадлежит данному множеству» теряют смысл, поскольку необходимо указать «насколько сильно» или с какой степенью конкретный элемент удовлетворяет свойствам данного множества.

Популярность подхода, основанного на формализации нечеткостей, свидетельствует о многочисленных областях его практического применения, что способствовало формированию специального направления в области искусственного интеллекта — исследованию нечетких систем. Математическая теория нечетких множеств, предложенная Лотфи Заде, позволяет описывать нечеткие понятия и знания, оперировать этими знаниями и делать нечеткие выводы.

2.1 Нечеткие множества

Нечетким множеством C в X называется совокупность пар вида $(x, \mu_c(x))$, где $x \in C$, а $\mu_c(x)$ – функция принадлежности, определенная на интервале [0, 1].

Функция принадлежности $\mu_c(x)$ полностью характеризует x (она еще называется характеристической функцией). Поэтому справедливо утверждение: нечеткое множество вполне описывается своей функцией принадлежности.

Обычные (четкие) множества составляют собой подкласс нечетких множеств. Действительно, функцией принадлежности обычного множества $B \subset X$ является его характеристическая функция, представленная Формулой 2.1.

$$\mu_B(x) = \begin{cases} 1, \text{если } x \in B \\ 0, \text{если } x \notin B \end{cases} \quad (2.1)$$

Таким образом, нечеткое множество представляет собой более широкое понятие, чем обычное множество, а функция принадлежности нечеткого множества может быть произвольной.

Каждому нечеткому множеству соответствует свое множество функций принадлежности $\mu_c(x)$.

Функцией принадлежности называется функция, которая позволяет вычислить степень принадлежности произвольного элемента универсального множества к нечеткому множеству.

Пусть A и B – нечеткие множества в X, а $\mu_A(x)$ и $\mu_B(x)$ – их четкие функции принадлежности соответственно. Справедливо утверждение, что A включает в себя B (т.е. $B \subseteq A$ нестрогое подмножество), если для любого $x \in X$ выполнено неравенство $\mu_B(x) \leq \mu_A(x)$.

Лингвистической переменной называется переменная, значениями которой могут быть слова или словосочетания некоторого естественного или искусственного языка.

Терм–множеством называется множество всех возможных значений лингвистической переменной.

Термом называется любой элемент терм–множества. В теории нечетких множеств терм формализуется нечетким множеством с помощью функции принадлежности.

2.2 Свойства нечетких множеств

Высотой нечеткого множества A называется верхняя граница его функции принадлежности: $hgt(A) = sup\mu_A(x)$ (супремум – верхняя граница, hgt сокращенное height – верх). Для дискретного универсального множества

супремум становится максимумом, а значит высотой нечеткого множества будет максимум степеней принадлежности его элементов.

Нечеткое множество A называется нормальным, если выполнено равенство $sup\mu_A(x) = 1; x \in X$. В противном случае нечеткое множество называется субнормальным.

Носителем нечеткого множества A называется четкое подмножество универсального множества X, элементы которого имеют ненулевые степени принадлежности $sup\,\mathrm{A} = \{x \,/\, x \in X, \mu_A(x) > 0\}$. Другими словами, sup A определяет верхнюю границу A при положительных значениях μ.

Нечеткое множество называется пустым, если его носитель является пустым множеством.

Ядром нечеткого множества A называется четкое подмножество универсального множества X, элементы которого имеют степени принадлежности равные единице: $core(A) = \{x \,/\, x \in X, \mu_A(x) = 1\}$. Ядро субнормального нечеткого множества пустое.

α-сечением (или множеством α-уровня) нечеткого множества A называется четкое подмножество универсального множества X, элементы которого имеют степени принадлежности большие или равные α: $\mathrm{A}\alpha = \{x \,/\, x \in X, \mu_A(x) \geq \alpha\}, \alpha \in [0,1]$. Значение α называют α-уровнем. Носитель (ядро) можно рассматривать как сечение нечеткого множества на нулевом (единичном) α-уровне.

На Рисунке 2.1 продемонстрированы понятия носителя, ядра и α-сечения.

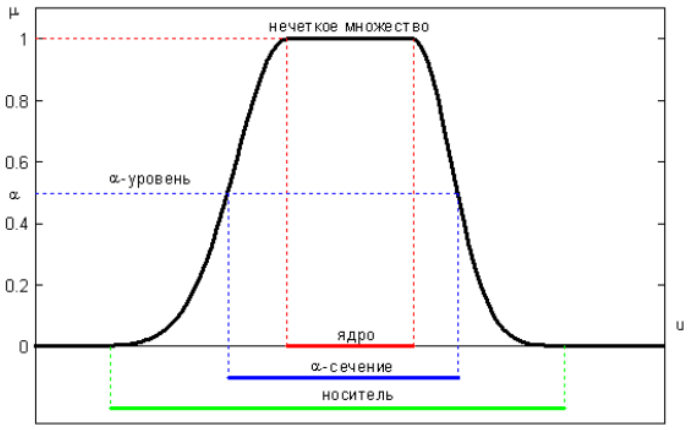

Рисунок 2.1 – Носитель, ядро и α-сечение

Нечеткое множество будет выпуклым, если все его α-сечением - выпуклые множества.

2.3 Операции над нечеткими множествами

Операции над нечеткими множествами можно определить различными способами. Выбор конкретного из них (объединение, пересечение и пр.) зависит от смысла решаемой задачи. Следует отметить, что класс нечетких множеств охватывает и множества в обычном смысле. Поэтому вводимые определения должны соответствовать обычным операциям, принятым в теории множеств.

Объединением нечетких множеств A и B в X называется нечеткое множество $A \cup B$ с функцией принадлежности вида $\mu_{A\cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}, x\epsilon X$.

Пересечением нечетких множеств A и B в X называется нечеткое множество $A \cap B$ с функцией принадлежности вида $\mu_{A\cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}, x\epsilon X$.

Два нечетких множества A и B в X равны $(A = B)$ тогда и только тогда, когда выполняется равенство $\mu_A(x) = \mu_B(x); \forall x\epsilon X$.

Дополнением нечеткого множества A в X называется нечеткое множество A' с функцией принадлежности вида $\mu_{A'}(x) = 1 - \mu_A(x), x\epsilon X$.

Операция перемещения изменяет положение функции принадлежности на величину λ. При $\lambda > 0$ происходит перемещение вправо, а при $\lambda < 0$ влево. Выражение для функции принадлежности: $\mu_B(x) = \mu_A(x - \lambda), \lambda\epsilon R, \forall x\epsilon X$.

Операция нормализации осуществляется в соответствии с Формулой 2.2.

$$\mu_B(x) = \frac{\mu_A(x)}{\max(\mu_A(x))}, \forall x\epsilon X \tag{2.2}$$

2.4 Отношения и операции над ними

Очевидно, что все отношения между объектами некоторого множества можно, в конечном счете, свести к отношению пар этих объектов при том, что каждый из них может одновременно иметь разные отношения с некоторыми другими.

Отношение на множествах X и Y — это подмножество их декартова произведения $X \times Y$. Элемент $(x, y) \in X \times Y$ принадлежит отношению $R \subseteq X \times Y$, если между объектами x и y существует заданное бинарное соответствие. Формально отношение можно представить как характеристическую функцию (Формула 2.3).

$$\mu_R(x, y) = \begin{cases} 1, \text{если } (x, y) \in B \\ 0, \text{если } (x, y) \notin B \end{cases} \quad (2.3)$$

Если $X = Y$, отношение называется бинарным отношением на множестве X.

Для анализа и обработки отношений широко используется матричное представление. Если множества конечны, $X = \{x_1, \dots, x_m\}$, $Y = \{y_1, \dots, y_n\}$, то отношение R задается матрицей (Формула 2.4).

$$R = [r_{ij}], r_{ij} = \mu_R(x_i, y_j) \in \{0,1\} \quad (2.4)$$

2.5 Нечеткие отношения

Характерная черта четких отношений – их определенность: либо есть отношение, либо его нет. Среди множества всяких отношений есть такие, которые не могут быть нечеткими.

Нечеткое отношение R между множествами X и Y— это обобщение обычного отношения, в котором каждой паре элементов (x, y) ставится в соответствие степень принадлежности (Формула 2.5).

$$\mu_R(x, y) \in [0,1] \tag{2.5}$$

Нечеткое отношение можно рассматривать как функцию: $R: X \times Y \to [0,1]$.

В матричном виде нечеткое отношение задается матрицей значений принадлежности (Формула 2.4).

В логике нечетких отношений важное место занимает представление продукционных правил вида «если U, то V». Такие правила связывают два лингвистических понятия, описываемых нечеткими множествами U и V, заданными на соответствующих универсальных множествах. В нечеткой логике продукция «если U, то V» интерпретируется как нечеткое отношение, определяемое на декартовом произведении этих универсумов.

Пусть U — нечеткое множество на X, заданное функцией принадлежности $\mu_U(x)$, а V — нечеткое множество на Y, заданное функцией $\mu_V(y)$. Тогда нечеткое отношение $R \subseteq X \times Y$, моделирующее правило «ЕСЛИ U ТО V» обычно определяется произведением нечетких множеств $U \times V$ (Формула 2.6).

$$\mu_R(x, y) = \min(\mu_U(x), \mu_V(y)) \tag{2.6}$$

Таким образом, отношение $R = U \times V$ задает степень истинности утверждения «объект x соответствует объекту y в соответствии с правилом «если U, то V».

2.6 Композиция нечетких отношений

Композиция (или свертка) нечетких отношений — ключевая операция, позволяющая строить цепочки нечетких соответствий и реализовывать механизм нечеткого логического вывода на основе правил вида «если U, то V».

Пусть заданы два нечетких отношения: $R \subseteq X \times Y$, заданное функцией принадлежности $\mu_R(x, y) \in [0,1]$ и $S \subseteq Y \times Z$, заданное функцией $\mu_S(y, z) \in [0,1]$.

Эти отношения моделируют два звена цепочки нечетких соответствий: $X \xrightarrow{R} Y \xrightarrow{S} Z$. Тогда их композиция $T = R \circ S$ является нечетким отношением на $X \times Z$, таким что $X \xrightarrow{T} Z$.

Композиция нечетких отношений определяется по Формуле 2.7.

$$\mu_T(x,z) = max\min\left(\mu_R(x,y), \mu_S(y,z)\right) \tag{2.7}$$

2.7 Нечеткий вывод

Нечеткие выводы чаще всего основаны на правиле заключения (modus ponens). Это правило касается импликации и говорит о том, что если вся импликация истинна и посылка истинна, то и заключение истинно (Формула 2.8).

$$\frac{p, p \to q}{q} \tag{2.8}$$

В числителе – высказывания, истинность которых уже доказана, а в знаменателе – высказывания, истинность которых логически следует из верхних высказываний.

3 ПРАКТИЧЕСКАЯ ЧАСТЬ

3.1 Предметная область

Counter-Strike 2 (CS2) — это многопользовательский тактический шутер, в котором существует сложная экономическая система виртуальных предметов (скинов). Скины — это визуальные оформления оружия, персонажей и аксессуаров, которые не влияют на игровой процесс, но имеют значительную ценность для сообщества игроков.

Рынок виртуальных предметов игры Counter-Strike 2 представляет собой динамичную экономическую экосистему, в которой стоимость, ликвидность и инвестиционная привлекательность отдельных скинов формируются под воздействием широкого набора факторов. Рынок включает как предметы массового спроса, так и редкие коллекционные позиции, стоимость которых может варьироваться от нескольких копеек до десятков тысяч долларов.

