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

Институт Информационных Технологий

Кафедра Вычислительной Техники

ПРАКТИЧЕСКАЯ РАБОТА №2

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 4
1 ПОСТАНОВКА ЗАДАЧИ 5
2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 6
2.1 Нечеткие множества 6
2.2 Свойства нечетких множеств 7
2.3 Операции над нечеткими множествами 9
2.4 Отношения и операции над ними 9
2.5 Нечеткие отношения 10
2.6 Композиция нечетких отношений 11
2.7 Нечеткий вывод 12
2.8 Нечеткие системы 12
2.9 Нечеткая база знаний 13
2.10 Фаззификация 14
2.11 Агрегирование условий правила 14
2.12 Активизация (импликация) 14
2.13 Аккумуляция (объединение заключений правил) 15
2.14 Дефаззификация 15
3 ПРАКТИЧЕСКАЯ ЧАСТЬ 16
3.1 Предметная область 16
3.2 Определение нечетких множеств 17
3.2.1 Степень износа 17
3.2.2 Ликвидность 18
3.2.3 Рыночная цена 19
3.2.4 Возраст 20
3.2.5 Инвестиционная привлекательность 21
3.3 Проектирование системы нечеткого управления 23
3.3.1 Создание функций принадлежности 23
3.3.2 Определение лингвистических переменных 23

3.3.3 Построение базы правил..... 26
3.3.4 Фаззификация входных данных 27
3.3.5 Таблица результатов 27
ЗАКЛЮЧЕНИЕ 28
СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 29
ПРИЛОЖЕНИЯ..... 30

ВВЕДЕНИЕ

Современные цифровые платформы и игровые экосистемы формируют новые полноценные рынки, на которых обращаются виртуальные активы, такие как предметы персонализации, коллекционные объекты и внутриигровые модели. Одним из крупнейших подобных рынков является экономика игрового сообщества Counter-Strike, где предметы имеют рыночную стоимость, ликвидность и могут выступать объектами инвестирования. Вследствие значительных колебаний цен, разнообразия характеристик предметов и отсутствия однозначных аналитических критериев оценка их инвестиционной привлекательности представляет собой сложную экспертную задачу.

Традиционные количественные методы анализа зачастую оказываются недостаточно эффективными из-за размытости границ между качествами предметов, субъективности эстетических параметров и неполноты доступной информации. В подобных условиях особую ценность приобретают методы мягких вычислений, позволяющие формализовать неопределённость и экспертные рассуждения. Одним из наиболее зрелых и широко применяемых подходов является аппарат нечеткой логики, обеспечивающий возможность обработки лингвистических переменных, нечётких границ и комплексных правил принятия решений.

Целью данной работы является разработка и исследование нечеткой системы оценки инвестиционной привлекательности предметов игровой экономики на основе их технических и рыночных характеристик.

Практическая значимость исследования заключается в возможности применения разработанной модели в аналитических сервисах, торговых платформах и инструментах поддержки принятия решений, связанных с прогнозированием стоимости цифровых активов, инвестиционными стратегиями и автоматизированным ранжированием предложений на рынке.

1 ПОСТАНОВКА ЗАДАЧИ

Цель работы: приобрести навыки проектирования и реализации систем нечёткого управления.

Задачи: добавить набор входных и выходных данных, которые описывают изменение степени принадлежности множества определённой величине путём математического распределения нечётких множеств относительно этой величины, добавить набор аксиом (продукционных правил), которые определяют зависимость между определёнными наборами выходных и выходных данных, при помощи графической интерпретации в двумерном и трёхмерном (если есть необходимое количество величин) пространствах определить правильность полученной модели и описать в отчёте.

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.

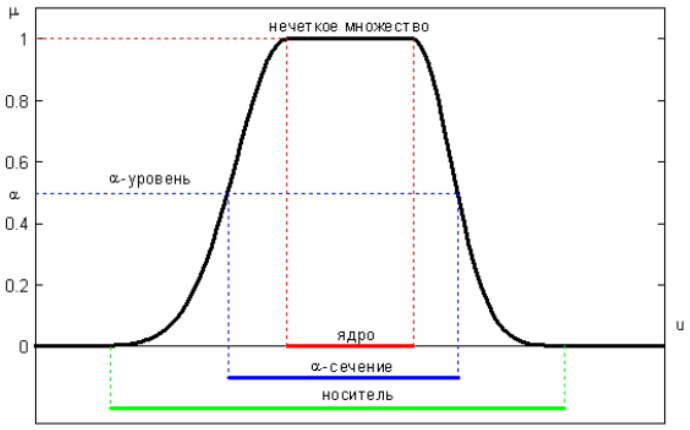

Рисунок 2.1 – Носитель, ядро и α-сечение

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: $X \overset{R}{\rightarrow} Y \overset{S}{\rightarrow} Z$. Тогда их композиция $T = R \circ S$ является нечетким отношением на $X \times Z$, таким что $X \overset{T}{\rightarrow} Z$.

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).

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

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

2.8 Нечеткие системы

Стандартная архитектура нечеткой системы включает три основные стадии: фаззификацию, нечеткий вывод, дефаззификацию. Архитектура нечеткой системы представлена на Рисунке 2.8.1.

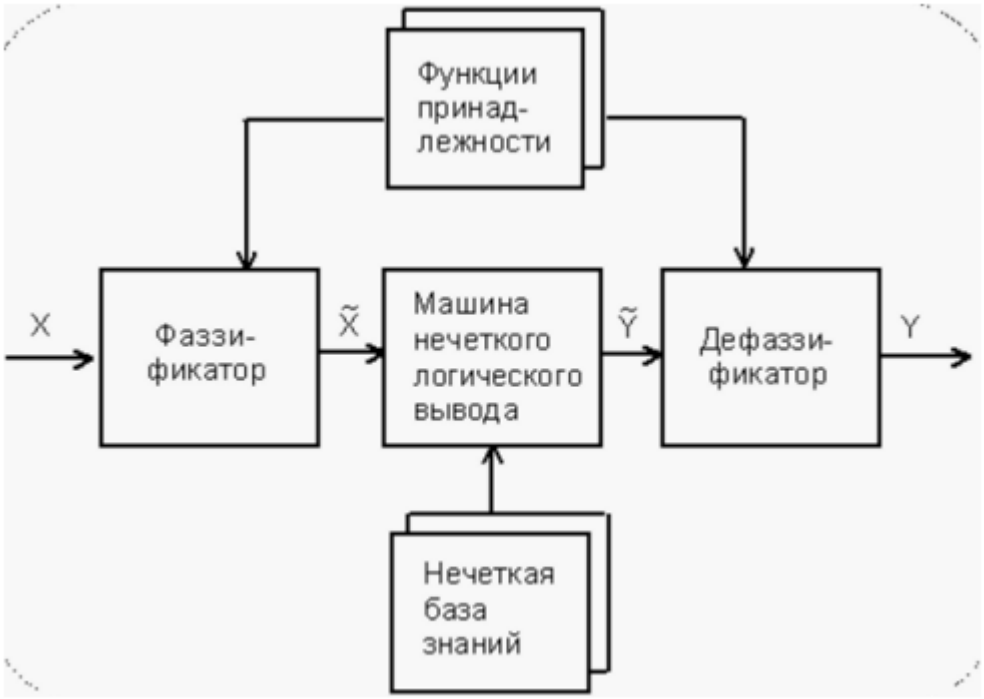

Рисунок 2.2 – Архитектура нечеткой логической системы

На Рисунке 2.2 приняты следующие обозначения:

- X— входной четкий вектор;
- $\tilde{X}$— вектор нечетких значений входов;
- $\tilde{Y}$— нечеткий результат вывода;
- Y — итоговый выход в четком виде.

