



МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт ИВТИ
Кафедра ВТ

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)**


Направление 09.03.01 Информатика и вычислительная техника
(код и наименование)

Образовательная программа Системы автоматизированного проектирования

Форма обучения очная
(очная/очно-заочная/заочная)

Тема: Изучение и программная реализация алгоритма RS1 построения обобщенных понятий на основе теории приближенных множеств

Студент А-06-19  Швец Г.В.
группа подпись фамилия и инициалы

Руководитель ВКР К.Т.Н. доцент  Фомина М.В.
уч. степень должность подпись фамилия и инициалы

организация

«Работа допущена к защите»

Заведующий кафедрой Д.Т.Н. профессор  Топорков В.В.
уч. степень звание подпись фамилия и инициалы

Дата

Москва, 2023



МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт	ИВТИ
Кафедра	ВТ


**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
(БАКАЛАВРСКУЮ РАБОТУ)**

Направление 09.03.01 Информатика и вычислительная техника
(код и наименование)

Образовательная программа Системы автоматизированного проектирования

Форма обучения очная
(очная/очно-заочная/заочная)

Тема: Изучение и программная реализация алгоритма RS1 построения обобщенных понятий на основе теории приближенных множеств

Студент	А-06-19		Швец Г.В.
	группа	подпись	фамилия и инициалы

Руководитель ВКР	к.т.н.	доцент		Фомина М.В.
	уч. степень	должность	подпись	фамилия и инициалы

	организация			
Заведующий кафедрой	д.т.н.	профессор		Топорков В.В.
	уч. степень	звание	подпись	фамилия и инициалы

Место выполнения работы ФГБОУ ВО «НИУ «МЭИ»

СОДЕРЖАНИЕ РАЗДЕЛОВ ЗАДАНИЯ И ИСХОДНЫЕ ДАННЫЕ

Целью работы является изучение возможностей построения обобщенных понятий с целью формирования модели поведения робота. Алгоритм RS1 представляет собой набор продукционных правил, позволяющих принять решение по выбору действия на основе анализа неполных, неточных и противоречивых данных. Необходимо:

- изучить алгоритм RS1
- обосновать выбор среды моделирования
- отладить программную систему
- разработать программный интерфейс
- проверить работоспособность программы на тестовых примерах
- дать анализ полученным результатам
- сформировать пояснительную записку и презентацию

ПЕРЕЧЕНЬ ГРАФИЧЕСКОГО МАТЕРИАЛА

Количество листов

Количество слайдов в презентации 12

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

Вагин В.Н., Головина Е.Ю., Загорянская А.А., Фомина М.В., Поспелов Д.А. «Достоверный и правдоподобный вывод в интеллектуальных системах», ООО «Физмалит», 2008, 978-9221-0962-8

Zdzislaw I. Pawlak «Rough sets: Theoretical aspects of reasoning about data», Kluwer Academic Publishers, 1991, 978-0792314727

Примечания:

1. Задание брошюруется вместе с выпускной работой после титульного листа (страницы задания имеют номера 2, 3).
2. Отзыв руководителя, рецензия(и), отчет о проверке на объем заимствований и согласие студента на размещение работы в открытом доступе вкладываются в конверт (файловую папку) под обложкой работы.

АННОТАЦИЯ

Целью данной выпускной квалификационной работы является изучение и программная реализация алгоритма RS1 построения обобщенных понятий на основе теории приближенных множеств с целью внедрения алгоритма в робототехнические комплексы и системы для возможности формирования модели их поведения. Дается подробный обзор алгоритма и той теории, на которой он основан, проводится сравнительный анализ его сильных и слабых сторон, а также дается описание программной реализации настоящего алгоритма, включая разработку для него программного интерфейса. Проводится тестирование на различных наборах данных из известных коллекций и анализ работы разработанного программного комплекса.

СОДЕРЖАНИЕ

АННОТАЦИЯ	4
СОДЕРЖАНИЕ	5
ВВЕДЕНИЕ	6
ГЛАВА 1. ИЗУЧЕНИЕ АЛГОРИТМА RS1, ПОСТРОЕННОГО НА ТЕОРИИ ПРИБЛИЖЕННЫХ МНОЖЕСТВ	8
1.1 ОСНОВНЫЕ ПОНЯТИЯ ТЕОРИИ ПРИБЛИЖЕННЫХ МНОЖЕСТВ	8
1.2 АЛГОРИТМ RS1, ОСНОВАННЫЙ НА ТЕОРИИ ПРИБЛИЖЕННЫХ МНОЖЕСТВ	14
1.3 ПСЕВДОКОД АЛГОРИТМА RS1	17
1.4 ПРЕИМУЩЕСТВА И НЕДОСТАТКИ ИСПОЛЬЗОВАНИЯ АЛГОРИТМА RS1	20
ГЛАВА 2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА RS1	22
2.1 ВЫБОР СРЕДЫ МОДЕЛИРОВАНИЯ И ЯЗЫКА ПРОГРАММИРОВАНИЯ	22
2.2 РАЗРАБОТКА МОДУЛЯ НА PYTHON, РЕАЛИЗУЮЩЕГО ОСНОВНЫЕ ПОЛОЖЕНИЯ АЛГОРИТМА RS1	24
2.3 РАЗРАБОТКА ПРОГРАММНОГО ИНТЕРФЕЙСА И ЕГО ОПИСАНИЕ	29
2.4 ОТЛАДКА ПРОГРАММНОГО КОМПЛЕКСА И ТЕСТИРОВАНИЕ	35
ГЛАВА 3. ТЕСТИРОВАНИЕ ПРОГРАММНОГО КОМПЛЕКСА НА БОЛЬШИХ ДАННЫХ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ	37
3.1 ТЕСТИРОВАНИЕ АЛГОРИТМА НА БОЛЬШИХ ДАННЫХ	37
3.2 АНАЛИЗ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ	43
ЗАКЛЮЧЕНИЕ	46
СПИСОК ЛИТЕРАТУРЫ	47
ПРИЛОЖЕНИЕ	48

ВВЕДЕНИЕ

В современном мире, где автоматизация и роботизация становятся все более распространенными, разработка эффективных алгоритмов для управления робототехническими комплексами становится всё более актуальной задачей. Именно поэтому в данной бакалаврской дипломной работе исследуется алгоритм RS1, который был разработан на основе теории приближенных множеств, основанной знаменитым польским математиком и ученым Здиславом Павлаком. Этот алгоритм мог бы быть применен для построения обобщенных моделей поведения робототехнических комплексов. Работа предусматривает также и программную реализацию данного алгоритма, что позволит провести экспериментальное исследование его эффективности и точности в задачах моделирования поведения робототехнического комплекса. Результаты данной работы могут быть в дальнейшем использованы в качестве основы для разработки новых алгоритмов и методов, а также для улучшения уже существующих систем управления робототехническими комплексами.

В настоящее время повсеместно разрабатываются и используются различные системы принятия решений, целью которых является автоматизация многих процессов, начиная от бытовых, заканчивая производственными, управленческими и т.д., что в свою очередь приводит к следующим положительным чертам: сокращению рутинных задач для человека, удешевлению работы, росту производительности, исключению ошибок, совершенных из-за человеческого фактора, увеличению точности и стабильности. В подобных системах крайне важной частью является построение обобщенных моделей на основе обработки огромных массивов данных, поступающих на вход для анализа. Источниками этих данных могут являться такие области как: математика, физика, медицина, биология, экономика и финансы, космонавтика, производственные конвейерные процессы, атомная, и в целом, вся энергетическая промышленность, геоинформатика, криптография, машинное обучение и многое другое. Естественно, обычный человек не сможет

выполнить анализ и построить систему решений для столь большого количества информации быстро и точно. Развитие и использование таких систем принятия решений становится все более значимым в современном мире, поскольку они могут помочь в решении сложных проблем, связанных с прогнозированием, оптимизацией и управлением процессами в различных областях. Однако для того, чтобы эти системы работали эффективно, необходимо разрабатывать и совершенствовать алгоритмы, которые позволят обрабатывать и анализировать большие объемы данных, а также принимать правильные решения на основе этих данных. Именно поэтому разработка и исследование новых алгоритмов, таких как алгоритм RS1, является важным направлением в науке и информационных технологиях.

ГЛАВА 1. ИЗУЧЕНИЕ АЛГОРИТМА RS1, ПОСТРОЕННОГО НА ТЕОРИИ ПРИБЛИЖЕННЫХ МНОЖЕСТВ

1.1 ОСНОВНЫЕ ПОНЯТИЯ ТЕОРИИ ПРИБЛИЖЕННЫХ МНОЖЕСТВ

Теория приближенных, или, как её еще называют, грубых, множеств (Rough Sets Theory) была впервые разработана и предложена в 1982 году известным польским ученым и математиком Здиславом Павлаком в качестве принципиально нового инструмента, который позволяет работать с нечеткостями [2]. Главной идеей, вокруг которой была построена данная теория, является идея классификации объектов. К отличительным особенностям разработанной теории можно отнести то, что для её применения не требуется никакой предварительной или же дополнительной информации о наборе данных, к которому она применяется. С её помощью возможно решить проблему неточности, нечёткости, противоречивости или схожести знаний в различных больших массивах данных. Данные знания довольно с большой точностью можно классифицировать с помощью таких понятий, как верхнее и нижнее приближения, которые являются фундаментальными атрибутами теории приближенных множеств. Дадим четкие определения данным атрибутам. Нижнее приближение (Lower approximation) – это множество, включающее в себя все те объекты набора данных, которые с наибольшей вероятностью будут принадлежать заданному понятию. Верхнее приближение (Upper approximation) – это множество, содержащее в себе объекты, которые точно нельзя отнести к заданному понятию. В результате разницы между нижним и верхним приближениями образуется некая область, которая будет содержать в себе объекты, для которых невозможно точно определить их классификацию на основе имеющейся информации о них [1]. Данная область называется граничной (Boundry region). Теперь немного глубже рассмотрим саму теорию и её структуру.