Каждый скин в CS2 обладает набором характеристик, непосредственно влияющих на его ценность, спрос и поведение на торговой площадке. К числу наиболее значимых относятся следующие параметры:

1. Состояние износа (Float Value). Параметр float value представляет собой вещественное число в диапазоне от 0 до 1, характеризующее степень визуального износа скина.
2. Шаблон раскраски (Paint Seed). Параметр paint seed определяет вариант текстурного паттерна, который получает скин при генерации. Значения находятся в диапазоне от 0 до 1000.
3. Редкость (Rarity) является одним из ключевых качественных параметров скина. Игра использует иерархическую шкалу, включающую следующие уровни: ширпотреб, промышленное качество, армейское качество, запрещенное, засекреченное, тайное. Редкость оказывает значительное влияние на рыночную цену, так как более редкие предметы встречаются реже в дропах и обладают

большим спросом среди коллекционеров.

3.2 Определение нечетких множеств

Для анализа рынка виртуальных предметов CS2 в рамках интеллектуальной системы оценки и прогнозирования стоимости скинов предлагается использование метода нечеткой логики. Основная идея заключается в том, что многие параметры скинов обладают размытыми границами и не могут быть точно классифицированы. Для каждого из ключевых признаков формируются нечеткие множества, определяющие степень принадлежности конкретного скина к соответствующим категориям.

3.2.1 Степень износа

Терм-множество для данной лингвистической переменной: {«Прямо с завода», «Немного поношенный», «После полевых испытаний», «Поношенный», «Закаленный в боях»}. Для каждого терма определена своя функция принадлежности:

1. «Прямо с завода».

$$\mu_{FN}(x) = \begin{cases} 1, \text{если } 0 \le x < 0.01 \\ \dfrac{0.07 - x}{0.06}, \text{если } 0.01 \le x < 0.07 \\ 0, \text{если } x > 0.07 \end{cases}$$

2. «Немного поношенный».

$$\mu_{MW}(x) = \begin{cases} 0, \text{если } x \le 0.06 \\ \dfrac{x - 0.6}{0.02}, \text{если } 0.06 < x \le 0.08 \\ 1, \text{если } 0.08 \le x \le 0.10 \\ \dfrac{0.15 - x}{0.05}, \text{если } 0.10 < x \le 0.15 \\ 0, \text{если } x > 0.15 \end{cases}$$

3. «После полевых испытаний».

$$\mu_{FT}(x) = \begin{cases} 0, \text{если } x \leq 0.12 \\ \dfrac{x - 0.12}{0.03}, \text{если } 0.12 < x \leq 0.15 \\ 1, \text{если } 0.15 \leq x \leq 0.30 \\ \dfrac{0.38 - x}{0.08}, \text{если } 0.30 < x \leq 0.38 \\ 0, \text{если } x > 0.38 \end{cases}$$

4. «Поношенный».

$$\mu_{WW}(x) = \begin{cases} 0, \text{если } x \leq 0.35 \\ \dfrac{x - 0.35}{0.03}, \text{если } 0.35 < x \leq 0.38 \\ 1, \text{если } 0.38 \leq x \leq 0.40 \\ \dfrac{0.45 - x}{0.05}, \text{если } 0.40 < x \leq 0.45 \\ 0, \text{если } x > 0.45 \end{cases}$$

5. «Закаленный в боях».

$$\mu_{BS}(x) = \begin{cases} 0, \text{если } x \leq 0.40 \\ \dfrac{x - 0.40}{0.05}, \text{если } 0.40 < x \leq 0.45 \\ 1, \text{если } 0.45 \leq x \leq 1 \end{cases}$$

3.2.2 Ликвидность

Терм-множество для данной лингвистической переменной: {«Очень низкая», «Низкая», «Средняя», «Высокая», «Очень высокая»}. Для каждого терма определена своя функция принадлежности:

1. «Очень низкая».

$$\mu_{VL}(x) = \begin{cases} 1, \text{если } 0 \leq x \leq 10 \\ \dfrac{50 - x}{40}, \text{если } 10 < x \leq 50 \\ 0, \text{если } x > 50 \end{cases}$$

2. «Низкая».

$$\mu_L(x) = \begin{cases} 0, \text{если } x \leq 30 \\ \dfrac{x-30}{70}, \text{если } 30 < x \leq 100 \\ 1, \text{если } 100 \leq x \leq 150 \\ \dfrac{300-x}{150}, \text{если } 150 < x \leq 300 \\ 0, \text{если } x > 300 \end{cases}$$

3. «Средняя».

$$\mu_M(x) = \begin{cases} 0, \text{если } x \leq 200 \\ \dfrac{x-200}{100}, \text{если } 200 < x \leq 300 \\ 1, \text{если } 300 \leq x \leq 400 \\ \dfrac{600-x}{200}, \text{если } 400 < x \leq 600 \\ 0, \text{если } x > 600 \end{cases}$$

4. «Высокая».

$$\mu_H(x) = \begin{cases} 0, \text{если } x \leq 500 \\ \dfrac{x-500}{100}, \text{если } 500 < x \leq 600 \\ 1, \text{если } 600 \leq x \leq 700 \\ \dfrac{900-x}{200}, \text{если } 700 < x \leq 900 \\ 0, \text{если } x > 900 \end{cases}$$

5. «Очень высокая».

$$\mu_{VH}(x) = \begin{cases} 0, \text{если } x \leq 800 \\ \dfrac{x-800}{100}, \text{если } 800 < x \leq 900 \\ 1, \text{если } x \geq 900 \end{cases}$$

3.2.3 Рыночная цена

Терм-множество для данной лингвистической переменной: {«Очень низкая», «Низкая», «Средняя», «Высокая», «Очень высокая»}. Для каждого терма определена своя функция принадлежности:

1. «Очень низкая».

$$\mu_{VL}(x) = \begin{cases} 1, \text{если } 0 \le x \le 0.5 \\ \dfrac{3-x}{2.5}, \text{если } 0.5 < x \le 3 \\ 0, \text{если } x > 3 \end{cases}$$

2. «Низкая».

$$\mu_{L}(x) = \begin{cases} 0, \text{если } x \le 1 \\ \dfrac{x-1}{20}, \text{если } 1 < x \le 21 \\ 1, \text{если } 21 \le x \le 30 \\ \dfrac{70-x}{40}, \text{если } 30 < x \le 70 \\ 0, \text{если } x > 70 \end{cases}$$

3. «Средняя».

$$\mu_{M}(x) = \begin{cases} 0, \text{если } x \le 50 \\ \dfrac{x-50}{50}, \text{если } 50 < x \le 100 \\ 1, \text{если } 100 \le x \le 150 \\ \dfrac{300-x}{150}, \text{если } 150 < x \le 300 \\ 0, \text{если } x > 300 \end{cases}$$

4. «Высокая».

$$\mu_{H}(x) = \begin{cases} 0, \text{если } x \le 250 \\ \dfrac{x-250}{150}, \text{если } 250 < x \le 400 \\ 1, \text{если } 400 \le x \le 800 \\ \dfrac{1500-x}{700}, \text{если } 800 < x \le 1500 \\ 0, \text{если } x > 1500 \end{cases}$$

5. «Очень высокая».

$$\mu_{VH}(x) = \begin{cases} 0, \text{если } x \le 1000 \\ \dfrac{x-1000}{500}, \text{если } 1000 < x \le 1500 \\ 1, \text{если } x \ge 1500 \end{cases}$$

3.2.4 Возраст

Возраст – временная характеристика предмета с момента его появления в игре. Терм-множество для данной лингвистической переменной: {«Новый»,

«Современный», «Средний», «Старый», «Винтажный»}. Для каждого терма определена своя функция принадлежности:

1. «Новый».

$$\mu_N(x) = \begin{cases} 1, \text{если } 0 \le x \le 0.5 \\ \dfrac{2-x}{1.5}, \text{если } 0.5 < x \le 2 \\ 0, \text{если } x > 2 \end{cases}$$

2. «Современный».

$$\mu_M(x) = \begin{cases} 0, \text{если } x \le 1 \\ \dfrac{x-1}{1}, \text{если } 1 < x \le 2 \\ 1, \text{если } 2 \le x \le 3 \\ \dfrac{5-x}{2}, \text{если } 3 < x \le 5 \\ 0, \text{если } x > 5 \end{cases}$$

3. «Средний».

$$\mu_{MA}(x) = \begin{cases} 0, \text{если } x \le 3 \\ \dfrac{x-3}{2}, \text{если } 3 < x \le 5 \\ 1, \text{если } 5 \le x \le 6 \\ \dfrac{8-x}{2}, \text{если } 6 < x \le 8 \\ 0, \text{если } x > 8 \end{cases}$$

4. «Старый».

$$\mu_O(x) = \begin{cases} 0, \text{если } x \le 6 \\ \dfrac{x-6}{2}, \text{если } 6 < x \le 8 \\ 1, \text{если } 8 \le x \le 9 \\ \dfrac{10-x}{1}, \text{если } 9 < x \le 10 \\ 0, \text{если } x > 10 \end{cases}$$

5. «Винтажный».

$$\mu_V(x) = \begin{cases} 0, \text{если } x \le 8 \\ \dfrac{x-8}{2}, \text{если } 8 < x \le 10 \\ 1, \text{если } x \ge 10 \end{cases}$$

3.2.5 Инвестиционная привлекательность

Терм-множество для данной лингвистической переменной: {«Очень низкая», «Низкая», «Средняя», «Высокая», «Очень высокая»}. Для каждого терма определена своя функция принадлежности:

1. «Очень низкая».

$$\mu_{VL}(x) = \begin{cases} 1, \text{если } 0 \le x \le 0.1 \\ \dfrac{0.3 - x}{0.2}, \text{если } 0.1 < x \le 0.3 \\ 0, \text{если } x > 0.3 \end{cases}$$

2. «Низкая».

$$\mu_{L}(x) = \begin{cases} 0, \text{если } x \le 0.2 \\ \dfrac{x - 0.2}{0.2}, \text{если } 0.2 < x \le 0.4 \\ 1, \text{если } 0.4 \le x \le 0.5 \\ \dfrac{0.6 - x}{0.1}, \text{если } 0.5 < x \le 0.6 \\ 0, \text{если } x > 0.6 \end{cases}$$

3. «Средняя».

$$\mu_{M}(x) = \begin{cases} 0, \text{если } x \le 0.5 \\ \dfrac{x - 0.5}{0.1}, \text{если } 0.5 < x \le 0.6 \\ 1, \text{если } 0.6 \le x \le 0.7 \\ \dfrac{0.8 - x}{0.1}, \text{если } 0.7 < x \le 0.8 \\ 0, \text{если } x > 0.8 \end{cases}$$

4. «Высокая».

$$\mu_{H}(x) = \begin{cases} 0, \text{если } x \le 0.7 \\ \dfrac{x - 0.7}{0.1}, \text{если } 0.7 < x \le 0.8 \\ 1, \text{если } 0.8 \le x \le 0.9 \\ \dfrac{1 - x}{0.1}, \text{если } 0.9 < x \le 1 \\ 0, \text{если } x > 1 \end{cases}$$

5. «Очень высокая».

$$\mu_{VH}(x) = \begin{cases} 0, \text{если } x \le 0.85 \\ \dfrac{x - 0.85}{0.1}, \text{если } 0.85 < x \le 0.95 \\ 1, \text{если } 0.95 \le x \le 1 \end{cases}$$

3.3 Построение продукционных правил

Для упрощения построения продукционных правил введены следующие нечеткие множества:

- «низкая степень износа»;
- «высокая ликвидность»;
- «высокая цена»;
- «старые скины»;
- «инвестиционная привлекательность».

Реализован класс Skin, представляющий предмет из игры Counter-Strike 2. Код файла skins.py представлен в Приложении А.