2.9 Нечеткая база знаний

Нечеткая база знаний — это совокупность правил вида «Если посылка, то заключение». Посылка правила — всегда лингвистическое выражение: «x есть низкий», где низкий — терм, то есть заранее определённое нечеткое множество, заданное функцией принадлежности.

В реальной системе несколько входов объединяются с помощью логических операций «И» и «ИЛИ» (Формула 2.9).

$$A_1^{(i)}(x_1) \text{ И } A_2^{(i)}(x_2) \text{ И } \dots \text{ И } A_n^{(i)}(x_n) \rightarrow B^{(i)}(y) \tag{2.9}$$

Всю базу правил можно задать согласно Формуле 2.10.

$$\begin{gathered}\text{ЕСЛИ } A_1^{(1)} \wedge A_2^{(1)} \wedge \ldots \text{ТО } B^{(1)} \\ \text{ЕСЛИ } A_1^{(2)} \wedge A_2^{(2)} \wedge \ldots \text{ТО } B^{(2)} \\ \ldots \\ \text{ЕСЛИ } A_1^{(m)} \wedge A_2^{(m)} \wedge \ldots \text{ТО } B^{(m)}\end{gathered} \tag{2.10}$$

где m – количество правил;

$A_j^{(i)}$ – терм j-ой переменной в правиле i;

$B^{(i)}$ – заключение.

2.10 Фаззификация

Фаззификация — преобразование четкого входа x в значения функций принадлежности всех термов. Пусть имеется набор термов $A_1, A_2, \ldots, A_k$ соответствующей входной переменной x. Для конкретного входа $x = x_0$ вычисляются значения: $\mu_{A_1}(x_0)$, $\mu_{A_2}(x_0)$, …, $\mu_{A_k}(x_0)$. Получаем фаззифицированный набор: $\tilde{X} = \{(A_i, \mu_{A_1}(x_0))\}$.

2.11 Агрегирование условий правила

Каждое правило имеет посылку. Если оно простое: ЕСЛИ x есть A, то её истинность равна $\mu_A(x_0)$. Если посылка сложная: $A(x_1)$ И $B(x_2)$, то используется нечеткая операция конъюнкции (AND) (Формула 2.11).

$$\mu_{A \wedge B} = \min(\mu_A, \mu_B) \tag{2.11}$$

2.12 Активизация (импликация)

На этом этапе вычисляется степень истинности заключения правила.

Если степень истинности посылки равна α, то имплицированное заключение:

- метод минимума:

$$\mu_{B'}(y) = min(\alpha, \mu_B(y))$$

- метод произведения:

$$\mu_{B'}(y) = \alpha * \mu_B(y)$$

В классическом выводе Мамдани используется метод минимума.

2.13 Аккумуляция (объединение заключений правил)

Заключения всех правил для одной выходной переменной объединяются. Используется нечеткое объединение (Формула 2.12).

$$\mu_{B^{\backslash *}}(y) = \max_{i=1..m} \mu_{B_i'}(y) \tag{2.12}$$

Каждая «правило-кривая» добавляет вклад, финальное множество — их максимум.

2.14 Дефаззификация

На выходе после аккумуляции получается одно нечеткое множество $B^{\backslash *}$. Нужно преобразовать его в число. Основной метод использует центр тяжести фигуры (Формула 2.13).

$$y^{\backslash *} = \frac{\int y \mu_{B^{\backslash *}}(y) dy}{\int \mu_{B^{\backslash *}}(y) dy} \tag{2.13}$$

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 Проектирование системы нечеткого управления

В данном разделе подробно рассматривается реализация нечеткой экспертной системы, разработанной для оценки инвестиционной привлекательности игровых предметов (скинов) в виде логики Мамдани.

Реализация выполнена в файле main.py, содержание которого представлено в Приложении А.

3.3.1 Создание функций принадлежности

Перед определением лингвистических переменных в программе задаются две универсальные функции построения термов:

- треугольная функция принадлежности;
- трапециевидная функция принадлежности.

Эти функции позволяют компактно описывать все термы системы.

3.3.2 Определение лингвистических переменных

Для построения системы используется класс LinguisticVariable, который хранит:

- границы универсума (min, max);
- число точек дискретизации;
- набор термов (экземпляры FuzzySet);
- методы фаззификации и визуализации.

В системе определено четыре входные переменные и одна выходная.

Для каждой лингвистической переменной в разработанной системе предусмотрена возможность графического отображения её термов с помощью метода plot_terms, входящего в класс LinguisticVariable. Этот метод строит графики функций принадлежности всех нечётких множеств, определённых для данной переменной, на её универсуме.

В работе приведены графики для всех переменных: Wear, Liquidity, Price, Age, а также выходной переменной Investment.

Каждый из этих графиков иллюстрирует структуру разбиения универсума на пересекающиеся нечёткие множества.

Графическое отображение термов лингвистической переменной «Wear» представлено на Рисунке 3.3.1.