Положим, что на некотором множестве U установлено отношение $\tilde{R} \subseteq U \times U$. Данное отношение будет означать неразличимость или же эквивалентность произвольных объектов на установленном множестве U . Примем $F = (U, \tilde{R})$ в качестве пространства аппроксимации или пространства приближений, поскольку данные понятия являются эквивалентными. Пусть, если (x, y) принадлежат отношению эквивалентности, то x и y неразличимы в F по значениям из множества решающих атрибутов D . Классы эквивалентности, применяемые по отношению к \tilde{R} , именуются элементарными множествами или атомами в F . Определим все множество классов эквивалентности как $R^{\sim}(D) = \{a_1, a_2, \dots, a_n\}$. Стоит к тому же заметить, что всякое пространство приближений также включает пустое множество в качестве элементарного. Множество, полученное в результате конечного объединения элементарных множеств в пространстве приближений F , будем называть составным множеством в F , а $Def(F)$ – семейством составных множеств в F .

Скажем, что $X \subseteq U$. Под нижним приближением множества X в F примем максимальное составное множество в F , которое является подмножеством X . А под верхним приближением X в F примем минимальное составное множество в F , которое содержит X . Обозначим верхнее и нижнее приближения множества X в пространстве F соответственно через $\underline{F}X$ и $\overline{F}X$. Множество $Boundry_F(X) = \underline{F}X - \overline{F}X$ именуем в качестве границы X в пространстве приближений F . Определим нижнее приближение как $\underline{F}X = Lower_F(X)$. Верхнее приближение обозначим как $\overline{F}X = Upper_F(X)$. Полученная пара $\langle Lower(x), Upper(X) \rangle$ будет являться приближенным множеством.

Для множества X в пространстве приближений $F = (O, \tilde{R})$ можно определить меру точности аппроксимации. Стоит заметить, что эта мера определяется только для конечных множеств и вычисляется следующим образом:

$$\mu_F(X) = \frac{|Lower(X)|}{|Upper(X)|},$$

где $|X|$ является мощностью множества X .

Также заметим, что справедливо $0 \leq \mu_P(X) \leq 1$, и $\mu_P(X) = 1$ в случае, если X определяемо в пространстве аппроксимаций. В противном случае, если X не определяемо в пространстве аппроксимаций, то $\mu_P(X) < 1$.

Представим изложенную выше теорию на примере.

Пример 1. Допустим, что у нас есть набор данных, описывающий взаимосвязь уровня образования человека и его дохода. Набор данных представлен в Таблице 1.

Таблица 1. Пример набора данных

Имя	Уровень Образования	Высокий доход
Андрей	Среднее	Нет
Василиса	Среднее	Да
Василий	Неоконченное высшее	Нет
Владимир	Высшее	Да
Екатерина	Аспирантура	Да

Положим, что множество решающих атрибутов $D = \{\text{Уровень образования}\}$.

Исходя из этого, мы имеем четыре класса эквивалентности:

$$R^{\sim}(X) = \{\{\text{Андрей, Василиса}\}, \{\text{Василий}\}, \{\text{Владимир}\}, \{\text{Екатерина}\}\}$$

В данной ситуации $a_1 = \{\text{Андрей, Василиса}\}$, $a_2 = \{\text{Василий}\}$, $a_3 = \{\text{Владимир}\}$, $a_4 = \{\text{Екатерина}\}$.

Множество, которое может быть получено в результате объединений двух или более элементарных множеств, будем называть составным множеством. Таким образом, например, множество $\{e_2, e_3\} = \{\text{Василий, Владимир}\}$ можно принять за составное.

Допустим, что X – класс людей с высоким доходом. В результате получаем, что $\underline{F}X = \{\text{Владимир, Екатерина}\}$, а $\overline{F}X = \{\text{Владимир, Екатерина, Андрей, Василиса}\}$.

Приближенные множества могут применяться для классификации данных нам объектов по следующему принципу.

Примем $POS(X) = \underline{F}X$ за положительную область, $NEG(X) = S \setminus \overline{F}X$ за отрицательную область, а все то, что не вошло ни в одну из вышеперечисленных областей $BND(X) = \overline{F}(X) / \underline{F}(X)$ за граничную область, образованную в результате разницы положительной и отрицательной. Теперь сформируем следующие правила:

$POS(X) \rightarrow X$

$NEG(X) \rightarrow \bar{X}$

$BND(X) \rightarrow \text{возможно } X,$

где описание множества представлено в виде наборов атрибутов. Рассматривая наш пример, мы в результате получим следующие классификационные правила:

- 1) (Уровень образования = Высшее) \vee (Уровень образования = Аспирантура) \Rightarrow Высокий доход
- 2) (Уровень образования = Неполное высшее) \Rightarrow Не высокий доход
- 3) (Уровень образования = Среднее) \Rightarrow Возможно высокий доход

Изобразим приближенное множество в графическом виде (см. рисунок 1.).

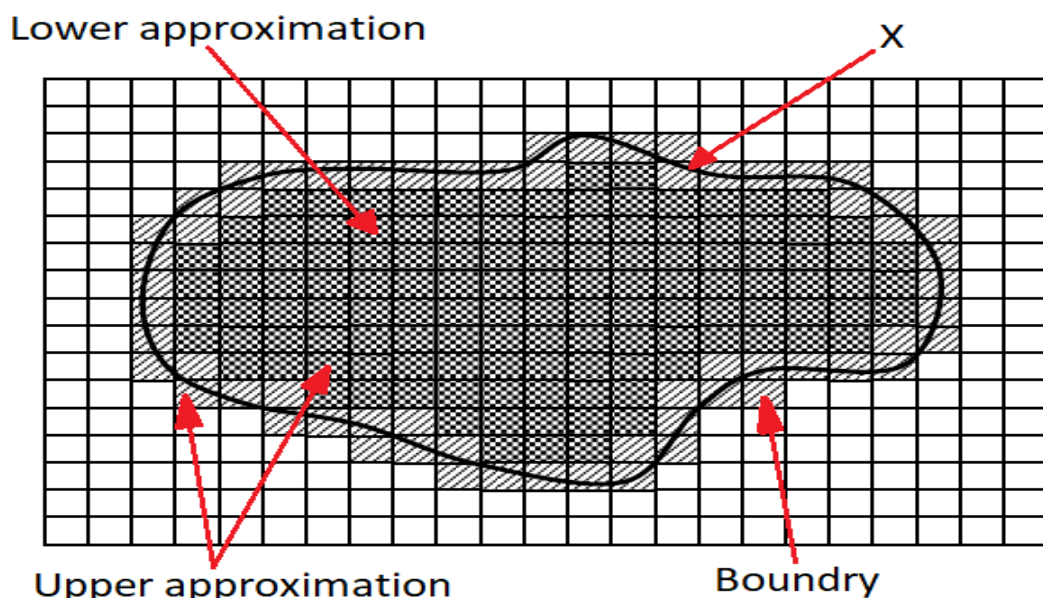


Рисунок 1. Представление приближенного множества в графическом виде.

На рисунке 1 сплошной линией представлено все множество X . Нижнее приближение, включающее в себя все те объекты множества X , которые точно можно отнести к заданному понятию, представлено в виде области, закрашенной квадратами. Верхнее приближение, включающее в себя объекты, возможно принадлежащие X , изображены как поля, закрашенные квадратами и косой линией. Граничный регион представляет же собой поля, заштрихованные косой линией. Поля, которые никак не заполнены являются отрицательными объектами.

Разберем так же следующие ключевые свойства приближенных множеств. Как уже было введено ранее, $\tilde{R} \subseteq X \times X$ является отношением неразличимости или эквивалентности на X . Упорядоченная пара $F = (X, \tilde{R})$ представляет собой пространство приближений. Теперь обратим наше внимание на множества $Y \subset X$, $Z \subset X$.

Данные множества X и Z приближенно равны сверху тогда и только тогда, когда совпадают их верхние приближения, что означает $\overline{F}Y = \overline{F}Z$. Обозначим это отношение как $Y \simeq Z$. То же самое определим и для нижнего приближения. Два множества X и Z приближенно равны снизу в пространстве приближений тогда и только тогда, когда совпадают их нижние приближения, что означает $\underline{F}Y = \underline{F}Z$. Обозначим это как $F \approx Z$.

В таком случае, если эти два множества приближенно равны сверху и снизу одновременно, то их называют приближенно равными и пишут $Y \approx Z$. Справедливы также следующие соотношения:

- 1) $\underline{F}Y \subset Y \subset \overline{F}Y$;
- 2) $\underline{F}1 = \overline{F}1 = 1$ (1 – все множество X);
- 3) $\underline{F}0 = \overline{F}0 = \emptyset$;
- 4) $\underline{F}(Y \cap Z) = \underline{F}Y \cap \underline{F}Z$;
- 5) $\overline{F}(Y \cup Z) = \overline{F}Y \cup \overline{F}Z$;
- 6) $\overline{F}Y = -\underline{F}(-Y)$;

$$7) \underline{F}Y = -\overline{F}(-Y);$$

Рассмотрим особый тип пространств приближений, используемый при классификации объектов на основе значений их атрибутов. Каждому атрибуту соответствует множество значений. Описание объекта задается, когда выбрано одно значение для каждого атрибута. Чтобы точнее выразить эту концепцию, введем определение информационной системы.