Функция create_sample_skins предназначена для создания нескольких объектов класса Skin. Созданные объекты представлены на Рисунке 3.3.1.

```
def create_sample_skins():
    """Создает список скинов для демонстрации"""
    return [
        Skin(name="AK-47 | Redline", float_value=0.156, liquidity=127, price=32.99, age_days=4302, paint_seed=randint(1, 999)),
        Skin(name="AWP | Dragon Lore", float_value=0.035, liquidity=1, price=11850, age_days=4171, paint_seed=randint(1, 999)),
        Skin(name="M4A1-S | Hyper Beast", float_value=0.366, liquidity=30, price=128.75, age_days=3883, paint_seed=randint(1, 999)),
        Skin(name="Karambit | Fade", float_value=0.0102, liquidity=5, price=2350, age_days=4492, paint_seed=randint(1, 999)),
        Skin(name="Sport Gloves | Vice", float_value=0.09, liquidity=3, price=2400, age_days=2846, paint_seed=randint(1, 999))
    ]
```

Рисунок 3.3.1 – Созданные скины

Нечеткое множество «Низкая степень износа» отображено на Рисунке 3.3.2.

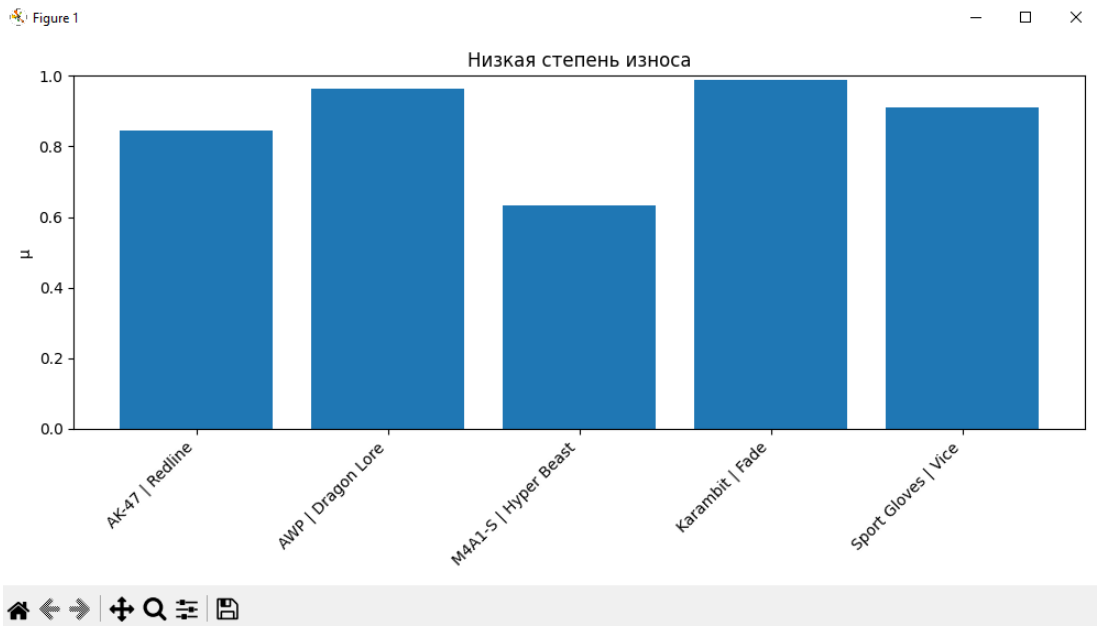

Рисунок 3.3.2 - Нечеткое множество «Низкая степень износа»

Нечеткое множество «Высокая ликвидность» отображено на Рисунке 3.3.3.

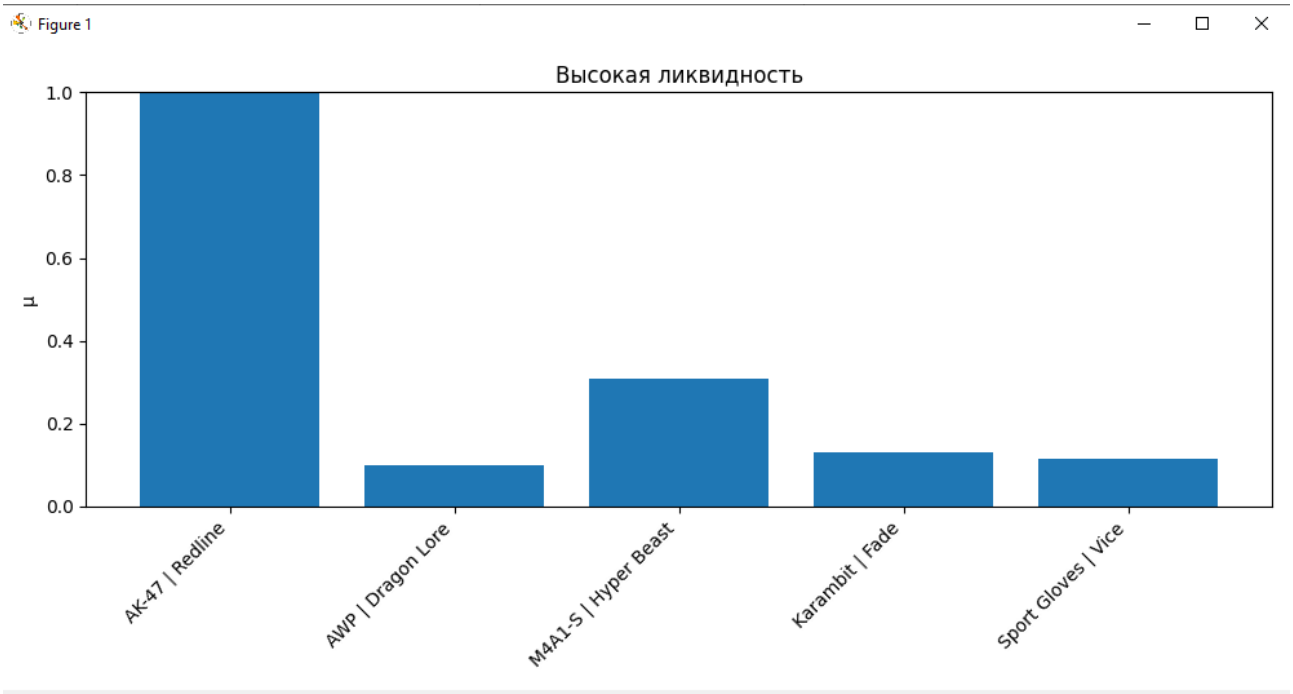

Рисунок 3.3.3 - Нечеткое множество «Высокая ликвидность»

Нечеткое множество «Высокая цена» отображено на Рисунке 3.3.4.

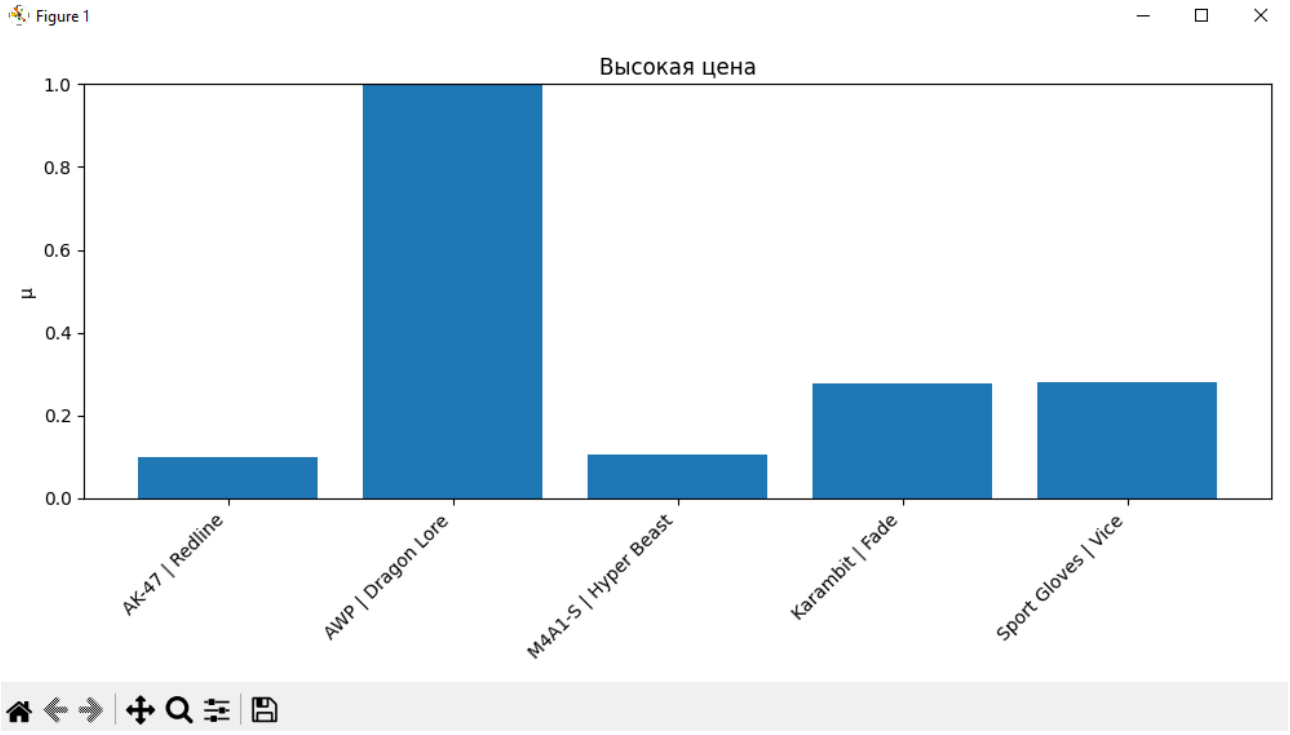

Рисунок 3.3.4 - Нечеткое множество «Высокая цена»

Нечеткое множество «Старые скины» отображено на Рисунке 3.3.5.

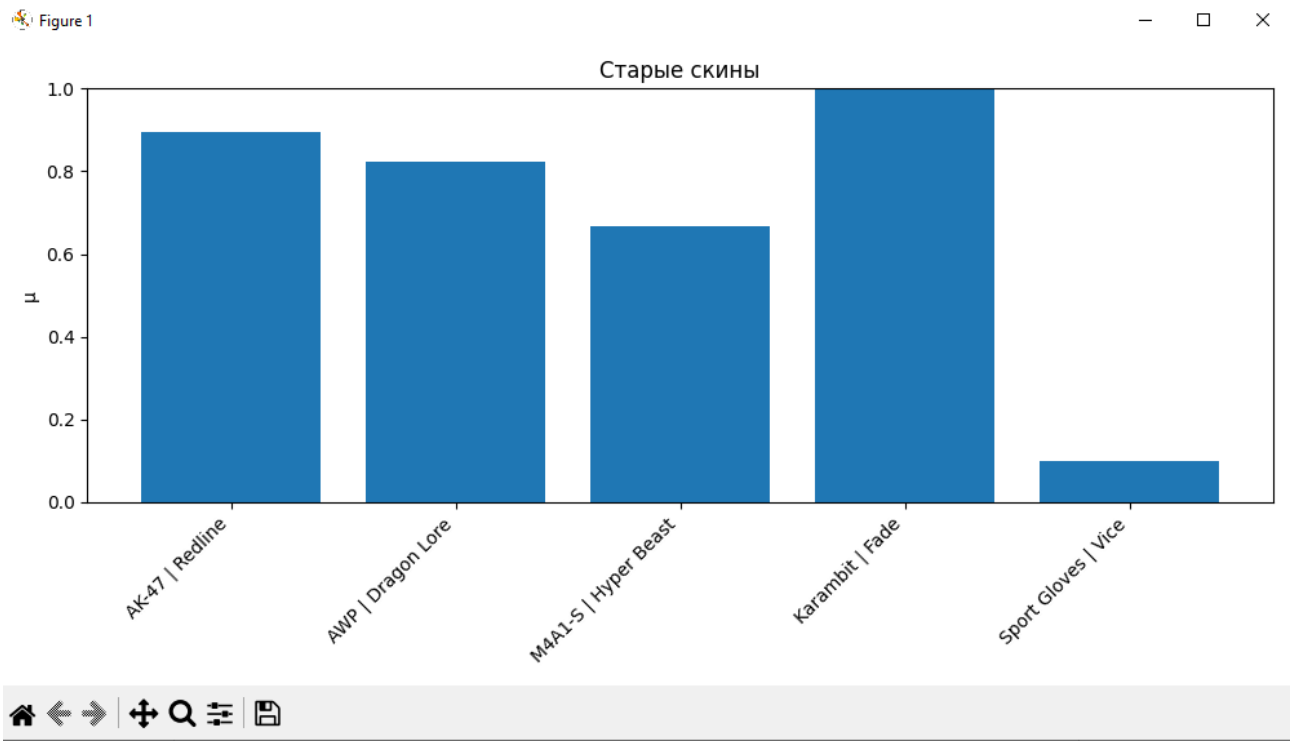

Рисунок 3.3.5 - Нечеткое множество «Старые скины»

Нечеткое множество «Инвестиционная привлекательность» отображено на Рисунке 3.3.6.