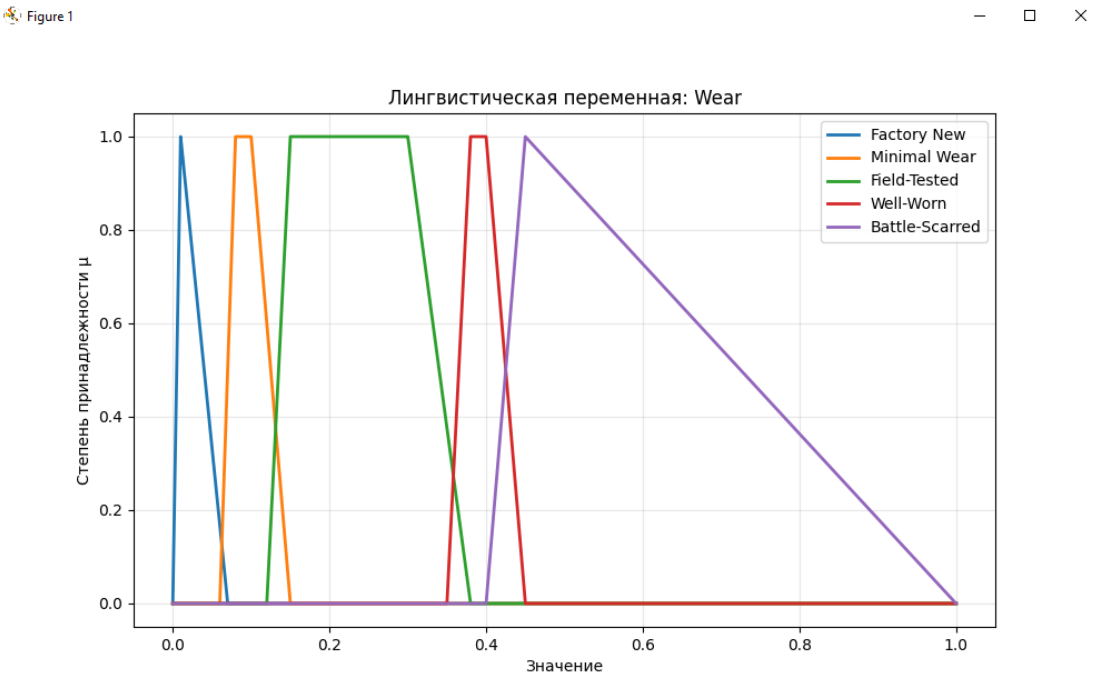

Рисунок 3.3.1 - Графическое отображение термов лингвистической переменной «Wear»

Графическое отображение термов лингвистической переменной «Liquidity» представлено на Рисунке 3.3.2.

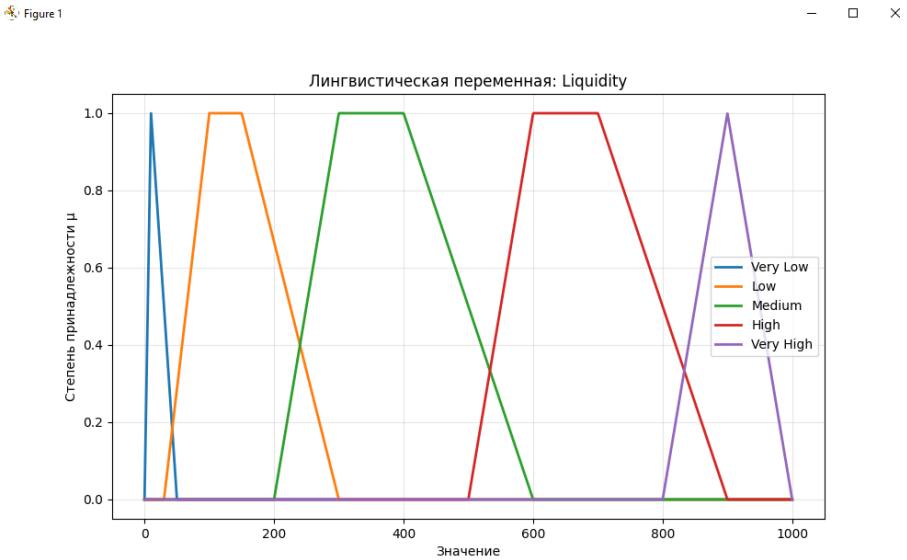

Рисунок 3.3.2 - Графическое отображение термов лингвистической переменной «Liquidity»

Графическое отображение термов лингвистической переменной «Price» представлено на Рисунке 3.3.3.

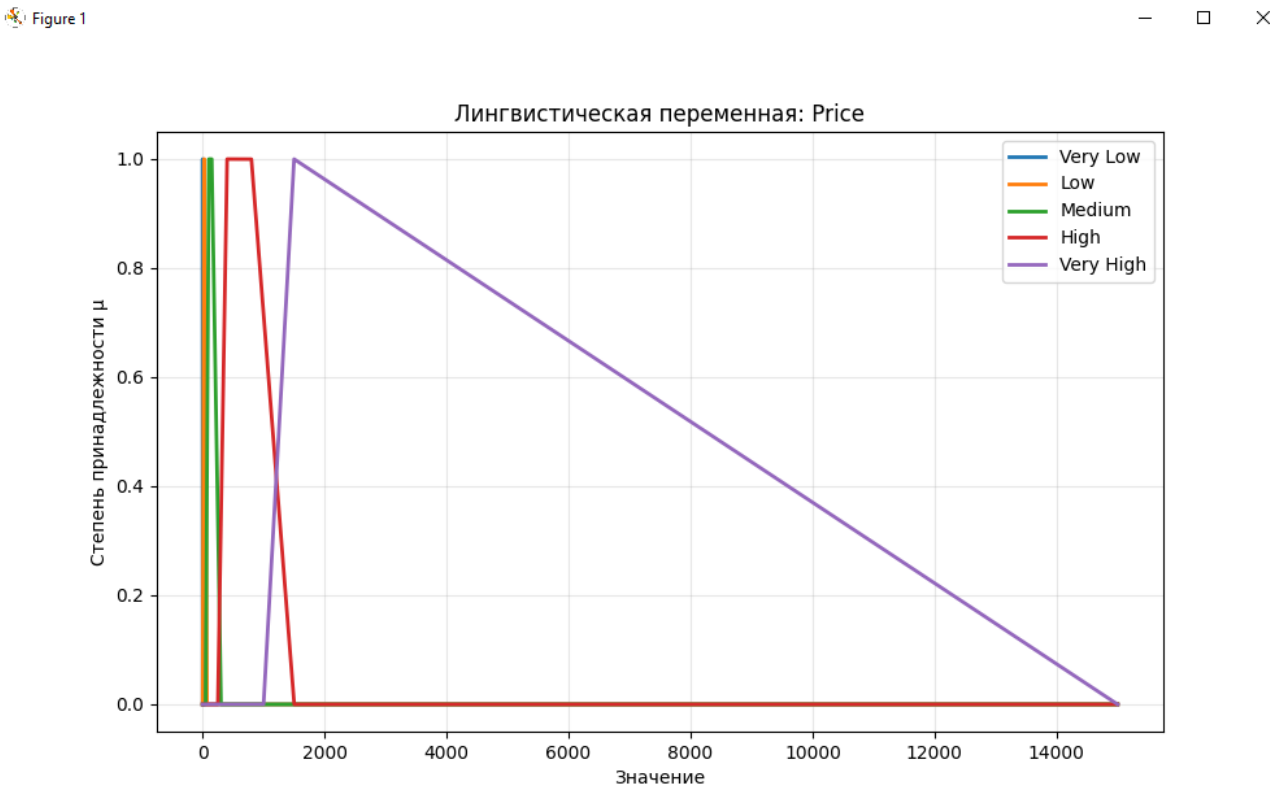

Рисунок 3.3.3 - Графическое отображение термов лингвистической переменной «Price»

Графическое отображение термов лингвистической переменной «Age» представлено на Рисунке 3.3.4.