Информационной системой, что по сути своей является таблицей атрибут-значение, называется $IS = (U, Q, V, \varphi)$, где U – множество объектов, Q – множество атрибутов объектов, которые разделяются на условные C (condition attributes) и решающие D (decisional attributes) непересекающиеся между собой множества атрибутов ($Q = C \cup D$), $V = \bigcup_{q \in Q} V_q$ – множество значений атрибутов [2]. Смотри на описанный выше пример, множество значений атрибутов будет иметь следующий вид:

$$V_{\text{имя}} = \{\text{Андрей, Василиса, Василий, Владимир, Екатерина}\};$$

$$V_{\text{образование}} = \{\text{Среднее, Неоконченное высшее, Высшее, Аспирантура}\};$$

$$V_{\text{высокий доход}} = \{\text{Да, нет}\};$$

Функция φ является описательной функцией объектов. То есть, $\rho(x, q) \in V_q$ для каждого атрибута множества q из Q и объекта x из U . Исходя из вышесказанного, данная описательная функция определяет значение атрибута для конкретного объекта x из всего множества объектов U [1].

Установим полученную выше описательную функцию $\rho_x: Q \rightarrow V$, такую что $\rho_x(q) = \rho(x, q)$ для $\forall q \in Q$ и $x \in U$. Следовательно, ρ_x будет называться описанием объекта x в IS .

Положим, что $x, y \in O$ эквивалентны или неразличимы по отношению к $q \in Q$ в пространстве приближений, если $\rho_x(q) = \rho_y(q)$, записывая это как $x \tilde{q} y$, где \tilde{q} показывает отношение эквивалентности или неразличимости объектов в имеющемся наборе данных. В IS $x, y \in U$ эквивалентны по отношению к $F \subset Q$ при условии, что $\tilde{F} = \bigcap_{p \in F} \tilde{p}$. В случае $F = Q$ будем считать, что x и y

неразличимы в информационной системе. В такой ситуации запись будет иметь вид $xI\tilde{S}y$.

Совершенно очевидным является то, что \tilde{F} – это отношение неразличимости. Исходя из этого, $IS = (U, Q, V, \rho)$ определяет единственным образом пространство приближений $A_s(U, I\tilde{S})$.

1.2 АЛГОРИТМ RS1, ОСНОВАННЫЙ НА ТЕОРИИ ПРИБЛИЖЕННЫХ МНОЖЕСТВ

Теперь, когда мы познакомились с основными положениями теории приближенных множеств Здислава Павлака, можно применить полученные знания для реализации алгоритма RS1.

Допустим, что мы имеем некоторое обучающее множество S , содержащее в себе такие объекты, что $s \in U$. Сам по себе каждый объект описывается набором признаков, также называемых атрибутами. Все атрибуты делятся, как говорилось ранее, на два типа: описательные (x_1, x_2, \dots, x_n) и решающий атрибут d . В самом начале в обучающем множестве ищутся элементарные множества. Далее на основе полученных множеств строятся объединения и находятся верхнее и нижнее приближения. После этого необходимо обязательно выполнить редуцирование приближений, дабы устранить избыточность информации в датасете. В конечном итоге получаются две системы продукционных правил: для верхнего и нижнего приближений соответственно.

Формула точности аппроксимации для множества U в пространстве приближений $F = \langle S, R(D) \rangle$ будет иметь следующий вид:

$$\mu_p(U) = \frac{|Lower(U)|}{|Upper(U)|}$$

Основными концепциями приближенных множеств являются ядро и редуцирование (сокращение) [1]. Дадим определение редуцированию. Определим подмножество информационных атрибутов как $B \subset A$. Некоторый атрибут $q \in B$ примем в качестве лишнего в множестве атрибутов $B \subset A$ в том

случае, если $POS_B(D) = POS_{B-\{q\}}(D)$, иначе атрибут q будет являться необходимым атрибутом. Избыточный атрибут можно удалить, никак не изменяя при этом отношения эквивалентности между объектами. В том случае, если каждый атрибут $B \subset D$ является необходимым по отношению к D , то B называется ортогональным по отношению в D . Подмножество $B \subset A$ можно определить как сокращение в IS тогда, когда B ортогонально по отношению к D и справедливо равенство $POS_A(D) = POS_B(D)$.

Существенной частью информационной системы является сокращение, которое способно различить все объекты, где ядро – лишь общая часть всех сокращений.

Допустим, S – имеющаяся обучающая выборка, D – решающий атрибут, $F = \langle S, R(D) \rangle$ – пространство приближений, а $X \subset S$ – множество, для которого требуется найти описание. Для этого необходимо проделать следующие шаги:

- 1) Поиск атомов для S ;
- 2) Поиск верхнего и нижнего приближения для X в F ;
- 3) Вычислить точность аппроксимации μ ;
- 4) Поиск сокращения для нижнего и верхнего приближений;

Рассмотрим работу алгоритма на примере построения продукционных правил для задачи «игра в гольф». Исходные данные для задачи даны в Таблице 2

Таблица 2. Обучающая выборка для задачи «игра в гольф»

U	Погодные условия	Влажность, %	Ветер	Играть в гольф?
x_1	Пасмурно	87	Нет	Играть
x_2	Солнечно	80	Есть	Играть
x_3	Солнечно	80	Есть	Играть
x_4	Пасмурно	75	Есть	Играть
x_5	Пасмурно	75	Есть	Играть
x_6	Дождливо	80	Нет	Не играть
x_7	Солнечно	80	Есть	Не играть
x_8	Дождливо	80	Нет	Не играть

x_9	Дождливо	85	Нет	Не играть
x_{10}	Пасмурно	87	Нет	Играть

Дано обучающее множество U , представленное в таблице 2, которое содержит в себе записи о допустимых погодных условиях для игры в гольф. Необходимо получить описание множества X , состоящее из записей, в которых условия для игры являются допустимыми (т.е., для этих записей решающий атрибут «Играть в гольф?» принимает значение «Играть»). Итак, по шагам, описанным выше:

1) Атомами для U будут $a_1 = \{x_1, x_{10}\}$, $a_2 = \{x_2, x_3, x_7\}$, $a_3 = \{x_4, x_5\}$, $a_4 = \{x_6, x_8\}$, $a_5 = \{x_9\}$

2) $Lower_U(X) = (x_1, x_4, x_5, x_{10})$, $Upper_U(X) = \{x_1, x_2, x_3, x_4, x_5, x_7, x_{10}\}$

3) $\mu_U(X) = \frac{|Lower(X)|}{|Upper(X)|} = \frac{4}{7} \approx 0,57$

4) Редуцируем повторяющиеся записи для верхней и нижней аппроксимации

В конечном счете, результатом проделанной работы будет построение двух систем продукционных правил для верхней и нижней аппроксимации, которые будут иметь форму «ЕСЛИ...ТО...» для классификации погодных условий с конечной целью принятия решения играть в гольф или же нет. Построенные правила для нижней аппроксимации дают нам полностью уверенную классификацию решения, в то время как правила для верхней аппроксимации – возможную.

Правила для нижней аппроксимации:

1) **ЕСЛИ** (Прогноз погоды = Пасмурно) & (Влажность = 87) & (Ветер = Нет)

ТО (Играть в гольф = Играть)

2) **ЕСЛИ** (Прогноз погоды = Пасмурно) & (Влажность = 75) & (Ветер = Есть)

ТО (Играть в гольф = Играть)

Правила для верхней аппроксимации:

1) **ЕСЛИ** (Прогноз погоды = Пасмурно) & (Влажность = 87) & (Ветер = Нет)

ТО (Играть в гольф = Играть)

2) **ЕСЛИ** (Прогноз погоды = Солнечно) & (Влажность = 80) & (Ветер = Есть)

ТО (Играть в гольф = Играть)

3) **ЕСЛИ** (Прогноз погоды = Пасмурно) & (Влажность = 75) & (Ветер = Есть)

ТО (Играть в гольф = Играть)

На первый взгляд в глаза сразу бросается то, что вычисленное значение точности аппроксимации, в нашем случае, довольно невысоко, что, несомненно, может смутить. Однако, данная ситуация может быть вызвана множеством противоречивых данных. Таким образом, изучив более подробно наше обучающее множество игры в гольф из таблицы 2, можно обнаружить, что вторая или третья и седьмая записи объектов совпадают, но в то же время эти записи относятся к различным классам.

1.3 ПСЕВДОКОД АЛГОРИТМА RS1

Ранее мы изучили теорию приближенных множеств и алгоритм RS1, который основан на этой теории. Теперь ниже приведем пример псевдокода данного алгоритма, на который в дальнейшем будем опираться при переходе к программной реализации:

Функция RS1 (C : множество информационных атрибутов,

D : решающий атрибут,

S : обучающее множество)

Возвращает две системы продукционных правил.

Begin:

1. Находим все элементарные множества или же атомы c_1, c_2, \dots, c_n . Делаем просмотр обучающего множества S и ищем в нем неразличимые объекты среди тех, что определены полностью, далее строим элементарные

множества. Для этого 1-й объект сравнивается с $N-1$ оставшимся, 2-й – с $N-2$, и т. д. На следующем этапе мы просматриваем S и далее сравниваем каждый объект, который определен не полностью, с каждым атомом. Также стоит держать в голове, что сравнение производится только для информационных атрибутов из множества C .