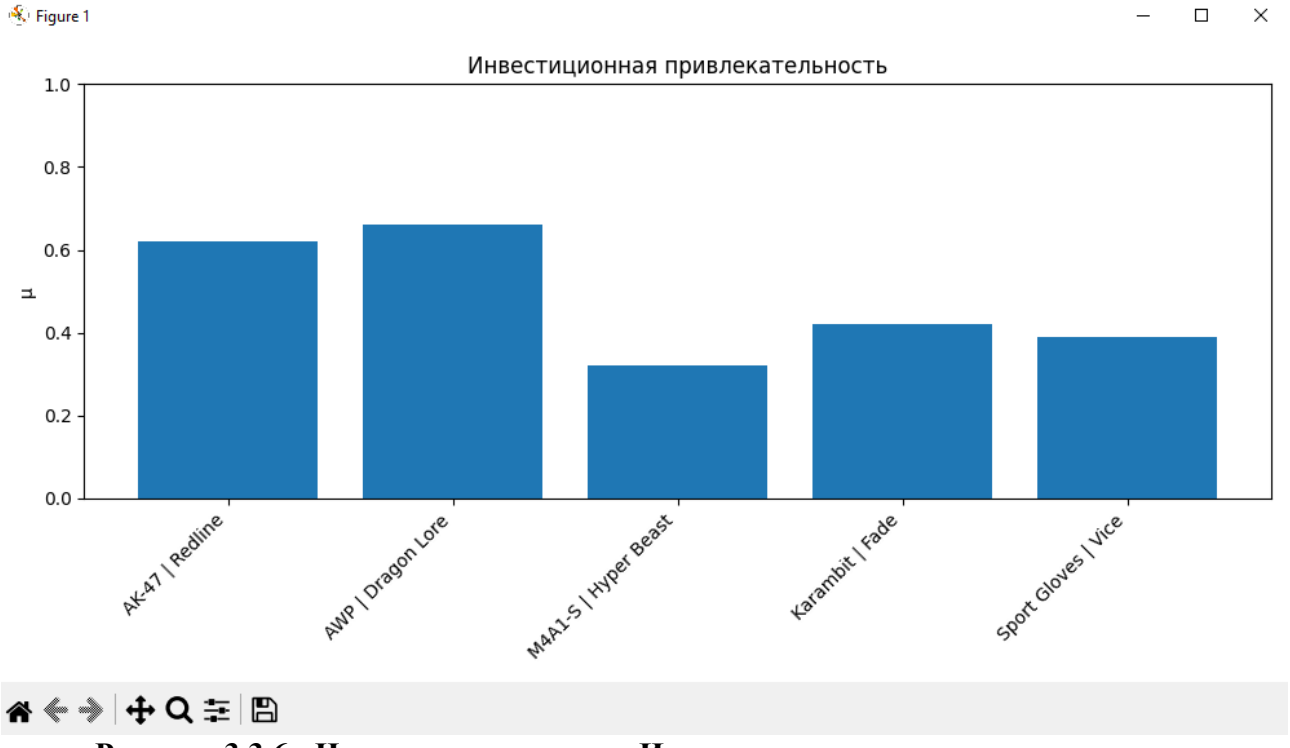

Рисунок 3.3.6 - Нечеткое множество «Инвестиционная привлекательность»

Для демонстрации логики построены следующие четыре правила:

1. ЕСЛИ степень износа низкая, ТО цена высокая (отношение: «Низкая степень износа → Высокая цена»).

2. ЕСЛИ цена высокая, ТО инвестиционная привлекательность высокая (отношение: «Высокая цена → Инвестиционная привлекательность»).
3. ЕСЛИ ликвидность высокая, ТО инвестиционная привлекательность высокая (отношение: «Высокая ликвидность → Инвестиционная привлекательность»).
4. ЕСЛИ скин старый, ТО цена высокая (отношение: Старые скины → Высокая цена).

3.4 Формирование матриц нечетких отношений

После построения нечетких множеств и определения продукционных правил необходимо перейти к формированию матриц нечетких отношений. Каждое продукционное правило вида $A \to B$ описывается матрицей размера $|A| \times |B|$, формируемой на основе операции пересечения нечетких множеств.

Таким образом, каждая строка матрицы соответствует элементу множества A, а каждый столбец — элементу множества B. Все значения лежат в диапазоне $[0; 1]$ и показывают силу связи между двумя элементами в продукционном правиле.

В отношении «Низкая степень износа → Высокая цена» посылка — множество «Низкая степень износа», а заключение — «Высокая цена». Для каждого скина x_i вычисляется его степень принадлежности обоим множествам, после чего элемент матрицы формируется по Формуле 2.6. Построенная матрица представлена на Рисунке 3.4.1 в виде тепловой карты.

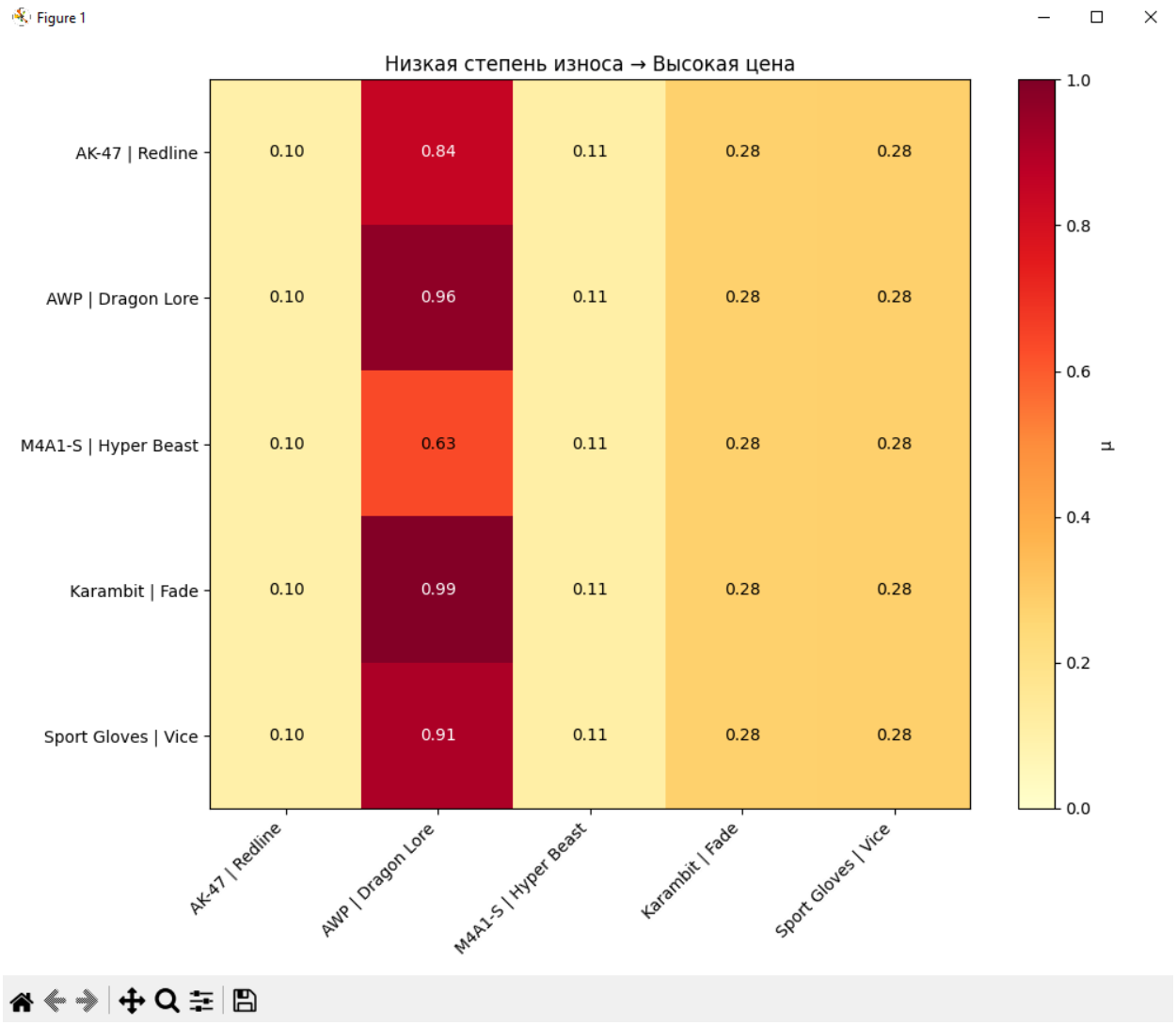

Рисунок 3.4.1 – Матрица для отношения «Низкая степень износа → Высокая цена»

Каждая ячейка матрицы отражает силу связи между конкретным скином (строка) и его принадлежностью к высокой цене (столбец) при условии низкой степени износа. Высокие значения (близкие к 1) указывают на то, что скин одновременно обладает низким износом и высокой ценой.

Матрица для отношения «Высокая цена → Инвестиционная привлекательность» представлена на Рисунке 3.4.2.

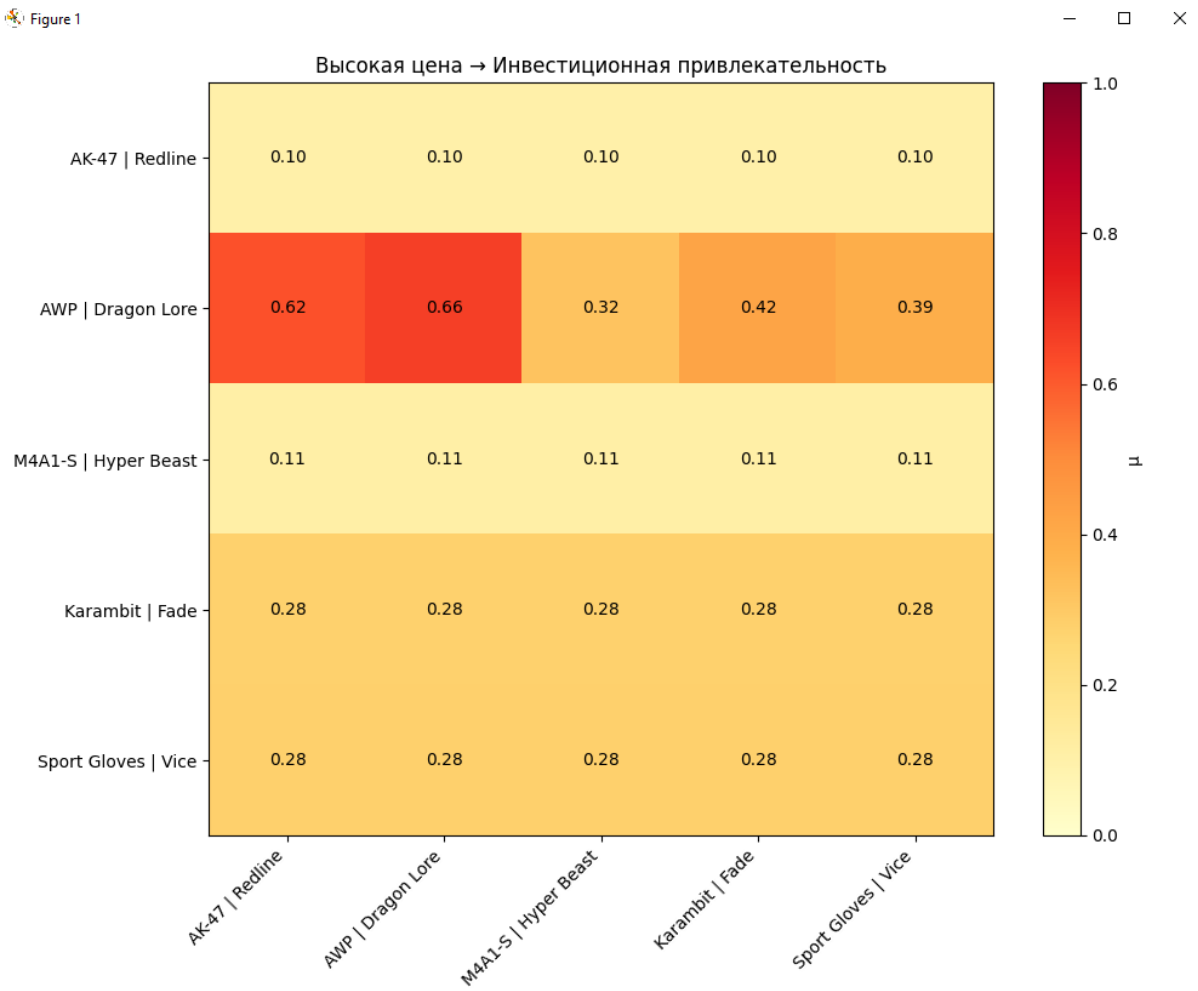

Рисунок 3.4.2 - Матрица для отношения «Высокая цена → Инвестиционная привлекательность»