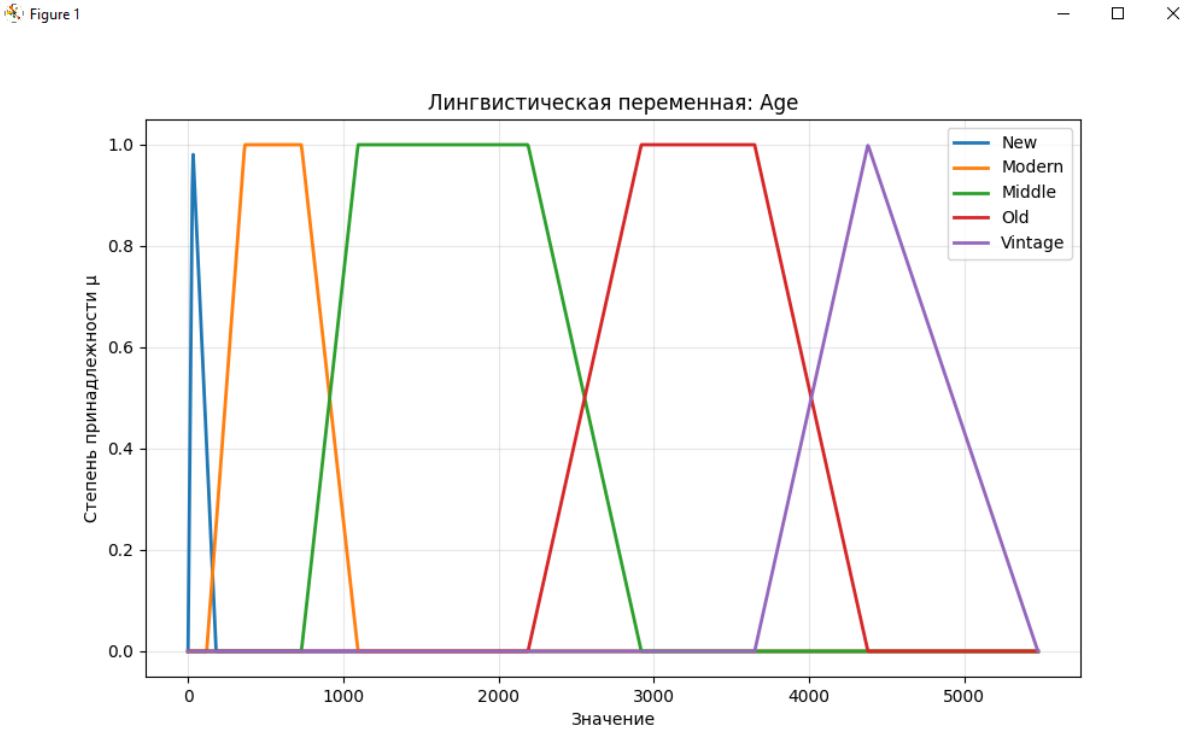

Рисунок 3.3.4 - Графическое отображение термов лингвистической переменной «Age»

Графическое отображение термов лингвистической переменной «Investment potential» представлено на Рисунке 3.3.5.

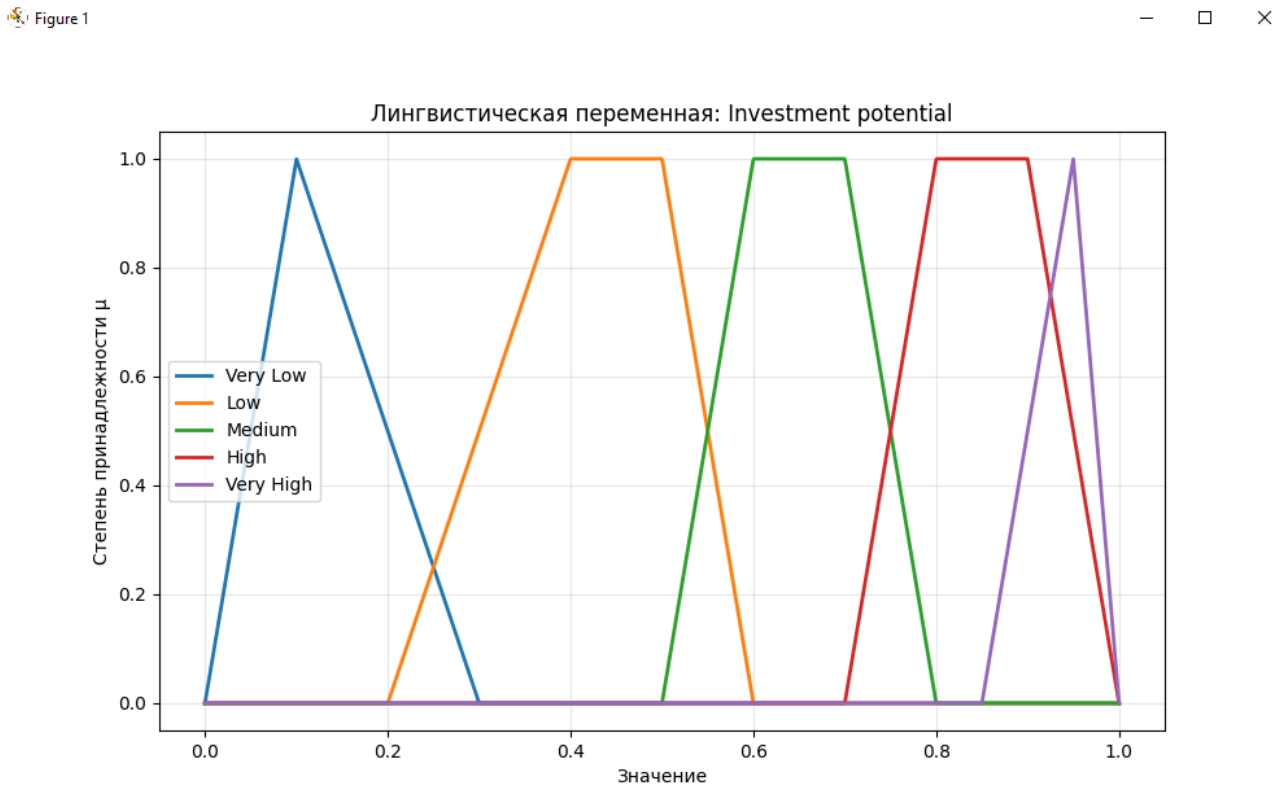

Рисунок 3.3.5 - Графическое отображение термов лингвистической переменной «Investment potential»

3.3.3 Построение базы правил

Логика системы задаётся через класс FuzzyRule. Каждое правило имеет вид, представленный Листингом 3.3.1.

Листинг 3.3.1 – Шаблон правила

```
IF (var₁ == term₁) AND (var₂ == term₂) ... THEN (Investment potential == termₖ)
```

Система поддерживает только конъюнкцию, степень истинности посылки вычисляется как минимум от функций принадлежности.