2. Вычисляем нижнюю и верхнюю аппроксимации. Для этого примем, что $\{c_n\}$ $\{n = 1..k\}$ является множеством атомов, а $X \subseteq S$ – это множество, приближения к которому мы ищем. Искомое множество X вычисляется на основе значений решающего атрибута D .

2.1. Определение нижней аппроксимации.

$Lower(X) := \emptyset;$

For $n = 1$ **to** k

If $(c_n \subseteq X)$ **Then** $Lower(X) := Lower(X) \cup c_n$

End

2.2. Определение верхней аппроксимации.

$Upper(X) := \emptyset;$

For $n = 1$ **to** k :

If $(X \subseteq (S \setminus c_n))$ **Then** c_n пометить, как лишний **Else**

$Upper(X) := Upper(X) \cup c_n$

End

3. Проводим вычисление точности аппроксимации по следующей формуле:

$$\mu(X) := \frac{Lower(X)}{Upper(X)}$$

4. Находим редуцирование для приближений, избавляясь от избыточной и повторяющейся информации. Таким образом, от каждого элементарного множества оставляем лишь по одному его представителю.

5. По вычисленным верхней и нижней аппроксимации формируем систему продукционных правил:

$$POS(X) \rightarrow X$$

$$NEG(X) \rightarrow \bar{X}$$

$$BND(X) \rightarrow \text{возможно принадлежит } X$$

End RS1.

Для ранее рассмотренного примера "Игра в гольф", результатом работы алгоритма RS1 будет следующая система построенных продукционных правил для верхней и нижней аппроксимации, а также граничной области соответственно [1]. В качестве значения решающего атрибута примем значение «Играть».

Lower (X):

1) **ЕСЛИ** (Прогноз погоды = Пасмурно) & (Влажность = 87%) &

(Ветер = безразлично) **ТО** (Играть ли в гольф = Играть)

3) **ЕСЛИ** (Прогноз погоды = Пасмурно) & (Влажность = 75%) & (Ветер = безразлично) **ТО** (Играть ли в гольф = Играть)

Upper (X):

3) **ЕСЛИ** (Прогноз погоды = Дождливо) & (Влажность = 80%) & (Ветер = Ложь) **ТО** (Играть ли в гольф = Не играть)

4) **ЕСЛИ** (Прогноз погоды = Дождливо) & (Влажность = 85%) & (Ветер = Ложь) **ТО** (Играть ли в гольф = Не играть)

Boundry (X):

5) **ЕСЛИ** (Прогноз погоды = Солнечно) & (Влажность = 80%) & (Ветер = Истина) **ТО** (Играть ли в гольф = Возможно играть)

1.4 ПРЕИМУЩЕСТВА И НЕДОСТАТКИ ИСПОЛЬЗОВАНИЯ АЛГОРИТМА RS1

Прежде всего стоит начать с того, что существенным моментом в алгоритме RS1 является его способность строить систему продукционных правил, что в нашем случае, а именно – внедрение данного алгоритма в различные рода роботехнические системы или целые комплексы, является несомненным плюсом. Данное решение способствует тому, что робот может беспрепятственно и корректно воспринимать данную информацию. Также, если роботом управляет некий оператор, то для него продукционные правила гораздо легче воспринимаются, нежели, если бы алгоритм строил дерево решений, которое может доходить до довольно больших размеров.

Ко второй положительной черте алгоритма можно отнести длину получаемых описаний. Все дело в том, что RS1 охватывает все доступные ему атрибуты в полученное описание.

Третье преимущество использования алгоритма RS1 – это его обработка данных, которые могут быть всячески искажены или иметь своего рода шумы. Но, в определенных ситуациях, данное преимущество алгоритма RS1 может перерасти и в отрицательную сторону, поэтому его использование должно быть обосновано.

Четвертая, и самая уникальная, положительная черта – это возможность обработки обучающей выборки, которая содержит в себе такие объекты, у которых могут отсутствовать значения атрибутов. Путем создания всех возможных элементарных множеств из объектов, атрибуты которых не полностью заданы, алгоритм способен заполнять неполноту информации имеющегося набора данных.

Основным недостатком алгоритма RS1, к сожалению, является его вычислительная сложность, которая определяется следующей формулой:

$$C_{RS1} = S \cdot (k \cdot N^2),$$

где S – обучающее множество, k – количество атомов, а N – количество объектов [1]. Тем самым, сравнивая алгоритм RS1, например, с алгоритмом, строящим деревья решений, ID3, который был разработан и предложен Джоном Р. Квинланом, вычислительная сложность которого определяется как

$$C_{ID3} = S \cdot (k^2 \cdot N)$$

можно увидеть, что RS1 будет работать существенно медленнее при довольно большом объеме обучающих множеств, где значение N будет много больше 100.

В сухом остатке мы имеем то, что при выборе определенного алгоритма стоит мыслить рационально, опираясь на те цели и ожидаемые результаты, которые мы хотим получить, поскольку все алгоритмы не без изъянов и имеют свои как сильные, так и слабые стороны.

ГЛАВА 2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА RS1

2.1 ВЫБОР СРЕДЫ МОДЕЛИРОВАНИЯ И ЯЗЫКА ПРОГРАММИРОВАНИЯ

Выбор правильной среды моделирования и языка программирования является одним из ключевых и важным этапом в процессе программной реализации алгоритма RS1, который оказывает влияние на результативность и качество разработки программного обеспечения.

Прежде всего необходимо выбрать язык программирования. На основе проведенного анализа, в качестве языка программирования был выбран объектно-ориентированный язык программирования Python и вот причины данного выбора:

- 1) Python – довольно понятный, читаемый и простой в освоении язык программирования. С его помощью можно написать довольно короткий код, который будет иметь мощную программную базу. Кроме того, он обладает широким спектром библиотек.
- 2) Это высокоуровневый, кроссплатформенный язык программирования, который может работать на разных операционных системах [3].
- 3) Данный язык программирования с открытым исходным кодом, что дает возможность просмотреть и изменить код на своё усмотрение. Это делает Python очень гибким и адаптивным для различных задач и проектов.
- 4) Python имеет очень широкую область применения. Он может использоваться для разработки веб-приложений, научных вычислений, автоматизации задач, создания игр, машинного обучения, анализа данных и многого другого [4].

В качестве среды моделирования было принято решение использовать Visual Studio Code, поскольку она является одной из самых популярных сред разработки (IDE) с открытым исходным кодом, которая позволяет работать с большим количеством языков программирования, включая Python [3]. Эта среда

моделирования имеет множество преимуществ, которые делают её идеальной для разработки алгоритма RS1 на языке Python.

Во-первых, Visual Studio Code имеет мощный интегрированный отладчик Python, который позволяет быстро и эффективно находить, и исправлять ошибки в коде. Он также обеспечивает поддержку автоматического завершения кода, встроенного контроля версий и других полезных функций, которые помогают повысить производительность и качество написанного кода.

Во-вторых, Visual Studio Code обеспечивает максимальную возможность для настройки, что позволяет легко настроить свою среду разработки под любые индивидуальные потребности. Можно установить различные плагины и расширения, чтобы улучшить функциональность вашей IDE, а также настроить среду разработки, начиная от изменения схемы цветов, шрифты, отображения и т.д., заканчивая более сложными вещами.

В-третьих, использование языка Python в Visual Studio Code предоставляет доступ к богатой экосистеме библиотек и пакетов, которые могут значительно ускорить процесс разработки. Вы можете легко собрать и установить любые необходимые библиотеки, такие как NumPy, Pandas, Matplotlib и т.д., прямо из вашей среды разработки просто написав команду в терминал.

В-четвертых, Visual Studio Code доступен на всех платформах, включая Windows, macOS и Linux, что делает его удобным для работы в различных операционных системах.

Наконец, Visual Studio Code имеет огромное сообщество разработчиков, которые постоянно работают над улучшениями для этой IDE и создают всё новые расширения и плагины, которые могут в свою очередь значительно облегчить работу с ней, предоставляя максимальные удобства.

Таким образом, использование Visual Studio Code для разработки на языке программирования Python является отличным выбором для выполнения поставленной цели.