Матрица связывает скины с высокой ценой и их инвестиционную привлекательность. Сильные связи показывают, что дорогие скины часто являются хорошими инвестиционными объектами. Однако не все дорогие скины одинаково привлекательны.

Матрица для отношения «Высокая ликвидность → Инвестиционная привлекательность» представлена на Рисунке 3.4.3.

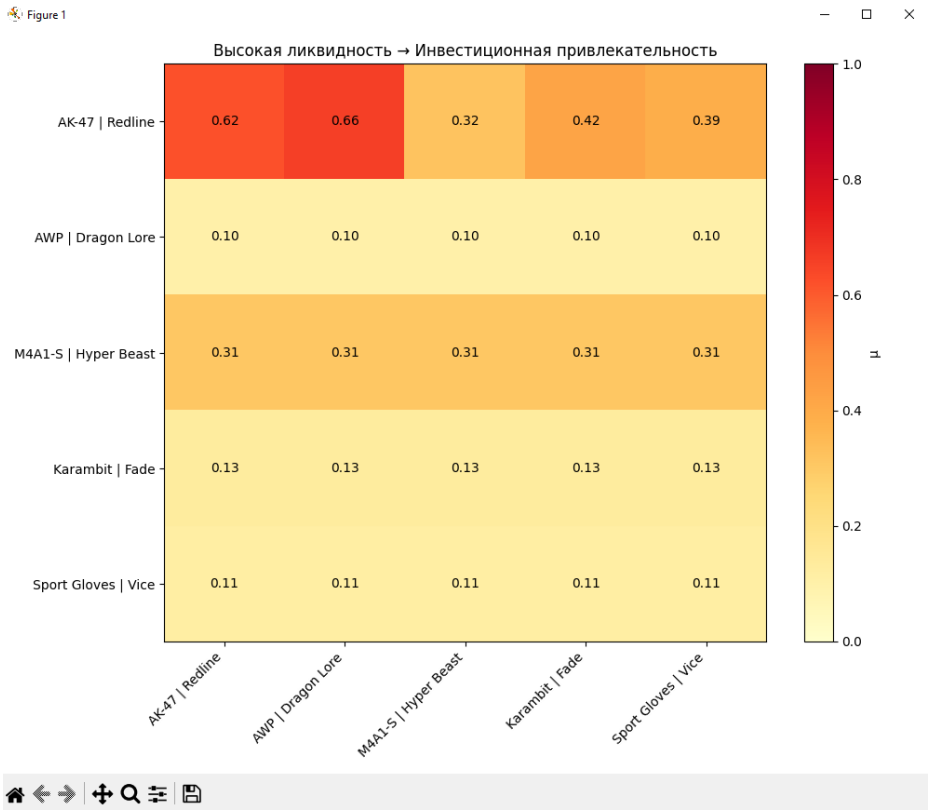

Рисунок 3.4.3 - Матрица для отношения «Высокая ликвидность → Инвестиционная привлекательность»

Высокая ликвидность обычно повышает привлекательность скина, так как его проще купить/продать без потери стоимости.

Матрица для отношения «Старые скины → Высокая цена» представлена на Рисунке 3.4.4.

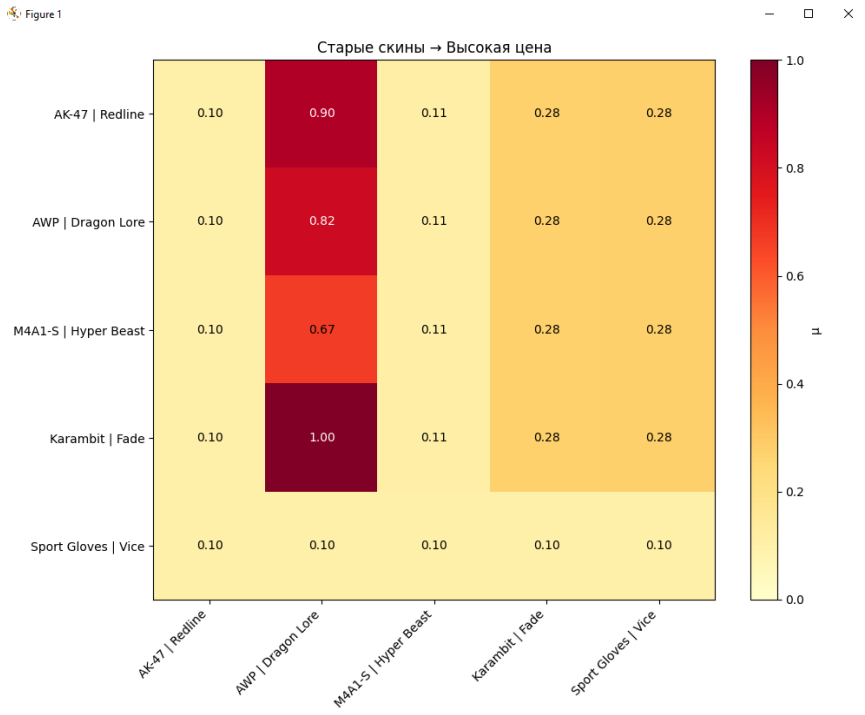

Рисунок 3.4.4 - Матрица для отношения «Старые скины → Высокая цена»

Матрица отражает связь возраста скина с его ценой. Старые скины часто имеют коллекционную ценность, что может приводить к высокой рыночной стоимости.

3.5 Композиция нечетких отношений

Композиция позволяет объединять два и более продукционных правила в единую логическую цепочку, что является важным шагом в построении сложных систем вывода.

В качестве примера рассматривается композиция отношений «Низкая степень износа → Высокая цена» и «Высокая цена → Инвестиционная привлекательность», которая позволяет получить правило «ЕСЛИ низкая степень износа, то высокая инвестиционная привлекательность».

Композиция позволяет исключить промежуточную лингвистическую переменную и построить отношение напрямую между двумя другими характеристиками.

Как и в случае формирования отдельных матриц, составленное отношение визуализируется с помощью тепловой карты. На Рисунке 3.5.1 приведена тепловая карта композиции отношений «Низкая степень износа → Высокая цена» и «Высокая цена → Инвестиционная привлекательность».

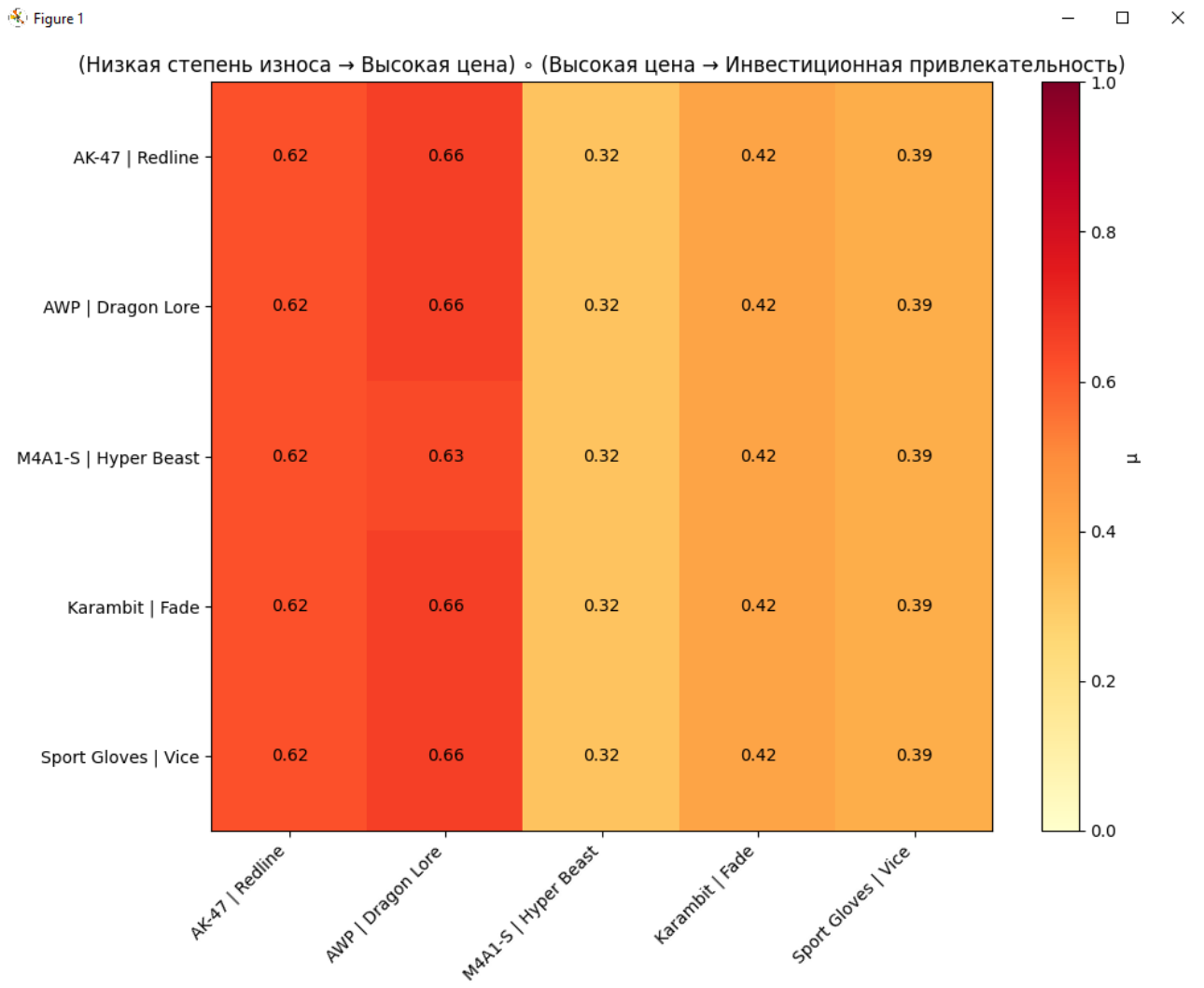

Рисунок 3.5.1 - Тепловая карта композиции отношений «Низкая степень износа → Высокая цена» и «Высокая цена → Инвестиционная привлекательность»

Визуально матрица отражает комплексную связь между износом и инвестиционной привлекательностью.

3.6 Транспонирование отношений

Транспонирование нечеткого отношения — это операция, обращающая направление отношения между множествами.