Создано 20 правил, отражающих интуитивные сценарии (Рисунок 3.3.6).

```
База правил (20 правил):
1. IF Wear == Factory New AND Liquidity == Very High AND Price == Very High AND Age == New THEN Investment potential == Very High
2. IF Wear == Factory New AND Liquidity == High AND Price == High THEN Investment potential == High
3. IF Wear == Minimal Wear AND Liquidity == Very High AND Price == High THEN Investment potential == High
4. IF Age == Vintage AND Price == Very High AND Liquidity == Medium THEN Investment potential == Very High
5. IF Wear == Field-Tested AND Liquidity == Medium AND Price == Medium THEN Investment potential == Medium
6. IF Wear == Factory New AND Liquidity == Medium AND Price == Medium THEN Investment potential == Medium
7. IF Age == Middle AND Price == Medium AND Liquidity == Medium THEN Investment potential == Medium
8. IF Price == Low AND Liquidity == High THEN Investment potential == High
9. IF Price == Low AND Liquidity == Medium THEN Investment potential == Medium
10. IF Price == Very Low AND Liquidity == High THEN Investment potential == Medium
11. IF Wear == Well-Worn AND Price == High AND Liquidity == Low THEN Investment potential == Low
12. IF Wear == Battle-Scarred AND Price == Very High AND Liquidity == Very Low THEN Investment potential == Very Low
13. IF Age == Old AND Price == High THEN Investment potential == High
14. IF Age == Old AND Liquidity == Low THEN Investment potential == Medium
15. IF Age == Vintage AND Wear == Factory New THEN Investment potential == Very High
16. IF Liquidity == Very Low THEN Investment potential == Very Low
17. IF Price == Very Low AND Liquidity == Low THEN Investment potential == Low
18. IF Wear == Battle-Scarred AND Liquidity == Medium THEN Investment potential == Low
19. IF Wear == Factory New AND Price == Low THEN Investment potential == High
20. IF Price == Very High AND Liquidity == Very High THEN Investment potential == Very High
```

Рисунок 3.3.6 – Созданные правила

Таким образом, сформирована база знаний системы.

3.3.4 Фаззификация входных данных

Для каждого входного параметра (wear, liquidity, price, age) вычисляется степень принадлежности ко всем термам. Вывод формируется в виде словаря (Листинг 3.3.2).

Листинг 3.3.2 – Словарь, полученный после фаззификации

```
{
    'Wear': {'Factory New': 0.12, 'Minimal Wear': 0.88, ...},
    'Price': {...},
    ...
}
```

Программа выводит только значения большие 0.01.

3.3.5 Таблица результатов

По завершении анализа всех образцов создаётся таблица (Рисунок 3.3.7).

Skin	Wear	Liquidity	Price	Age	Result
AK-47 | Redline	0.156	127	$32.99	11.8 years	0.5
AWP | Dragon Lore	0.035	1	$11850.00	11.4 years	0.148
M4A1-S | Hyper Beast	0.366	30	$128.75	10.6 years	0.139
Karambit | Fade	0.0102	5	$2350.00	12.3 years	0.139
Sport Gloves | Vice	0.09	3	$2400.00	7.8 years	0.143
Dreams & Nightmares Case	0	84118	$1.37	3.9 years	0.5

Рисунок 3.3.5 – Таблица с результатами оценки инвестиционной привлекательности

ЗАКЛЮЧЕНИЕ

В ходе выполненной работы разработана и исследована нечеткая система оценки инвестиционной привлекательности предметов игровой экономики на примере цифровых объектов из экосистемы Counter-Strike. Применение нечеткой логики позволило формализовать экспертные рассуждения и реализовать механизм плавного, интерпретируемого и адаптивного вывода.

В теоретической части работы рассмотрены ключевые понятия нечетких множеств, функции принадлежности, лингвистические переменные, правила вывода и основные этапы модели Мамдани. Особое внимание уделено операциям фазификации, агрегирования, импликации, аккумуляции и дефазификации, а также роли продукционных правил в формировании семантической связи между входными и выходными параметрами системы.

Практическая часть включала создание полнофункциональной программной реализации системы нечеткого управления. Были определены четыре входные лингвистические переменные — степень износа, ликвидность, рыночная цена и возраст предмета. Для каждой переменной сформированы терм-множества, задаваемые треугольными и трапециевидными функциями принадлежности. Выходной переменной выступила инвестиционная привлекательность, описанная аналогичным набором термов. На основе экспертных предположений сформирована база из двадцати продукционных правил, позволяющих учитывать различные комбинации характеристик и отражающих типичные закономерности рынка цифровых предметов.

Реализованный механизм нечёткого вывода по методу Мамдани позволил корректно вычислять итоговую оценку привлекательности, а визуализация функций принадлежности и результатов работы системы обеспечила наглядность поведения модели. Тестирование на наборе реальных игровых предметов показало, что система адекватно распознаёт логические зависимости, а также способна выдавать интерпретируемый результат в лингвистической форме.

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 main.py.

Приложение Б — Код файлы skins.py.

Приложение В — Код файлы relations.py.

Приложение Г — Код файлы linguistic_variable.py.

Приложение Д — Код файлы fuzzy_sets.py.

Приложение Е — Код файлы fuzzy_rules.py.

Приложение Ж — Код файлы fuzzy_control_system.py.

Приложение А

Код файлы main.py

Листинг А – Код файлы main.py

```
from linguistic_variable import LinguisticVariable
from fuzzy_sets import FuzzySet
from fuzzy_control_system import FuzzyControlSystem
from fuzzy_rules import FuzzyRule
from skins import create_sample_skins
import pandas as pd

def create_triangular(a: float, b: float, c: float):
    """Создает треугольную функцию принадлежности."""

    def mu(x: float) -> float:
        if x <= a:
            return 0.0
        elif a < x <= b:
            return (x - a) / (b - a)
        elif b < x <= c:
            return (c - x) / (c - b)
        else:
            return 0.0

    return mu

def create_trapezoidal(a: float, b: float, c: float, d: float):
    """Создает трапециевидную функцию принадлежности."""

    def mu(x: float) -> float:
        if x <= a:
            return 0.0
        elif a < x <= b:
            return (x - a) / (b - a)
        elif b < x <= c:
            return 1.0
        elif c < x <= d:
            return (d - x) / (d - c)
        else:
            return 0.0

    return mu

def setup_cs_fuzzy_system() -> FuzzyControlSystem:
    """
    Настраивает систему нечеткого управления с полным набором продукционных
правил.
    """
    system = FuzzyControlSystem()

    # 1. Степень износа
    wear_var = LinguisticVariable("Wear", 0.0, 1.0, 10000)
    wear_var.add_term(FuzzySet("Factory New", create_triangular(0.0, 0.01,
0.07)))
    wear_var.add_term(
        FuzzySet("Minimal Wear", create_trapezoidal(0.06, 0.08, 0.10, 0.15))
    )
    wear_var.add_term(
        FuzzySet("Field-Tested", create_trapezoidal(0.12, 0.15, 0.30, 0.38))
    )
```