2.2 РАЗРАБОТКА МОДУЛЯ НА PYTHON, РЕАЛИЗУЮЩЕГО ОСНОВНЫЕ ПОЛОЖЕНИЯ АЛГОРИТМА RS1

В данном разделе перед нами будет стоять цель – программно реализовать основные моменты теории приближенных множеств и алгоритма RS1, построенном на этой теории. Прежде всего, перед началом работы, необходимо реализовать функцию, которая бы получала на свой вход датасет и сформировывала из него обучающую выборку O . Структура функции, возвращающая нам множество O , будет представлена ниже (см. Рисунок 2).

```
Def getU(self):  
    return set([o for o in range(len(self.DS))])
```

Рисунок 2. Реализация получения обучающего множества

После того как мы получили обучающее множество O перейдем к следующему шагу. А именно – формирование множества X ($getX$), для которого необходимо найти описание, поиск атомов, описывающих и решающего атрибутов ($getVa$ и $getVd$ соответственно). Реализация описанных функций представлена на рисунке 3.


```

Def getX(self,Vd):
    X = []
    for o in range(len(self.DS)):
        if self.DS[o][len(self.A)]==Vd:
            X.append(o)
    return set(X)

def getVa(self,a=""):
    Va = []
    if a in self.A:
        Va = set([i[self.A.index(a)] for i in self.DS])
    elif a=="":
        for a in self.A:
            Va.append( set([i[self.A.index(a)] for i in self.DS]) )
    return Va

def getVd(self):
    return set([i[len(self.A)] for i in self.DS])

```

Рисунок 3. Реализация формирования множества X и выделение информационных и решающих атрибутов

Так же, согласно теории, необходимым пунктом перед началом поиска верхнего и нижнего приближений является поиск пространства приближений. В данной реализации пространство приближений представлено в качестве P . Программная реализация данных функций описана ниже (см. Рисунок 4).

```

def getIND(self,P=[]):
    IND = []
    unique = []

    if P == []:
        P = self.A

    for o in range(len(self.DS)):
        Vo = []
        for a in self.A:
            if a in P:
                Vo.append( self.DS[o][self.A.index(a)] )
        if Vo not in unique:
            unique.append(Vo)
            IND.append([o])
        elif Vo in unique:
            i = unique.index(Vo)
            if isinstance(IND[i], list):
                IND[i].append(o)
            else:
                IND[i]=[IND[i]]
                IND[i].append(o)
    return IND

```

Рисунок 4. Поиск пространства приближений P

В соответствии с теорией, изложенной в главе 1, следующий этап является одним из самых существенных, поскольку на его основе далее будут строиться продукционные правила, которые, по своей сути, и являются ключевым моментом алгоритма RS1, основанного на теории приближенных множеств. Генерация верхнего и нижнего приближений происходит за счет реализованных функций *getUpperAX* и *getLowerAX* соответственно.

После поиска и формирования приближений мы можем найти граничный регион (*getBNDX*), в который попадут все те объекты из набора данных, которые точно не принадлежат ни одной из областей. Результат разработки данного этапа представлен на рисунке 5.

```

def getLowerAX(self,X,IND):
    lowerAX = []
    for i in range(len(IND)):
        if set(IND[i]).issubset(X):
            for n in range(len(IND[i])):
                lowerAX.append(IND[i][n])
    return set(lowerAX)

def getUpperAX(self,X,IND):
    upperAX = []
    for i in range(len(IND)):
        if len(set(IND[i]).intersection(set(X)))>0:
            for n in range(len(IND[i])):
                upperAX.append(IND[i][n])
    return set(upperAX)

def getPOSX(self,X,IND):
    return self.getLowerAX(X,IND)

def getBNDX(self,X,IND):
    return self.getUpperAX(X,IND) - self.getLowerAX(X,IND)

def getNEGX(self,X,IND):
    return self.getU()-(self.getPOSX(X,IND).union(self.getBNDX(X,IND)))

```

Рисунок 5. Реализация поиска верхнего и нижнего приближений, а также граничного региона

Принципиально ключевым моментом в алгоритме RS1 является построение систем продукционных правил для полученных аппроксимаций, имеющих форму «ЕСЛИ <условие> ТО <решение>». Столь большой акцент на текущий этап обуславливается еще тем, что за счет построенной системы продукционных правил, робототехнический комплекс, в который будет внедрена разработанная программа, сможет самостоятельно принимать какие-то решения в ответ на сложившуюся ситуацию. Решение задачи представлено на рисунке 6.

```

def getRules(self, region, P=[]):
    if P == []:
        P = self.A
    rules = ""
    count = 0
    for i in region:
        count+=1
        rule = ""
        for n in range(len(self.A)):
            if not rule:
                rule += " IF " + " ( " + P[n] + " == " + str(self.DS[i][n]) +
" ) "
            else:
                if n > 0:
                    rule += " & "
                rule += " ( " + P[n] + " == " + str(self.DS[i][n]) + " ) "
        if rule not in rules:
            rules += "\n [" + str(count) + "] " + rule
    return rules

```

Рисунок 6. Реализация формирования продукционных правил

Завершающим и немаловажным этапом будет подсчет точности аппроксимации и редуцирование избыточной информации, чего требует по условию алгоритм RS1 (см. рисунок 7). Для расчёта точности аппроксимации используется формула, описанная в теории.

```

def precision(self, X, IND):
    return len(self.getLowerAX(X, IND)) / len(self.getUpperAX(X, IND))

def getReduct(self):
    ind = self.getIND()
    red = []
    comb = self.comb(self.A)
    for a in comb:
        if self.getIND(a) == ind and a != self.A:
            red.append(a)
    return red

```

Рисунок 7. Реализация вычисления точности аппроксимации и редуцирования избыточной информации.

Казалось бы, что все основные этапы реализации алгоритма RS1, использующего теорию приближенных множеств, проделаны. С одной стороны, это так и есть. Все основные функции работы алгоритма RS1 реализованы. Но, с другой, необходимо понять, насколько точно и корректно работает разработанный программный комплекс, от которого зависит модель поведения робота. Для этого реализуем дополнительные функции для подсчета точности

классификации, выраженную в процентах, а также найдем процент неразличимости объектов в имеющемся датасете, чтобы можно было провести экзамен и проанализировать полученные результаты (см. рисунок 8).

```
def quality(self,X,IND):  
    return  
(len(self.getPOSX(X,IND))+len(self.getNEGX(X,IND)))/len(self.getU())*100  
  
def roughness(self,X,IND):  
    return (len(self.getBNDX(X,IND))/len(self.getU()))*100
```

Рисунок 8. Вычисление точности классификации и процента неразличимых объектов.

Полный исходный код разработанного модуля для программного комплекса, реализующего основные положения алгоритма, будет представлен в приложении к данной ВКРБ.

2.3 РАЗРАБОТКА ПРОГРАММНОГО ИНТЕРФЕЙСА И ЕГО ОПИСАНИЕ

Алгоритм RS1 – это мощный инструмент для классификации данных с неразличимыми значениями описательных атрибутов. Однако, чтобы использовать его на практике, нужен удобный и интуитивно понятный программный интерфейс.

Разработка программного интерфейса для алгоритма RS1 представляет собой задачу, требующую внимательного изучения требований пользователей и особенностей самого алгоритма. Необходимо предоставить возможность загрузки данных, настройки параметров алгоритма, его запуска, а также визуализации получившихся результатов.

При разработке интерфейса необходимо учитывать возможные ошибки и исключения, которые могут возникнуть в процессе использования программы пользователем. Также следует уделить внимание удобству использования интерфейса для пользователя, делая его не слишком перегруженным, чтобы он мог быстро освоиться с его функциональностью и достичь желаемых результатов.

В качестве вспомогательной библиотеки для построения графического интерфейса была выбрана PySimpleGUI. PySimpleGUI – это простая библиотека для создания графических интерфейсов пользователя для приложений, написанных на языке программирования Python.

PySimpleGUI предлагает простой и интуитивно понятный подход к созданию графических интерфейсов, используя множество элементов управления, таких как кнопки, текстовые поля, выпадающие списки и многое другое. Библиотека обладает достаточным функционалом для создания GUI приложений любой сложности, включая многопоточные и многопользовательские приложения.

Одним из главных преимуществ PySimpleGUI является возможность создания кроссплатформенных приложений. Кроме того, библиотека предлагает простую синтаксическую структуру и широкий спектр возможностей настройки внешнего вида элементов интерфейса.

Благодаря своей простоте и интуитивной структуре, PySimpleGUI позволяет быстро создавать графические интерфейсы для широкого круга приложений, от простых десктопных до научных и инженерных, в которых требуется обработка и визуализация данных.

Итак, разработка программного интерфейса для алгоритма RS1 является немаловажной задачей, которая поможет расширить возможности использования этого алгоритма и сделает его более доступным для пользователей с разным уровнем подготовки в компьютерных областях.

Вид главного окна приложения представлен на рисунке 9:



Рисунок 9. Главное окно приложения

Графический интерфейс разработанного программного комплекса предоставляет пользователю следующие возможности:

- **Open folder** – кнопка открытия проводника для выбора загружаемого датасета (расширение данного файла должно быть .txt)
- **Load data** – загрузка выбранного пользователем датасета
- В поле «Please enter the value of the decisive attribute of dataset» реализована возможность пользователю самостоятельно вводить значение решающего атрибута. Подсказки о возможных значениях данного атрибута выведены сверху во избежание ввода некорректных данных пользователем.
- **Run the algorithm** – кнопка запуска работы алгоритма
- В поле «Enter file name to export results» пользователь задает название и расширение файла, в который после нажатия кнопки **Export** будут записываться полученные результаты. Файл создается в той же

директории, где расположена программа. При наличии файла с таким же именем, что указал пользователь, старый файл будет перезаписан.

- **Exit** – кнопка выхода из программы

В качестве защиты от некорректной работы программы или же ошибок пользователя, в силу возможности его неопытности или других причин, были обработаны сопутствующие исключения:

- 1) Пользователь не может начать работу, не загрузив датасет. В данном случае программа выдаст ошибку с инструкцией о том, что прежде всего надо выбрать и загрузить данные.
- 2) Запуск алгоритма без ввода значения решающего атрибута невозможен. Программа выдаст предупреждение об этом.
- 3) Пользователь не может загрузить пустой файл, программа сразу выдаст ошибку с подсказкой об этом.
- 4) Невозможно экспортировать результаты в файл, не указав его имя.

Примеры защиты от некорректной работы программного комплекса и его реакции на подобного рода ситуации представлены на рисунках 10, 11, 12 и 13.