В матричном представлении транспонирование соответствует операции транспонирования матрицы: строки становятся столбцами, а столбцы — строками.

Исходное отношение «Низкая степень износа → Высокая цена» транспонировано в отношение «Высокая цена → Низкая степень износа» (Рисунок 3.6.1).

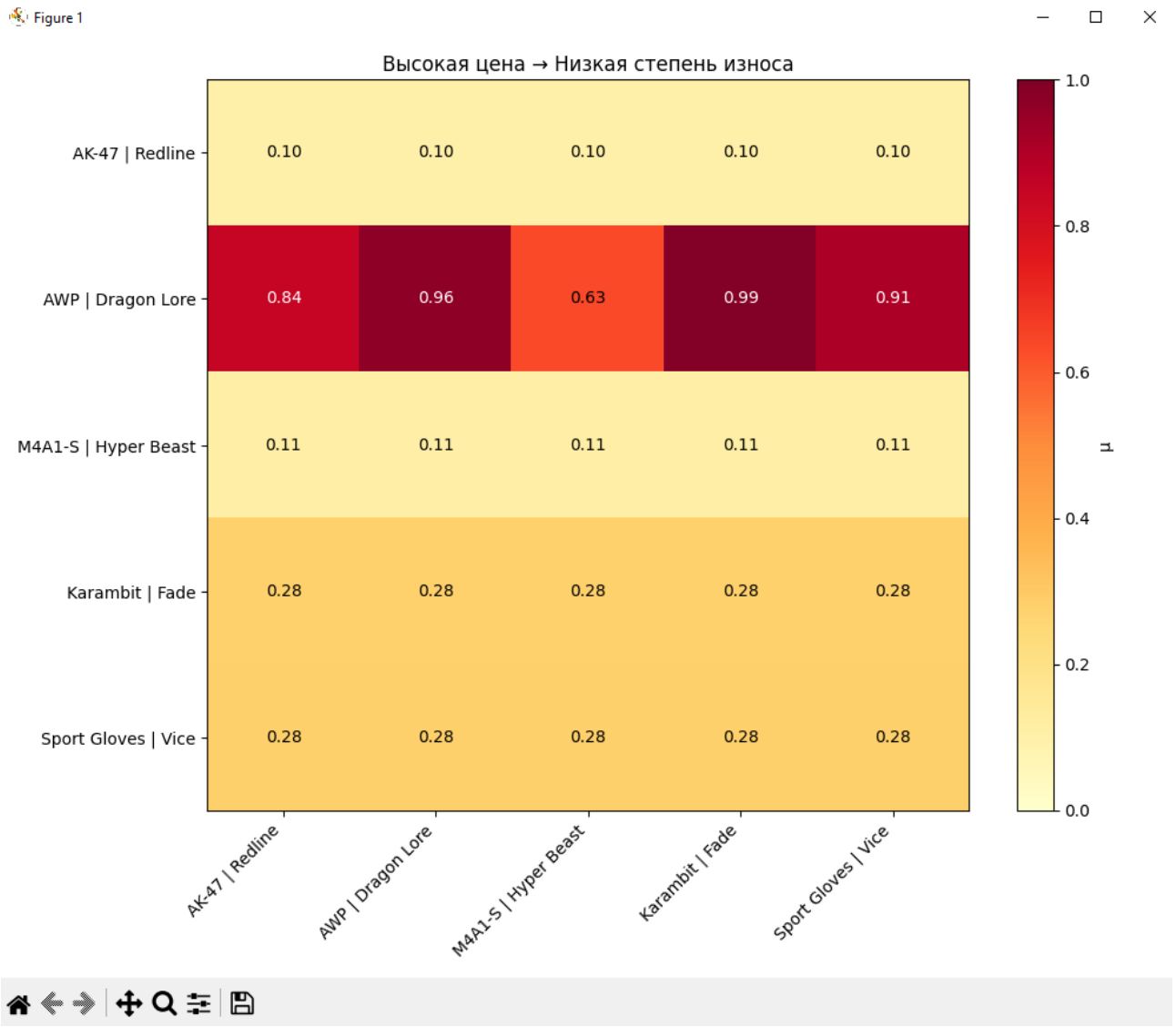

Рисунок 3.6.1 – Транспонирование отношения «Низкая степень износа → Высокая цена»

Полученная матрица является транспонированной для матрицы, представленной на Рисунке 3.4.1.

3.7 Дополнение нечеткого множества

Дополнение нечеткого множества — это унарная операция, которая для каждого элемента множества определяет степень его принадлежности к противоположному понятию.

Для множества «Низкая степень износа» операция дополнения создает множество «Не низкая степень износа» (или «Высокая степень износа»), где степень принадлежности каждого скина равна $1 - \mu_A(x)$. Дополнение множества «Низкая степень износа» представлено на Рисунке 3.7.1.

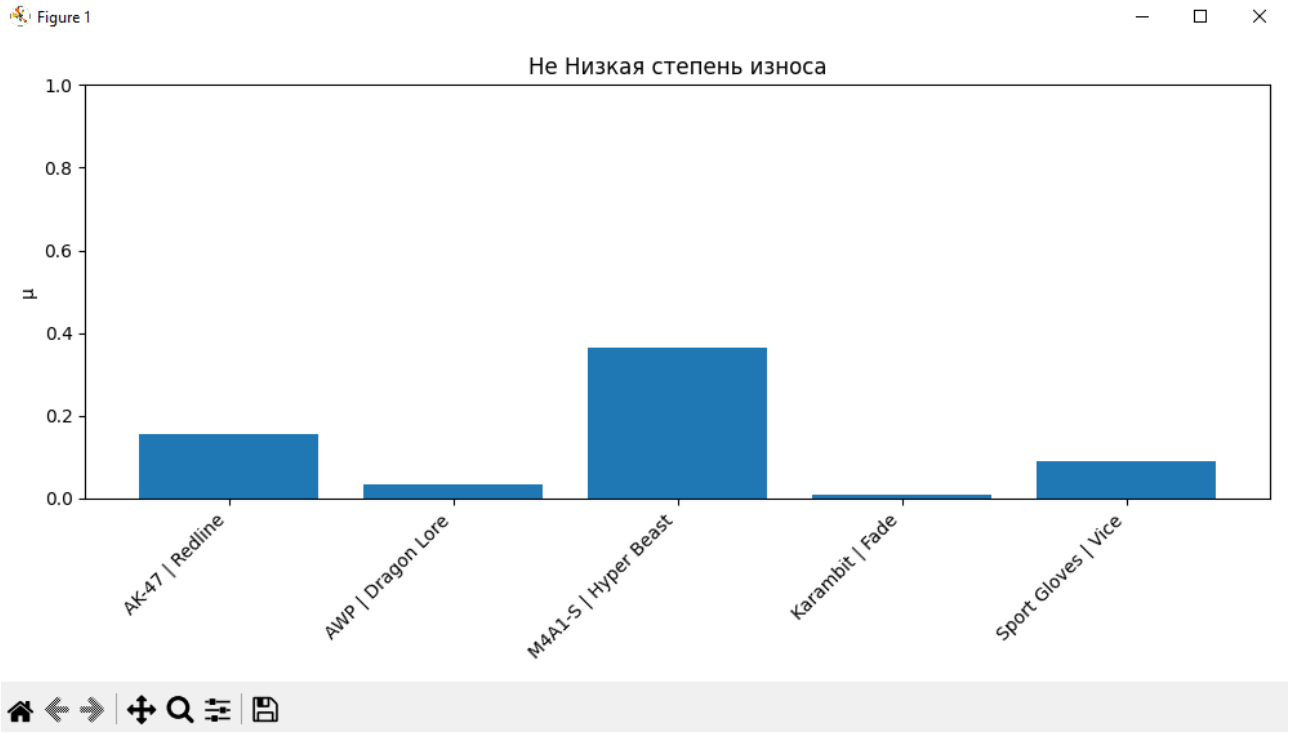

Рисунок 3.7.1 – Дополнение множества «Низкая степень износа»

3.8 Нечеткий вывод

Нечеткий вывод представляет собой процесс получения нового нечеткого множества на основе имеющегося факта и заданного продукционного правила вида «ЕСЛИ A, ТО B».

В работе демонстрируется применение нечеткого вывода на примере правила: ЕСЛИ степень износа низкая, ТО цена высокая (отношение «Низкая степень износа → Высокая цена»). В качестве фактического множества создается новое нечеткое множество: A′ = «Очень низкая степень износа» (Рисунок 3.8.1).

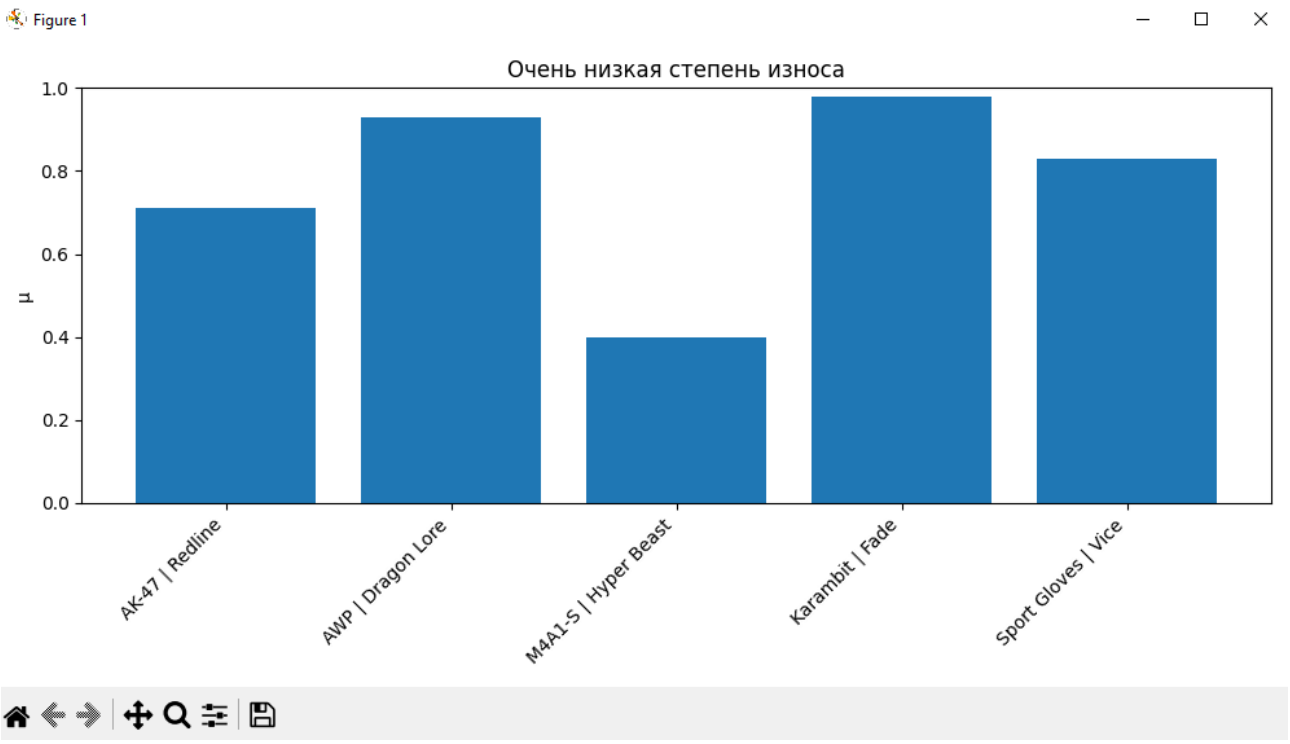

Рисунок 3.8.1 – Нечеткое множество «Очень низкая степень износа»

В итоге формируется новое нечеткое множество: «Вывод из очень низкая степень износа», которое показывает, насколько каждый предмет должен считаться «высокоценным», если его степень износа крайне низкая (Рисунок 3.8.2).