Продолжение Листинга А

```
    wear_var.add_term(
        FuzzySet("Well-Worn", create_trapezoidal(0.35, 0.38, 0.40, 0.45))
    )
    wear_var.add_term(
        FuzzySet("Battle-Scarred", create_triangular(0.40, 0.45, 1.0))
    )
    system.add_input_variable(wear_var)

    # 2. Ликвидность
    liquidity_var = LinguisticVariable("Liquidity", 0, 1000, 1000)
    liquidity_var.add_term(FuzzySet("Very Low", create_triangular(0, 10, 50)))
    liquidity_var.add_term(FuzzySet("Low", create_trapezoidal(30, 100, 150,
300)))
    liquidity_var.add_term(FuzzySet("Medium", create_trapezoidal(200, 300, 400,
600)))
    liquidity_var.add_term(FuzzySet("High", create_trapezoidal(500, 600, 700,
900)))
    liquidity_var.add_term(FuzzySet("Very High", create_triangular(800, 900,
1000)))
    system.add_input_variable(liquidity_var)

    # 3. Рыночная цена
    price_var = LinguisticVariable("Price", 0, 15000, 150000)
    price_var.add_term(FuzzySet("Very Low", create_triangular(0, 0.5, 3)))
    price_var.add_term(FuzzySet("Low", create_trapezoidal(1, 21, 30, 70)))
    price_var.add_term(FuzzySet("Medium", create_trapezoidal(50, 100, 150,
300)))
    price_var.add_term(FuzzySet("High", create_trapezoidal(250, 400, 800,
1500)))
    price_var.add_term(FuzzySet("Very High", create_triangular(1000, 1500,
15000)))
    system.add_input_variable(price_var)

    # 4. Возраст
    max_days = 15 * 365
    age_var = LinguisticVariable("Age", 0.0, float(max_days), num_points=1000)
    age_var.add_term(FuzzySet("New", create_triangular(0.0, 30.0, 180.0)))
    age_var.add_term(FuzzySet("Modern", create_trapezoidal(120.0, 365.0, 730.0,
1095.0)))
    age_var.add_term(FuzzySet("Middle", create_trapezoidal(730.0, 1095.0,
2190.0, 2920.0)))
    age_var.add_term(FuzzySet("Old", create_trapezoidal(2190.0, 2920.0, 3650.0,
4380.0)))
    age_var.add_term(FuzzySet("Vintage", create_triangular(3650.0, 4380.0,
float(max_days))))
    system.add_input_variable(age_var)

    # Инвестиционная привлекательность [0.0, 1.0] - Выходная переменная
    investment_var = LinguisticVariable("Investment potential", 0.0, 1.0, 1000)
    investment_var.add_term(FuzzySet("Very Low", create_triangular(0.0, 0.1,
0.3)))
    investment_var.add_term(FuzzySet("Low", create_trapezoidal(0.2, 0.4, 0.5,
0.6)))
    investment_var.add_term(FuzzySet("Medium", create_trapezoidal(0.5, 0.6, 0.7,
0.8)))
    investment_var.add_term(FuzzySet("High", create_trapezoidal(0.7, 0.8, 0.9,
1.0)))
    investment_var.add_term(
        FuzzySet("Very High", create_triangular(0.85, 0.95, 1.0))
    )
    system.add_output_variable(investment_var)
```

Продолжение Листинга А

```
## ======= ПРАВИЛА ============

rules = []

# 1
r = FuzzyRule()
r.add_condition("Wear", "Factory New", "AND")
r.add_condition("Liquidity", "Very High", "AND")
r.add_condition("Price", "Very High", "AND")
r.add_condition("Age", "New", "AND")
r.set_conclusion("Investment potential", "Very High")
rules.append(r)

# 2
r = FuzzyRule()
r.add_condition("Wear", "Factory New", "AND")
r.add_condition("Liquidity", "High", "AND")
r.add_condition("Price", "High", "AND")
r.set_conclusion("Investment potential", "High")
rules.append(r)

# 3
r = FuzzyRule()
r.add_condition("Wear", "Minimal Wear", "AND")
r.add_condition("Liquidity", "Very High", "AND")
r.add_condition("Price", "High", "AND")
r.set_conclusion("Investment potential", "High")
rules.append(r)

# 4
r = FuzzyRule()
r.add_condition("Age", "Vintage", "AND")
r.add_condition("Price", "Very High", "AND")
r.add_condition("Liquidity", "Medium", "AND")
r.set_conclusion("Investment potential", "Very High")
rules.append(r)

# 5
r = FuzzyRule()
r.add_condition("Wear", "Field-Tested", "AND")
r.add_condition("Liquidity", "Medium", "AND")
r.add_condition("Price", "Medium", "AND")
r.set_conclusion("Investment potential", "Medium")
rules.append(r)

# 6
r = FuzzyRule()
r.add_condition("Wear", "Factory New", "AND")
r.add_condition("Liquidity", "Medium", "AND")
r.add_condition("Price", "Medium", "AND")
r.set_conclusion("Investment potential", "Medium")
rules.append(r)

# 7
r = FuzzyRule()
r.add_condition("Age", "Middle", "AND")
r.add_condition("Price", "Medium", "AND")
r.add_condition("Liquidity", "Medium", "AND")
r.set_conclusion("Investment potential", "Medium")
rules.append(r)
```

Продолжение Листинга А

```
# 8
r = FuzzyRule()
r.add_condition("Price", "Low", "AND")
r.add_condition("Liquidity", "High", "AND")
r.set_conclusion("Investment potential", "High")
rules.append(r)

# 9
r = FuzzyRule()
r.add_condition("Price", "Low", "AND")
r.add_condition("Liquidity", "Medium", "AND")
r.set_conclusion("Investment potential", "Medium")
rules.append(r)

# 10
r = FuzzyRule()
r.add_condition("Price", "Very Low", "AND")
r.add_condition("Liquidity", "High", "AND")
r.set_conclusion("Investment potential", "Medium")
rules.append(r)

# 11
r = FuzzyRule()
r.add_condition("Wear", "Well-Worn", "AND")
r.add_condition("Price", "High", "AND")
r.add_condition("Liquidity", "Low", "AND")
r.set_conclusion("Investment potential", "Low")
rules.append(r)

# 12
r = FuzzyRule()
r.add_condition("Wear", "Battle-Scarred", "AND")
r.add_condition("Price", "Very High", "AND")
r.add_condition("Liquidity", "Very Low", "AND")
r.set_conclusion("Investment potential", "Very Low")
rules.append(r)

# 13
r = FuzzyRule()
r.add_condition("Age", "Old", "AND")
r.add_condition("Price", "High", "AND")
r.set_conclusion("Investment potential", "High")
rules.append(r)

# 14
r = FuzzyRule()
r.add_condition("Age", "Old", "AND")
r.add_condition("Liquidity", "Low", "AND")
r.set_conclusion("Investment potential", "Medium")
rules.append(r)

# 15
r = FuzzyRule()
r.add_condition("Age", "Vintage", "AND")
r.add_condition("Wear", "Factory New", "AND")
r.set_conclusion("Investment potential", "Very High")
rules.append(r)

# 16
r = FuzzyRule()
```