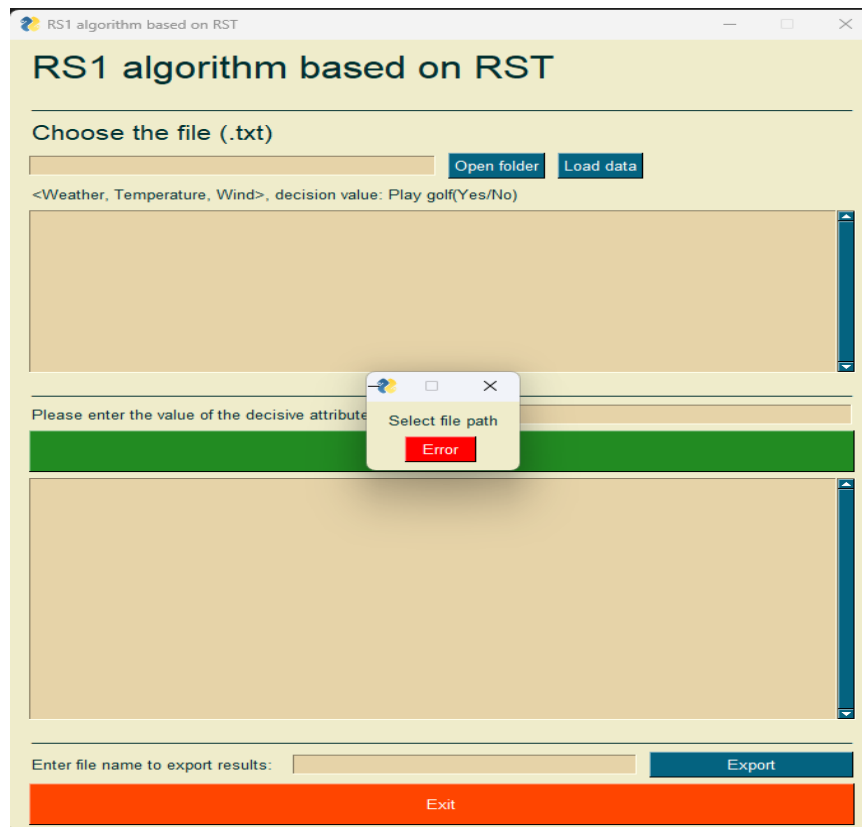


Рисунок 10. Попытка запуска без выбора датасета.

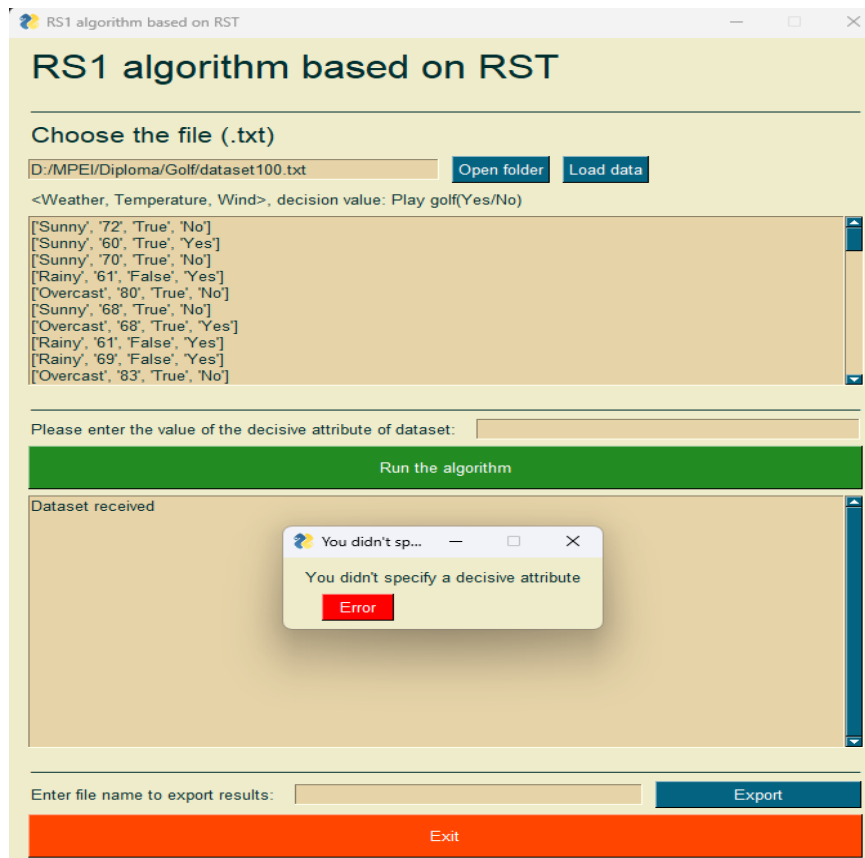


Рисунок 11. Попытка запуска без указания значения решающего атрибута

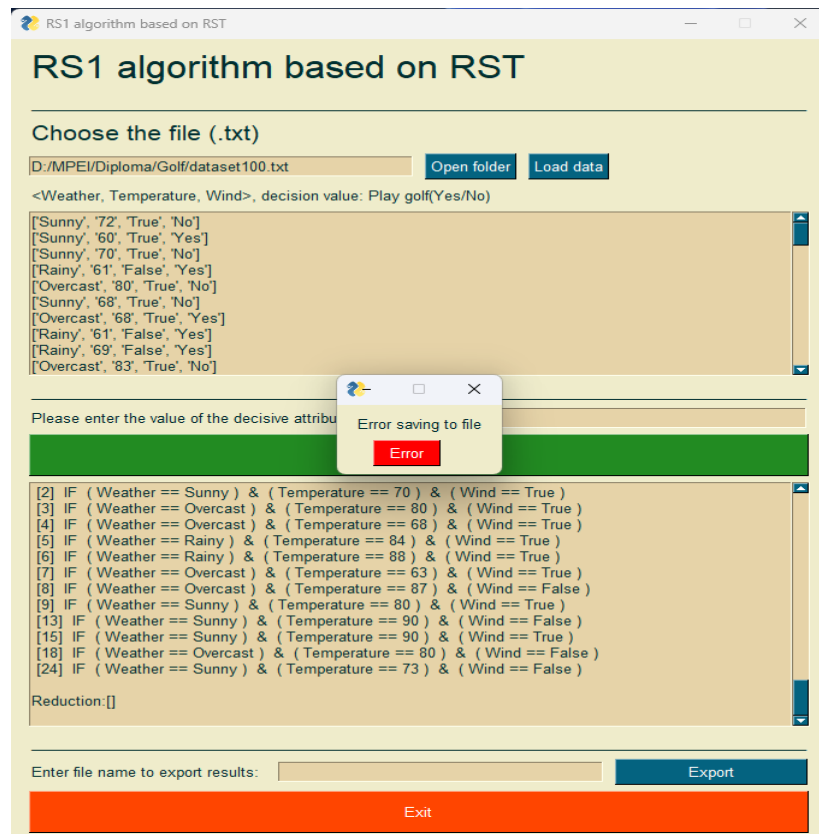


Рисунок 12. Попытка экспортировать результаты без указания файла

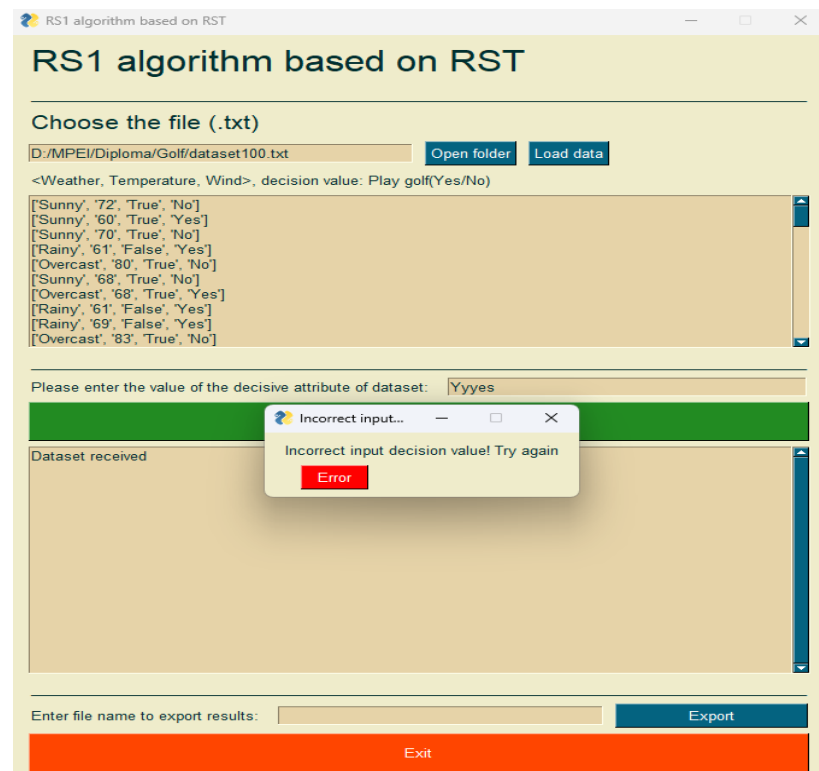


Рисунок 13. Попытка ввода некорректного значения решающего атрибута

2.4 ОТЛАДКА ПРОГРАММНОГО КОМПЛЕКСА И ТЕСТИРОВАНИЕ

Для отладки разработанной программы и тестирования сперва проверим её работоспособность на тестовом датасете «Гольф». Как уже рассказывалось ранее, в данном датасете существует 3 описательных атрибута, указывающих на возможные погодные условия, такие как: прогноз погоды (пасмурно, солнечно, дождливо), влажность воздуха, значения которого варьируются в пределах от 75 до 87 и наличие ветра (истина или ложь). Решающим атрибутом является «играть ли в гольф?», значения которого могут принимать лишь 2 вида – «Играть» или же «Не играть». Всего в датасете присутствует 10 объектов обучающего множества. Для того, чтобы убедиться в правильности вычислений и построении продукционных правил, сравнив их с теоретическими данными, зададим значение решающего атрибута равным «Играть». Полученные результаты работы программы представлены ниже (см. таблица 3).