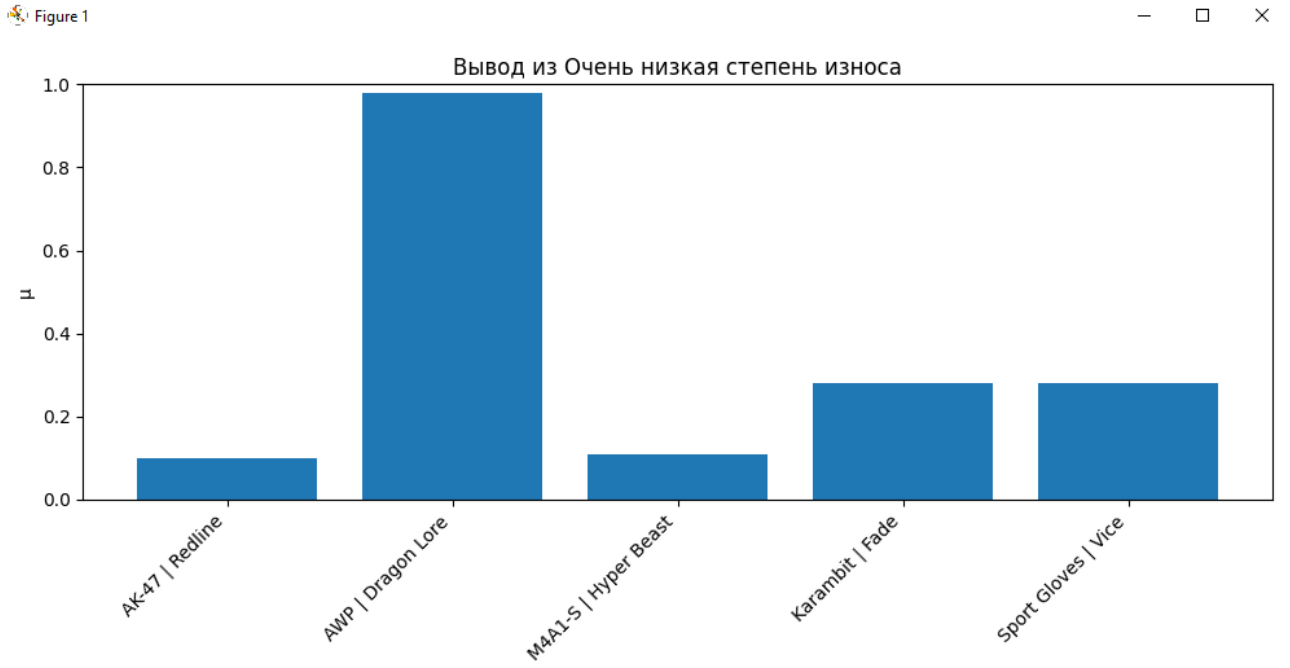

Рисунок 3.8.2 – Вывод из «Очень низкая степень износа»

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы изучены и реализованы основные положения теории нечетких множеств и нечетких отношений, включая построение продукционных правил, формирование матриц отношений и выполнение нечеткого логического вывода. На примере предметной области рынка скинов Counter-Strike 2 продемонстрирована применимость нечеткой логики для анализа объектов, характеристики которых имеют размытые границы и не поддаются строгой формализации в традиционных моделях.

В практической части заданы ключевые нечеткие множества, описывающие степень износа, ликвидность, цену, возраст и инвестиционную привлекательность игровых предметов. Для этих параметров сформированы функции принадлежности, построены графические представления и выполнено сравнение объектов по лингвистическим критериям.

На основе заданных множеств были сформированы продукционные правила типа «ЕСЛИ A, ТО B», что позволило построить матрицы нечетких отношений и выполнить их визуализацию в виде тепловых карт. Такая форма представления существенно облегчила интерпретацию взаимосвязей между характеристиками скинов.

Дополнительно была реализована композиция нечетких отношений, что позволило получить более сложные цепочки логических зависимостей и убедиться в корректности применения операции максимин-композиции. На завершающем этапе был продемонстрирован механизм нечеткого вывода по правилу нечеткого модус поненс, в результате которого сформировано новое нечеткое множество — заключение, отражающее влияние факта на целевую лингвистическую переменную.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Сорокин, А. Б. Безусловная оптимизация. [Электронный ресурс] : учебно-метод. пособие / А. Б. Сорокин, О. В. Платонова, Л. М. Железняк — М. РТУ МИРЭА , 2020.
2. Сорокин, А. Б. Введение в генетические алгоритмы: теория, расчеты и приложения. [Электронный ресурс] : учебно-метод. пособие / А. Б. Сорокин — М. МИРЭА , 2018.
3. Нечеткая логика [Электронный ресурс]: URL: https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D1%87%D1%91%D1%82%D0%BA%D0%B0%D1%8F_%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0 (Дата обращения: 12.10.2025).
4. Введение в нечеткую логику [Электронный ресурс]: URL: https://habr.com/ru/companies/timeweb/articles/713620/ (Дата обращения: 14.10.2025).
5. Нечеткая логика — математические основы [Электронный ресурс]: URL: https://loginom.ru/blog/fuzzy-logic (Дата обращения: 15.10.2025).

ПРИЛОЖЕНИЯ

Приложение А — Код файлы skins.py.

Приложение Б — Код файлы relations.py.

Приложение В — Код файлы main.py.

Приложение Г — Код файлы linguistic_variable.py.

Приложение Д — Код файлы fuzzy_system.py.

Приложение Е — Код файлы fuzzy_sets.py.

Приложение А

Код файлы skins.py

Листинг А – Код файлы skins.py

```
from random import randint

class Skin:
    """
    Класс для представления конкретного скина с его параметрами.
    """

    def __init__(self, name: str, float_value: float, liquidity: float,
                 price: float, age_days: float, paint_seed: int = 0):
        self.name = name
        self.float_value = float_value
        self.liquidity = liquidity
        self.price = price
        self.age_days = age_days
        self.paint_seed = paint_seed

    def __str__(self):
        return f"{self.name}: float={self.float_value:.3f}, price=${self.price},
age={self.age_days} days"

    def __repr__(self):
        return f"Skin('{self.name}')"

def create_sample_skins():
    """Создает список скинов для демонстрации"""
    return [
        Skin(name="AK-47 | Redline", float_value=0.156, liquidity=127,
price=32.99, age_days=4302, paint_seed=randint(1, 999)),
        Skin(name="AWP | Dragon Lore", float_value=0.035, liquidity=1,
price=11850, age_days=4171, paint_seed=randint(1, 999)),
        Skin(name="M4A1-S | Hyper Beast", float_value=0.366, liquidity=30,
price=128.75, age_days=3883, paint_seed=randint(1, 999)),
        Skin(name="Karambit | Fade", float_value=0.0102, liquidity=5,
price=2350, age_days=4492, paint_seed=randint(1, 999)),
        Skin(name="Sport Gloves | Vice", float_value=0.09, liquidity=3,
price=2400, age_days=2846, paint_seed=randint(1, 999))
    ]
```

Приложение Б

Код файлы relations.py

Листинг Б – Код файлы relations.py

```
from typing import List

class FuzzyRelation:
    """
    Класс для представления нечеткого отношения между двумя множествами.
    """

    def __init__(self, rows: List[str], columns: List[str],
                 matrix: List[List[float]], name: str = "Неименованное
отношение"):
        """
        Args:
            rows: имена элементов для строк (посылка)
            columns: имена элементов для столбцов (заключение)
            matrix: матрица отношения
            name: имя отношения
        """
        self.rows = rows
        self.columns = columns
        self.matrix = matrix
        self.name = name

    def __str__(self):
        return f"FuzzyRelation: {self.name}
({len(self.rows)}x{len(self.columns)})"

    def transpose(self) -> 'FuzzyRelation':
        """Транспонирование отношения (меняет посылку и заключение местами)."""
        transposed_matrix = []
        for j in range(len(self.columns)):
            row = []
            for i in range(len(self.rows)):
                row.append(self.matrix[i][j])
            transposed_matrix.append(row)

        return FuzzyRelation(
            rows=self.columns,
            columns=self.rows,
            matrix=transposed_matrix,
            name=f"Транспонированное: {self.name}"
        )

    def complement(self) -> 'FuzzyRelation':
        """Дополнение отношения (1 - значение)."""
        complemented_matrix = []
        for i in range(len(self.rows)):
            row = [1.0 - val for val in self.matrix[i]]
            complemented_matrix.append(row)

        return FuzzyRelation(
            rows=self.rows,
            columns=self.columns,
            matrix=complemented_matrix,
            name=f"Дополнение: {self.name}"
        )
```

Приложение В

Код файлы main.py

Листинг В – Код файлы main.py

```
from skins import Skin, create_sample_skins
from fuzzy_sets import FuzzySet, TriangularFuzzySet, TrapezoidalFuzzySet
from relations import FuzzyRelation
import numpy as np
import matplotlib.pyplot as plt

def create_skin_fuzzy_sets(skins):
    """
    Создает нечеткие множества для скинов на основе их параметров.
    """
    # Множество "Низкая степень износа"
    good_condition = {}
    for skin in skins:
        value = 1.0 - skin.float_value
        good_condition[skin.name] = round(value, 3)

    # Множество "Высокая ликвидность"
    high_liquidity = {}
    max_liquidity = max(skin.liquidity for skin in skins)
    min_liquidity = min(skin.liquidity for skin in skins)
    for skin in skins:
        if max_liquidity == min_liquidity:
            value = 0.0
        else:
            offset = 0.1
            normalized = (skin.liquidity - min_liquidity) / (max_liquidity -
min_liquidity)
            value = offset + (1 - offset) * normalized
        high_liquidity[skin.name] = round(value, 3)

    # Множество "Высокая цена"
    high_price = {}
    max_price = max(skin.price for skin in skins)
    min_price = min(skin.price for skin in skins)
    for skin in skins:
        if max_price == min_price:
            value = 0.0
        else:
            offset = 0.1
            normalized = (skin.price - min_price) / (max_price - min_price)
            value = offset + (1 - offset) * normalized
        high_price[skin.name] = round(value, 3)

    # Множество "Старые скины"
    old_skins = {}
    max_age = max(skin.age_days for skin in skins)
    min_age = min(skin.age_days for skin in skins)
    for skin in skins:
        if max_age == min_age:
            value = 0.5
        else:
            offset = 0.1
            normalized = (skin.age_days - min_age) / (max_age - min_age)
            value = offset + (1 - offset) * normalized
        old_skins[skin.name] = round(value, 3)
```

Продолжение Листинга В

```
    # Множество "Инвестиционная привлекательность"
    investment = {}
    for skin in skins:
        condition_score = 1.0 - skin.float_value
        price_score = min(skin.price / 10000.0, 1.0)
        liquidity_score = min(skin.liquidity / 100.0, 1.0)
        value = (condition_score + price_score + liquidity_score) / 3.0
        investment[skin.name] = round(value, 2)

    return {
        "Низкая степень износа": FuzzySet("Низкая степень износа",
good_condition),
        "Высокая ликвидность": FuzzySet("Высокая ликвидность", high_liquidity),
        "Высокая цена": FuzzySet("Высокая цена", high_price),
        "Старые скины": FuzzySet("Старые скины", old_skins),
        "Инвестиционная привлекательность": FuzzySet("Инвестиционная
привлекательность", investment),
    }

def build_relation(set_a: FuzzySet, set_b: FuzzySet) -> FuzzyRelation:
    """Строит продукционное отношение между двумя нечеткими множествами."""
    elements_a = set_a.get_elements()
    elements_b = set_b.get_elements()

    matrix = []
    for elem_a in elements_a:
        row = []
        mu_a = set_a.mu(elem_a)
        for elem_b in elements_b:
            mu_b = set_b.mu(elem_b)
            row.append(min(mu_a, mu_b))
        matrix.append(row)

    return FuzzyRelation(elements_a, elements_b, matrix,
                         f"{set_a.name} → {set_b.name}")

def plot_relation(relation: FuzzyRelation, title = None):
    """Визуализация отношения как тепловой карты."""
    matrix = np.array(relation.matrix)
    rows = relation.rows
    cols = relation.columns

    plt.figure(figsize=(10, 8))
    plt.imshow(matrix, aspect='auto', cmap='YlOrRd', vmin=0, vmax=1)
    plt.colorbar(label='μ')

    plt.xticks(range(len(cols)), cols, rotation=45, ha='right')
    plt.yticks(range(len(rows)), rows)

    plt.title(title if title is not None else relation.name)

    for i in range(len(rows)):
        for j in range(len(cols)):
            plt.text(j, i, f'{matrix[i, j]:.2f}',
                    ha='center', va='center',
                    color='black' if matrix[i, j] < 0.7 else 'white')

    plt.tight_layout()
    plt.show()

def compose_relations(r1: FuzzyRelation, r2: FuzzyRelation) -> FuzzyRelation:
```