Продолжение Листинга А

```
    r.add_condition("Liquidity", "Very Low", "AND")
    r.set_conclusion("Investment potential", "Very Low")
    rules.append(r)

    # 17
    r = FuzzyRule()
    r.add_condition("Price", "Very Low", "AND")
    r.add_condition("Liquidity", "Low", "AND")
    r.set_conclusion("Investment potential", "Low")
    rules.append(r)

    # 18
    r = FuzzyRule()
    r.add_condition("Wear", "Battle-Scarred", "AND")
    r.add_condition("Liquidity", "Medium", "AND")
    r.set_conclusion("Investment potential", "Low")
    rules.append(r)

    # 19
    r = FuzzyRule()
    r.add_condition("Wear", "Factory New", "AND")
    r.add_condition("Price", "Low", "AND")
    r.set_conclusion("Investment potential", "High")
    rules.append(r)

    # 20
    r = FuzzyRule()
    r.add_condition("Price", "Very High", "AND")
    r.add_condition("Liquidity", "Very High", "AND")
    r.set_conclusion("Investment potential", "Very High")
    rules.append(r)

    # Добавление всех правил в систему
    for r in rules:
        system.add_rule(r)

    return system

def demonstrate_rule_evaluation(system: FuzzyControlSystem):
    skins = create_sample_skins()

    print(f"\nСоздано {len(skins)} скинов:")
    for i, skin in enumerate(skins, 1):
        print(f"{i}. {skin}")

    summary_data = []

    for skin in skins:
        print(f"\nАнализ скина: {skin.name}")

        inputs = {
            "Wear": skin.float_value,
            "Liquidity": skin.liquidity,
            "Price": skin.price,
            "SkinAge": skin.age_days
        }

        print(f"Параметры скина:")
        print(f"    - Степень износа (float): {skin.float_value:.4f}")
        print(f"    - Ликвидность: {skin.liquidity} транзакций/день")
        print(f"    - Цена: ${skin.price:.2f}")
```

Продолжение Листинга А

```
        print(f"    - Возраст: {skin.age_days} дней ({skin.age_days/365:.1f}
лет)")

        # Фазификация
        fuzzified = system.fuzzify(inputs)
        print(f"Фазификация входных параметров:")

        for var_name, terms in fuzzified.items():
            print(f"    {var_name}:")
            for term_name, degree in terms.items():
                if degree > 0.01:
                    print(f"    - {term_name}: {degree:.3f}")

        # Вывод по Мамдани
        result, universe, aggregated = system.infer_mamdani(
            inputs, "Investment potential"
        )

        print(f"Результат нечеткого вывода: {result:.3f}")

        # Определяем лингвистическое значение
        investment_var = system.output_vars["Investment potential"]

        # Находим термы с наибольшей степенью принадлежности
        fuzzified_investment = {}
        for term_name, term_set in investment_var.terms.items():
            degree = term_set.mu(result)
            if degree > 0.01:
                fuzzified_investment[term_name] = degree

        if fuzzified_investment:
            sorted_terms = sorted(fuzzified_investment.items(),
                                  key=lambda x: x[1], reverse=True)

            print(f"Лингвистическая интерпретация:")
            for term_name, degree in sorted_terms:
                print(f"    - {term_name}: {degree:.3f}")

        summary_data.append({
            "Skin": skin.name,
            "Wear": f"{skin.float_value:.4f}",
            "Liquidity": f"{skin.liquidity:.0f}",
            "Price": f"${skin.price:.2f}",
            "Age": f"{skin.age_days/365:.1f} years",
            "Result": f"{result:.3f}",
        })

    df = pd.DataFrame(summary_data)

    print(df.to_markdown(index=False))

    return df

def main():
    system = setup_cs_fuzzy_system()
    system.print_system_info()

    for var_name, var in system.input_vars.items():
        var.plot_terms(f"Лингвистическая переменная: {var_name}")

    for var_name, var in system.output_vars.items():
```

Окончание Листинга А

```
        var.plot_terms(f"Лингвистическая переменная: {var_name}")

    demonstrate_rule_evaluation(system)

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

Приложение Б

Код файлы skins.py

Листинг Б – 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),
```

Продолжение Листинга Б

```
        ),
        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),
        ),
        Skin(
            name="Dreams & Nightmares Case",
            float_value=0,
            liquidity=84118,
            price=1.37,
            age_days=1411,
            paint_seed=0
        )
    ]
```

Приложение В

Код файлы relations.py

Листинг В – Код файлы relations.py

```
from typing import List

class FuzzyRelation:
    """
    Класс для представления нечёткого 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."""
        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,
```

Окончание Листинга В

```
            matrix=complemented_matrix,
            name=f"Дополнение: {self.name}",
        )
```

Приложение Г

Код файлы linguistic_variable.py

Листинг Г – Код файлы linguistic_variable.py

```
from typing import Dict, List
from fuzzy_sets import FuzzySet
import numpy as np
import matplotlib.pyplot as plt

class LinguisticVariable:
    """
    Лингвистическая переменная: универсум (min,max), дискретизация
    и набор термов (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())})"

    def fuzzify_crisp(self, crisp_value: float) -> Dict[str, float]:
        """
        Фазификация четкого значения.
        Возвращает словарь {термин: степень принадлежности} для заданного
четкого значения.
        """
        result = {}
        for term_name, fuzzy_set in self.terms.items():
            result[term_name] = fuzzy_set.mu(crisp_value)
        return result
```

Окончание Листинга Г

```
    def plot_terms(self, title: str = None):
        """Визуализация всех терминов лингвистической переменной"""
        x = self.universe()
        plt.figure(figsize=(10, 6))

        for term_name, term_set in self.terms.items():
            y = [term_set.mu(xi) for xi in x]
            plt.plot(x, y, label=term_name, linewidth=2)

        plt.xlabel("Значение")
        plt.ylabel("Степень принадлежности μ")
        plt.title(title or f"Термы лингвистической переменной '{self.name}'")
        plt.legend()
        plt.grid(True, alpha=0.3)
        plt.show()

    def get_term_names(self) -> List[str]:
        """Возвращает список имен терминов."""
        return list(self.terms.keys())

    def get_term(self, term_name: str) -> FuzzySet:
        """Возвращает нечеткое множество для указанного термина."""
        return self.terms[term_name]
```

Приложение Д

Код файлы fuzzy_sets.py

Листинг Д – Код файлы fuzzy_sets.py

```
import matplotlib.pyplot as plt
from typing import Dict, Union, Callable, Optional, Tuple
import numpy as np
import matplotlib.pyplot as plt

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":
```

Продолжение Листинга Д

```
            print(f"Множество {self.name} не является дискретным для
визуализации")
            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:
            Нечёткое 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 нечёткого 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)

    def plot_continuous(
        self, x_range: Tuple[float, float] = None, num_points: int = 200
    ):
        """Визуализация непрерывной функции принадлежности."""
        if self.data_type != "continuous":
            print(f"Множество {self.name} не является непрерывным")
```

Окончание Листинга Д

```
            return

        if x_range is None:
            x_min, x_max = -10, 10
        else:
            x_min, x_max = x_range

        x = np.linspace(x_min, x_max, num_points)
        y = [self.mu(xi) for xi in x]

        plt.figure(figsize=(10, 5))
        plt.plot(x, y, "b-", linewidth=2)
        plt.fill_between(x, 0, y, alpha=0.3)
        plt.xlabel("x")
        plt.ylabel("μ(x)")
        plt.title(f"Функция принадлежности: {self.name}")
        plt.grid(True, alpha=0.3)
        plt.ylim(0, 1.1)
        plt.show()
```