Таблица 3. Результаты работы программы тестовой выборки "Игра в гольф"

$U \Rightarrow \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$X = \{x \mid \text{Play golf}(x) = \langle \text{Yes} \rangle\} \Rightarrow \{0, 1, 2, 3, 4, 9\}$
$V_a(\text{Weather}) \{ 'Sunny', 'Rainy', 'Cloudy' \}$
$V_a(\text{Temperature}) \{ '85', '80', '75', '87' \}$
$V_a(\text{Wind}) \{ 'True', 'False' \}$
$V_d \{ 'Yes', 'No' \}$
$V_a [\{ 'Sunny', 'Rainy', 'Cloudy' \}, \{ '85', '80', '75', '87' \}, \{ 'True', 'False' \}]$
$IND(A) [[0, 9], [1, 2, 6], [3, 4], [5, 7], [8]]$
$\text{Object's of LowerXA} \Rightarrow \{0, 9, 3, 4\}$
$\text{Object's of UpperXA} \Rightarrow \{0, 1, 2, 3, 4, 6, 9\}$
$POS(X) \{0, 9, 3, 4\}$
$BND(X) \{1, 2, 6\}$
$NEG(X) \{8, 5, 7\}$
Accuracy of Approximation: 0.571

Quality of Approximation: 70.0 %

Roughness: 30.0 %

Production rules for positive region for decision value: <Yes>

[1] IF (Weather == Cloudy) & (Temperature == 87) & (Wind == False)

[2] IF (Weather == Cloudy) & (Temperature == 75) & (Wind == True)

Production rules for negative region for decision value: <Yes>

[1] IF (Weather == Rainy) & (Temperature == 85) & (Wind == False)

[2] IF (Weather == Rainy) & (Temperature == 80) & (Wind == False)

Production rules for boundry region for decision value: <Yes>

[1] IF (Weather == Sunny) & (Temperature == 80) & (Wind == True)

Reduction: [['Weather', 'Temperature'], ['Temperature', 'Wind']]

Анализируя полученные результаты работы программы, можно удостовериться в том, что сама модель отлажена и построена верно, поскольку результаты в точности соответствуют тем, что были заявлены в теоретическом материале, в следствие чего можно смело переходить к тестированию программы на более крупных наборах данных, наиболее близких к реальным, которые используются на практике.

ГЛАВА 3. ТЕСТИРОВАНИЕ ПРОГРАММНОГО КОМПЛЕКСА НА БОЛЬШИХ ДАННЫХ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

В данной главе будет описано тестирование и дан анализ работы программного комплекса на достаточно больших наборах данных, которые наиболее приближены к тем, что используются на практике в разных сферах деятельности и организациях.

Прежде всего, перед тем как начать тестирование, необходимо правильно подобрать датасеты. Очевидным является то, что подобранные датасеты должны удовлетворять несколько условий, которые наш алгоритм позволяет обрабатывать. К таким условиям можно отнести следующее:

- 1) Наличие в датасете нечеткостей и противоречивых данных
- 2) Наличие в датасете повторяющихся записей
- 3) Датасет должен подходить для задач классификации

Без условий, описанных выше, поставленная задача тестирования и анализа полученных результатов при использовании алгоритма RS1 не имела, в таком случае, никакого смысла, поскольку не использовались и не проверялись бы основные преимущественные черты алгоритма, что в последствии приведет к некорректному анализу.

Источниками далее используемых датасетов, в том числе, являются репозиторий калифорнийского университета UCI Machine Learning Repository и сообщество специалистов по Data Science, принадлежащее корпорации Google, Kaggle.

3.1 ТЕСТИРОВАНИЕ АЛГОРИТМА НА БОЛЬШИХ ДАННЫХ

Первым на очереди будет довольно простой набор данных под названием «Flu», взятый из Kaggle. Данный датасет состоит всего из 50 объектов и описывает возможность заболевания человека гриппом, основываясь на трех информационных атрибутах. К этим информационным атрибутам относятся:

головная боль, мышечная боль и показания температуры тела человека. Значения атрибута «головная боль» могут принимать лишь два следующих вида: yes и no. Точно так же ситуация обстоит и с возможными значениями атрибута «мышечная боль», которые принимают значения «yes» и «no». Атрибут «температура тела человека» имеет четыре значения: very low, low, normal, high и very high. Решающим атрибутом является «болен ли человек гриппом?», который так же принимает два возможных значения «yes» и «no».

Примем значение решающего атрибута равным «yes». Экспортированные результаты работы алгоритма в файл изображены на рисунках 14 и 15.

```

U => {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
40, 41, 42, 43, 44, 45, 46, 47, 48, 49}
X={x| Flu(x)=<yes>} => {0, 1, 2, 5, 6, 8, 9, 11, 12, 14, 15, 18, 20, 22, 24, 25, 26, 27, 28, 33, 34, 35, 37, 40, 42, 45, 48}
Va(Headache){'yes', 'no'}
Va(Muscle Pain){'yes', 'no'}
Va(Temperature){'normal', 'high', 'veryhigh', 'verylow', 'low'}
Vd{'yes', 'no'}
Va[{'yes', 'no'}, {'yes', 'no'}, {'normal', 'high', 'veryhigh', 'verylow', 'low'}]
IND(A)[[0, 24], [1, 4], [2], [3, 9, 20, 36], [5, 37], [6, 19, 23, 26, 44, 45], [7, 39, 47], [8, 18], [10, 14, 32, 33], [11, 21, 29, 35, 42], [12, 13, 40, 49],
[15, 22], [16, 46], [17, 25, 43, 48], [27, 41], [28], [30], [31], [34, 38]]
Object's of LowerXA => {0, 2, 37, 5, 8, 15, 18, 22, 24, 28}
Object's of UpperXA => {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 32, 33, 34, 35, 36, 37, 38,
40, 41, 42, 43, 44, 45, 48, 49}
POS(X){0, 2, 37, 5, 8, 15, 18, 22, 24, 28}
BND(X){1, 3, 4, 6, 9, 10, 11, 12, 13, 14, 17, 19, 20, 21, 23, 25, 26, 27, 29, 32, 33, 34, 35, 36, 38, 40, 41, 42, 43, 44, 45, 48, 49}
NEG(X){39, 7, 46, 47, 16, 30, 31}
Accuracy of Approximation: 0.232
Quality of classification: 34.0 %
Roughness: 66.0 %

```

Рисунок 14. Результат работы алгоритма на датасете "Flu" при поиске приближений, точности классификации, аппроксимации и неразличимости объектов

Production rules for positive region for decision value: <yes>

- [1] IF (Headache == no) & (Muscle Pain == yes) & (Temperature == high)
- [2] IF (Headache == yes) & (Muscle Pain == yes) & (Temperature == veryhigh)
- [3] IF (Headache == no) & (Muscle Pain == yes) & (Temperature == veryhigh)
- [4] IF (Headache == no) & (Muscle Pain == no) & (Temperature == veryhigh)
- [5] IF (Headache == yes) & (Muscle Pain == yes) & (Temperature == high)
- [6] IF (Headache == yes) & (Muscle Pain == yes) & (Temperature == verylow)

Production rules for negative region for decision value: <yes>

- [1] IF (Headache == no) & (Muscle Pain == no) & (Temperature == normal)
- [2] IF (Headache == no) & (Muscle Pain == no) & (Temperature == high)
- [3] IF (Headache == no) & (Muscle Pain == yes) & (Temperature == verylow)
- [4] IF (Headache == yes) & (Muscle Pain == no) & (Temperature == normal)

Production rules for boundry region for decision value: <yes>

- [1] IF (Headache == yes) & (Muscle Pain == no) & (Temperature == high)
- [2] IF (Headache == no) & (Muscle Pain == yes) & (Temperature == normal)
- [3] IF (Headache == yes) & (Muscle Pain == no) & (Temperature == low)
- [4] IF (Headache == no) & (Muscle Pain == no) & (Temperature == low)
- [5] IF (Headache == yes) & (Muscle Pain == yes) & (Temperature == low)
- [6] IF (Headache == yes) & (Muscle Pain == no) & (Temperature == veryhigh)
- [7] IF (Headache == no) & (Muscle Pain == yes) & (Temperature == low)
- [8] IF (Headache == no) & (Muscle Pain == no) & (Temperature == verylow)
- [9] IF (Headache == yes) & (Muscle Pain == yes) & (Temperature == normal)

Рисунок 15. Построенные продукционные правила для датасета "Flu"

Второй датасет «Profit», взятый с того же Kaggle, описывает влияние четырех факторов выпуска продукции на прибыль компании. К информационным атрибутам относятся следующие: длительность выпуска товара, имеющий вариации в виде «Big», «Medium», «Low»; наличие конкуренции на рынке со значениями «Yes», «No»; тип выпускаемого товара с пятью значениями «Lwr», «Hwr», «Twr», «Kwr» и «Swr»; наличие финансового кризиса – «Yes» или «No». Решающим атрибутом является прогнозирование прибыли компании с проданных ей товаров с двумя значениями – «Raise» или же «Fall». Результат работы продемонстрирован на рисунках 16 и 17.