Продолжение Листинга В

```
    """Композиция двух отношений (max-min композиция)."""
    if r1.columns != r2.rows:
        raise ValueError("Несовместимые отношения для композиции")

    n = len(r1.rows)
    m = len(r2.columns)
    p = len(r1.columns)

    result_matrix = []
    for i in range(n):
        row = []
        for j in range(m):
            max_val = 0.0
            for k in range(p):
                val = min(r1.matrix[i][k], r2.matrix[k][j])
                max_val = max(max_val, val)
            row.append(max_val)
        result_matrix.append(row)

    return FuzzyRelation(r1.rows, r2.columns, result_matrix,
                         f"({r1.name}) ∘ ({r2.name})")

def main():
    # Создание объектов скинов
    skins = create_sample_skins()
    print(f"Создано {len(skins)} скинов:")
    for skin in skins:
        print(f"   - {skin}")

    # Построение множеств
    fuzzy_sets = create_skin_fuzzy_sets(skins)
    print(f"\nСоздано {len(fuzzy_sets)} нечетких множеств:")
    for name, fset in fuzzy_sets.items():
        print(f"   - {name}: {dict(zip(fset.get_elements(),
fset.get_values()))}")

    # Визуализация множеств
    print("\nВизуализация нечетких множеств:")
    for name, fset in fuzzy_sets.items():
        print(f"   - Визуализация {name}")
        fset.plot()

    # Построение отношений
    print("\nПостроение продукционных отношений:")

    # Отношение 1: Низкая степень износа → Высокая цена
    relation1 = build_relation(fuzzy_sets["Низкая степень износа"],
                               fuzzy_sets["Высокая цена"])
    print(f"   - {relation1.name}")
    plot_relation(relation1)

    # Отношение 2: Высокая цена → Инвестиционная привлекательность
    relation2 = build_relation(fuzzy_sets["Высокая цена"],
                               fuzzy_sets["Инвестиционная привлекательность"])
    print(f"   - {relation2.name}")
    plot_relation(relation2)

    # Отношение 3: Высокая ликвидность → Инвестиционная привлекательность
    relation3 = build_relation(fuzzy_sets["Высокая ликвидность"],
                               fuzzy_sets["Инвестиционная привлекательность"])
    print(f"   - {relation3.name}")
```

Окончание Листинга В

```
    plot_relation(relation3)

    # Отношение 4: Старые скины → Высокая цена
    relation4 = build_relation(fuzzy_sets["Старые скины"],
                               fuzzy_sets["Высокая цена"])
    print(f"   - {relation4.name}")
    plot_relation(relation4)

    # Композиция отношений
    print("\nКомпозиция отношений:")
    print(f"   - ({relation1.name}) ∘ ({relation2.name})")
    composition = compose_relations(relation1, relation2)
    plot_relation(composition)

    # Транспонирование
    print("\nТранспонирование отношения:")
    relation1_transposed = relation1.transpose()
    print(f"   - Транспонированное: {relation1_transposed.name}")
    plot_relation(relation1_transposed, "Высокая цена → Низкая степень износа")

    # Дополнение множеств
    print("\nДополнение нечетких множеств:")
    for name, fset in list(fuzzy_sets.items())[:1]:
        complement_set = fset.complement()
        print(f"   - Дополнение {name}: {complement_set.name}")
        complement_set.plot()

    # Правила вывода (modus ponens)
    print("\n8. Применение правила вывода (modus ponens):")

    # Создаем новое множество "Очень низкая степень износа"
    very_good_condition = {}
    for skin in skins:
        value = (1.0 - skin.float_value) ** 2
        very_good_condition[skin.name] = round(value, 2)

    fact_set = FuzzySet("Очень низкая степень износа", very_good_condition)
    print(f"   - Факт: {fact_set.name}")
    fact_set.plot()

    # Применяем правило modus ponens
    print(f"   - Применяем правило: {fact_set.name} → {relation1.name}")
    result = fact_set.apply_modus_ponens(relation1)
    if result:
        print(f"   - Результат вывода: {result.name}")
        result.plot()

if __name__ == "__main__":
    main()
```

Приложение Г

Код файлы linguistic_variable.py

Листинг Г – Код файлы linguistic_variable.py

```
from typing import Dict
from fuzzy_sets import FuzzySet
import numpy as np

class LinguisticVariable:
    """
    Лингвистическая переменная: универсум (min,max), дискретизация
    и набор терминов (нечетких множеств).
    """

    def __init__(
        self, name: str, domain_min: float, domain_max: float, num_points: int =
100
    ):
        self.name = name
        self.domain_min = float(domain_min)
        self.domain_max = float(domain_max)
        self.num_points = int(num_points)
        self.terms: Dict[str, FuzzySet] = {}

    def add_term(self, fuzzy_set: FuzzySet):
        self.terms[fuzzy_set.name] = fuzzy_set

    def universe(self):
        """Возвращает список дискретных точек универсума."""
        return np.linspace(self.domain_min, self.domain_max, self.num_points)

    def membership_vector(self, term_name: str):
        """Возвращает вектор значений mu(x) по дискретному универсу для
заданного терма."""
        if term_name not in self.terms:
            raise KeyError(f"Term {term_name} not found in {self.name}")
        fs = self.terms[term_name]
        xs = self.universe()
        return [fs.mu(x) for x in xs]

    def fuzzify(self, x: float):
        """Возвращает словарь {term: mu(x)} для одной точки x."""
        return {name: fs.mu(x) for name, fs in self.terms.items()}

    def __repr__(self):
        return f"LinguisticVariable({self.name}, [{self.domain_min},
{self.domain_max}], terms={list(self.terms.keys())})"
```

Приложение Д

Код файлы fuzzy_system.py

Листинг Д – Код файлы fuzzy_system.py

```
from typing import Dict, List
import numpy as np
from linguistic_variable import LinguisticVariable
from relations import FuzzyRelation

class FuzzySystem:
    """
    Класс для хранения переменных, правил (отношений) и выполнения операций:
    - построение отношения
    - композиция отношений
    - применение правила (вывод)
    """

    def __init__(self):
        self.variables: Dict[str, LinguisticVariable] = {}
        self.relations: Dict[str, FuzzyRelation] = {}

    def add_variable(self, var: LinguisticVariable):
        self.variables[var.name] = var

    def build_relation(
        self,
        var_x_name: str,
        term_x: str,
        var_y_name: str,
        term_y: str,
        relation_name: str,
    ):
        """
        Построить отношение R = term_x * term_y (product via min),
        дискретизация берется из переменных.
        """
        X = self.variables[var_x_name]
        Y = self.variables[var_y_name]
        x_uni = X.universe()
        y_uni = Y.universe()
        mu_x = X.membership_vector(term_x)
        mu_y = Y.membership_vector(term_y)
        R = FuzzyRelation.from_product(x_uni, y_uni, mu_x, mu_y)
        self.relations[relation_name] = R
        return R

    def compose_relations(self, r_name: str, s_name: str, out_name: str):
        R = self.relations[r_name]
        S = self.relations[s_name]
        T = R.compose_max_min(S)
        self.relations[out_name] = T
        return T

    def apply_rule(self, a_values: List[float], relation_name: str) ->
List[float]:
        """
        Применение правила: A' ∘ R = B' , где A' — вектор степеней по
дискретному универсуму X.
        Вход a_values должен иметь длину len(R.x_universe).
        Возвращает вектор длины len(R.y_universe) — B'.
```

Окончание Листинга Д

```
        """
        R = self.relations[relation_name]
        a = np.array(a_values, dtype=float)
        assert a.shape[0] == R.n
        b = np.zeros(R.m, dtype=float)
        for j in range(R.m):
            mins = [min(a[i], float(R.matrix[i, j])) for i in range(R.n)]
            b[j] = max(mins) if mins else 0.0
        return b.tolist()
```

Приложение Е

Код файлы fuzzy_sets.py

Листинг Е – Код файлы fuzzy_sets.py

```
import matplotlib.pyplot as plt
from math import exp
from typing import Dict, Union, Callable, Optional

class FuzzySet:
    """
    Универсальный класс нечеткого множества.
    """
    def __init__(self, name: str, data: Union[Dict, Callable] = None):
        """
        Args:
            name: имя множества
            data: либо словарь {элемент: степень_принадлежности},
                  либо функция принадлежности mu(x)
        """
        self.name = name
        if isinstance(data, dict):
            self.data_type = "discrete"
            self.data = data
        elif callable(data):
            self.data_type = "continuous"
            self.mu_func = data
        else:
            self.data_type = "empty"
            self.data = {}

    def mu(self, x):
        """Возвращает степень принадлежности x к множеству в [0,1]."""
        if self.data_type == "discrete":
            return self.data.get(x, 0.0)
        elif self.data_type == "continuous":
            val = self.mu_func(x)
            return max(0.0, min(1.0, val))
        return 0.0

    def __call__(self, x):
        return self.mu(x)

    def get_elements(self):
        """Возвращает список элементов (для дискретного множества)."""
        if self.data_type == "discrete":
            return list(self.data.keys())
        return []

    def get_values(self):
        """Возвращает список значений принадлежности (для дискретного
множества)."""
        if self.data_type == "discrete":
            return list(self.data.values())
        return []

    def plot(self):
        """Визуализация дискретного множества."""
        if self.data_type != "discrete":
            print(f"Множество {self.name} не является дискретным для
визуализации")
```

Окончание Листинга Е

```
            return

        elements = self.get_elements()
        values = self.get_values()

        plt.figure(figsize=(10, 5))
        plt.bar(range(len(elements)), values)
        plt.xticks(range(len(elements)), elements, rotation=45, ha='right')
        plt.ylim(0, 1)
        plt.ylabel('μ')
        plt.title(self.name)
        plt.tight_layout()
        plt.show()

    def apply_modus_ponens(self, relation) -> Optional['FuzzySet']:
        """
        Применяет правило modus ponens: B' = A' ∘ (A → B)

        Args:
            relation: отношение A → B
        Returns:
            Нечеткое множество B'
        """
        if self.data_type != "discrete" or relation.rows != self.get_elements():
            print("Невозможно применить modus ponens: несовместимые множества")
            return None

        b_prime = {}
        elements_a = self.get_elements()
        elements_b = relation.columns

        for j, elem_b in enumerate(elements_b):
            max_val = 0.0
            for i, elem_a in enumerate(elements_a):
                mu_a = self.mu(elem_a)
                mu_r = relation.matrix[i][j]
                max_val = max(max_val, min(mu_a, mu_r))
            b_prime[elem_b] = round(max_val, 2)

        return FuzzySet(f"Вывод из {self.name}", b_prime)

    def complement(self) -> 'FuzzySet':
        """Возвращает дополнение нечеткого множества."""
        if self.data_type == "discrete":
            comp_data = {k: 1.0 - v for k, v in self.data.items()}
            return FuzzySet(f"Не {self.name}", comp_data)
        else:
            def comp_func(x):
                return 1.0 - self.mu(x)
            return FuzzySet(f"Не {self.name}", comp_func)
```