Приложение Е

Код файлы fuzzy_rules.py

Листинг Е – Код файлы fuzzy_rules.py

```
from typing import List, Tuple, Dict, Any
import numpy as np

class FuzzyRule:
    """
    Класс для представления одного нечёткого правила.
    Формат: ЕСЛИ (условие1 И условие2 ...) ТО (заключение)
    """

    def __init__(self, name: str = "Неименованное правило"):
        self.name = name
        self.conditions: List[Tuple[str, str, str]] = (
            []
        )  # (var_name, term_name, connective)
        self.conclusion: Tuple[str, str] = None  # (var_name, term_name)

    def add_condition(self, var_name: str, term_name: str, connective: str =
"AND"):
        """
        Добавляет условие в правило.

        Args:
            var_name: имя лингвистической переменной
            term_name: имя термина
            connective: "AND"
        """
        self.conditions.append((var_name, term_name, connective))

    def set_conclusion(self, var_name: str, term_name: str):
        """Устанавливает заключение правила."""
        self.conclusion = (var_name, term_name)

    def evaluate(self, fuzzified_inputs: Dict[str, Dict[str, float]]) -> float:
        """
        Вычисляет степень активации правила на основе фазифицированных входов.

        Args:
            fuzzified_inputs: словарь {var_name: {term_name: degree}}

        Returns:
            Степень активации правила (0-1)
        """
        if not self.conditions:
            return 0.0

        activation = 1.0
        for var_name, term_name, connective in self.conditions:
            if connective == "AND":
                if (
                    var_name in fuzzified_inputs
                    and term_name in fuzzified_inputs[var_name]
                ):
                    activation = min(activation,
fuzzified_inputs[var_name][term_name])
                else:
                    activation = 0.0
```

Продолжение Листинга Е

```
        return activation

    def __str__(self):
        conditions_str = " AND ".join(
            [f"{var} == {term}" for var, term, _ in self.conditions]
        )
        conclusion_str = (
            f"{self.conclusion[0]} == {self.conclusion[1]}"
            if self.conclusion
            else "Нет заключения"
        )
        return f"IF {conditions_str} THEN {conclusion_str}"

class RuleBase:
    """База правил нечёткой системы."""

    def __init__(self):
        self.rules: List[FuzzyRule] = []

    def add_rule(self, rule: FuzzyRule):
        """Добавляет правило в базу."""
        self.rules.append(rule)

    def evaluate_all(
        self, fuzzified_inputs: Dict[str, Dict[str, float]]
    ) -> Dict[Tuple[str, str], float]:
        """
        Оценивает все правила и возвращает степени активации для каждого вывода.

        Returns:
            Словарь {(output_var, term): activation}
        """
        activations = {}

        for rule in self.rules:
            activation = rule.evaluate(fuzzified_inputs)
            if activation > 0 and rule.conclusion:
                key = (rule.conclusion[0], rule.conclusion[1])
                if key in activations:
                    activations[key] = max(activations[key], activation)
                else:
                    activations[key] = activation

        return activations

    def print_rules(self):
        """Выводит все правила."""
        print(f"\nБаза правил ({len(self.rules)} правил):")
        for i, rule in enumerate(self.rules, 1):
            print(f"{i}. {rule}")
```

Приложение Ж

Код файлы fuzzy_control_system.py

Листинг Ж – Код файлы fuzzy_control_system.py

```
import numpy as np
from typing import Dict, List, Tuple
from linguistic_variable import LinguisticVariable
from fuzzy_rules import FuzzyRule, RuleBase

class FuzzyControlSystem:
    """
    Система нечеткого управления с поддержкой:
    - фазификации
    - агрегации правил (Мамдани)
    - аккумуляции
    - дефазификации
    """

    def __init__(self):
        self.input_vars: Dict[str, LinguisticVariable] = {}
        self.output_vars: Dict[str, LinguisticVariable] = {}
        self.rule_base = RuleBase()

    def add_input_variable(self, var: LinguisticVariable):
        self.input_vars[var.name] = var

    def add_output_variable(self, var: LinguisticVariable):
        self.output_vars[var.name] = var

    def add_rule(self, rule: FuzzyRule):
        """Добавляет правило в систему."""
        self.rule_base.add_rule(rule)

    def fuzzify(self, input_values: Dict[str, float]) -> Dict[str, Dict[str,
float]]:
        """
        Фазификация всех входных значений.
        Возвращает словарь: {имя_переменной: {имя_терма:
степень_принадлежности}}
        """
        fuzzified = {}
        for var_name, value in input_values.items():
            if var_name in self.input_vars:
                var = self.input_vars[var_name]
                fuzzified[var_name] = {}
                for term_name, term_set in var.terms.items():
                    fuzzified[var_name][term_name] = term_set.mu(value)
        return fuzzified

    def infer_mamdani(
        self,
        input_values: Dict[str, float],
        output_var_name: str,
        num_points: int = 100,
    ) -> Tuple[float, np.ndarray, np.ndarray]:
        """
        Вывод по методу Мамдани.

        Steps:
        1. Фазификация входов
```

Окончание Листинга Ж

```
        2. Оценка правил
        3. Композиция (аккумуляция) выходных термов
        4. Дефазификация
        """
        # 1. Фазификация
        fuzzified_inputs = self.fuzzify(input_values)

        # 2. Оценка правил
        rule_activations = self.rule_base.evaluate_all(fuzzified_inputs)

        # 3. Композиция и аккумуляция
        output_var = self.output_vars[output_var_name]
        universe = np.linspace(output_var.domain_min, output_var.domain_max,
num_points)

        aggregated_output = np.zeros_like(universe, dtype=float)

        for (var_name, term_name), activation in rule_activations.items():
            if var_name == output_var_name and activation > 0:
                term_set = output_var.terms[term_name]

                # Получаем функцию принадлежности терма
                membership = np.array([term_set.mu(x) for x in universe])

                # "Обрезаем" функцию по степени активации (метод Мамдани)
                clipped = np.minimum(membership, activation)

                # Аккумулируем с помощью операции максимума
                aggregated_output = np.maximum(aggregated_output, clipped)

        # 4. Дефазификация (центр тяжести)
        if np.sum(aggregated_output) == 0:
            crisp_value = (output_var.domain_min + output_var.domain_max) / 2
        else:
            crisp_value = np.sum(universe * aggregated_output) / np.sum(
                aggregated_output
            )

        return crisp_value, universe, aggregated_output

    def print_system_info(self):
        print("\nВходные переменные:")
        for name, var in self.input_vars.items():
            print(f" {name}: [{var.domain_min:.2f}, {var.domain_max:.2f}]")
            print(f"    Термы: {', '.join(var.get_term_names())}")

        print("\nВыходные переменные:")
        for name, var in self.output_vars.items():
            print(f"  {name}: [{var.domain_min:.2f}, {var.domain_max:.2f}]")
            print(f"    Термы: {', '.join(var.get_term_names())}")

        self.rule_base.print_rules()
```