$U \Rightarrow \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51\}$
 $X = \{x | \text{Profit}(x) = \text{<Raise>}\} \Rightarrow \{2, 4, 5, 7, 9, 13, 14, 18, 19, 20, 22, 23, 24, 25, 27, 28, 29, 30, 31, 38, 39, 40, 42, 45, 46, 47, 48\}$
 $Va(\text{Duration})\{\text{'Low', 'Medium', 'Big'}\}$
 $Va(\text{Competition})\{\text{'No', 'Yes'}\}$
 $Va(\text{Type})\{\text{'Kwr', 'Hwr', 'Lwr', 'Twr', 'Swr', 'Mwr'}\}$
 $Va(\text{Crisis})\{\text{'No', 'Yes'}\}$
 $Vd\{\text{'Raise', 'Fall'}\}$
 $Va[\{\text{'Low', 'Medium', 'Big'}\}, \{\text{'No', 'Yes'}\}, \{\text{'Kwr', 'Hwr', 'Lwr', 'Twr', 'Swr', 'Mwr'}\}, \{\text{'No', 'Yes'}\}]$
 $IND(A)[[0], [1], [2], [3], [4, 10], [5], [6], [7], [8], [9], [11], [12], [13], [14], [15], [16], [17], [18], [19, 20, 21], [22], [23], [24], [25, 26], [27], [28], [29], [30], [31, 32], [33], [34], [35], [36], [37], [38], [39], [40, 43], [41], [42], [44, 45], [46], [47], [48], [49, 51], [50]]$
 $\text{Object's of LowerXA} \Rightarrow \{2, 5, 7, 9, 13, 14, 18, 22, 23, 24, 27, 28, 29, 30, 38, 39, 42, 46, 47, 48\}$
 $\text{Object's of UpperXA} \Rightarrow \{2, 4, 5, 7, 9, 10, 13, 14, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 38, 39, 40, 42, 43, 44, 45, 46, 47, 48\}$
 $POS(X)\{2, 5, 7, 9, 13, 14, 18, 22, 23, 24, 27, 28, 29, 30, 38, 39, 42, 46, 47, 48\}$
 $NEG(X)\{0, 1, 3, 6, 8, 11, 12, 15, 16, 17, 33, 34, 35, 36, 37, 41, 49, 50, 51\}$
 $BND(X)\{32, 4, 40, 10, 43, 44, 45, 19, 20, 21, 25, 26, 31\}$
 Accuracy of Approximation: 0.6
 Quality of classification: 75.0 %
 Roughness: 25.0 %

Рисунок 16. Результат работы алгоритма на датасете "Profit" при поиске приближений, точности классификации, аппроксимации и неразличимости объектов

```

Production rules for positive region for decision value: <Raise>
[1] IF ( Duration == Big ) & ( Competition == No ) & ( Type == Swr ) & ( Crisis == No )
[2] IF ( Duration == Big ) & ( Competition == Yes ) & ( Type == Lwr ) & ( Crisis == Yes )
[3] IF ( Duration == Big ) & ( Competition == No ) & ( Type == Kwr ) & ( Crisis == No )
[4] IF ( Duration == Big ) & ( Competition == No ) & ( Type == Lwr ) & ( Crisis == No )
[5] IF ( Duration == Big ) & ( Competition == No ) & ( Type == Twr ) & ( Crisis == Yes )
[6] IF ( Duration == Big ) & ( Competition == Yes ) & ( Type == Twr ) & ( Crisis == Yes )
[7] IF ( Duration == Medium ) & ( Competition == Yes ) & ( Type == Kwr ) & ( Crisis == No )
[8] IF ( Duration == Medium ) & ( Competition == Yes ) & ( Type == Lwr ) & ( Crisis == No )
[9] IF ( Duration == Medium ) & ( Competition == No ) & ( Type == Swr ) & ( Crisis == Yes )
[10] IF ( Duration == Medium ) & ( Competition == Yes ) & ( Type == Lwr ) & ( Crisis == Yes )
[11] IF ( Duration == Medium ) & ( Competition == No ) & ( Type == Mwr ) & ( Crisis == No )
[12] IF ( Duration == Medium ) & ( Competition == Yes ) & ( Type == Mwr ) & ( Crisis == No )
[13] IF ( Duration == Medium ) & ( Competition == No ) & ( Type == Mwr ) & ( Crisis == Yes )
[14] IF ( Duration == Medium ) & ( Competition == Yes ) & ( Type == Mwr ) & ( Crisis == Yes )
[15] IF ( Duration == Low ) & ( Competition == Yes ) & ( Type == Swr ) & ( Crisis == No )
[16] IF ( Duration == Low ) & ( Competition == Yes ) & ( Type == Kwr ) & ( Crisis == No )
[17] IF ( Duration == Low ) & ( Competition == No ) & ( Type == Swr ) & ( Crisis == Yes )
[18] IF ( Duration == Low ) & ( Competition == No ) & ( Type == Mwr ) & ( Crisis == No )
[19] IF ( Duration == Low ) & ( Competition == Yes ) & ( Type == Mwr ) & ( Crisis == No )
[20] IF ( Duration == Low ) & ( Competition == Yes ) & ( Type == Mwr ) & ( Crisis == Yes )

Production rules for negative region for decision value: <Raise>
[1] IF ( Duration == Big ) & ( Competition == Yes ) & ( Type == Swr ) & ( Crisis == Yes )
[2] IF ( Duration == Big ) & ( Competition == No ) & ( Type == Swr ) & ( Crisis == Yes )
[3] IF ( Duration == Big ) & ( Competition == No ) & ( Type == Hwr ) & ( Crisis == No )
[4] IF ( Duration == Big ) & ( Competition == Yes ) & ( Type == Kwr ) & ( Crisis == No )
[5] IF ( Duration == Big ) & ( Competition == Yes ) & ( Type == Lwr ) & ( Crisis == No )
[6] IF ( Duration == Big ) & ( Competition == No ) & ( Type == Mwr ) & ( Crisis == No )
[7] IF ( Duration == Big ) & ( Competition == No ) & ( Type == Mwr ) & ( Crisis == Yes )
[8] IF ( Duration == Medium ) & ( Competition == Yes ) & ( Type == Swr ) & ( Crisis == Yes )
[9] IF ( Duration == Medium ) & ( Competition == Yes ) & ( Type == Hwr ) & ( Crisis == Yes )
[10] IF ( Duration == Medium ) & ( Competition == Yes ) & ( Type == Kwr ) & ( Crisis == Yes )
[11] IF ( Duration == Medium ) & ( Competition == Yes ) & ( Type == Twr ) & ( Crisis == No )
[12] IF ( Duration == Medium ) & ( Competition == Yes ) & ( Type == Twr ) & ( Crisis == Yes )
[13] IF ( Duration == Low ) & ( Competition == No ) & ( Type == Lwr ) & ( Crisis == No )
[14] IF ( Duration == Low ) & ( Competition == Yes ) & ( Type == Lwr ) & ( Crisis == No )
[15] IF ( Duration == Low ) & ( Competition == No ) & ( Type == Lwr ) & ( Crisis == Yes )
[16] IF ( Duration == Low ) & ( Competition == Yes ) & ( Type == Hwr ) & ( Crisis == No )
[17] IF ( Duration == Low ) & ( Competition == Yes ) & ( Type == Twr ) & ( Crisis == No )
[18] IF ( Duration == Low ) & ( Competition == No ) & ( Type == Twr ) & ( Crisis == Yes )

Production rules for boundry region for decision value: <Raise>
[1] IF ( Duration == Medium ) & ( Competition == No ) & ( Type == Twr ) & ( Crisis == No )
[2] IF ( Duration == Big ) & ( Competition == No ) & ( Type == Lwr ) & ( Crisis == Yes )
[3] IF ( Duration == Low ) & ( Competition == No ) & ( Type == Hwr ) & ( Crisis == Yes )
[4] IF ( Duration == Low ) & ( Competition == Yes ) & ( Type == Lwr ) & ( Crisis == Yes )
[5] IF ( Duration == Medium ) & ( Competition == No ) & ( Type == Hwr ) & ( Crisis == No )
[6] IF ( Duration == Medium ) & ( Competition == No ) & ( Type == Lwr ) & ( Crisis == No )

```

Рисунок 17. Построение продукционных правил для датасета "Profit"

На этот раз мы усложним работу для нашего программного комплекса тем, что подадим ему датасет с количеством объектов, в два раза превосходящих ранее датасетов. Этим набором данных будет «Golf», что был приведен в пример в теоретическом материале. Поскольку ранее уже было его описание, то на этот раз просто получим результаты работы программного комплекса (см. рисунок 18), но без построенных продукционных правил, поскольку их количество вышло довольно велико. Так же попробуем немного исказить, добавить шума в нашу выборку, убрав букву информационного атрибута про ветер, написав в нем «Tue», вместо «True», чтобы посмотреть на реакцию алгоритма. Получившиеся продукционные правила для «Golf» будут представлены в приложении к настоящей ВКРБ.

```

U => {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68,
69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99}
X={x| Recruit(x)=<No>} => {0, 2, 4, 5, 9, 10, 11, 14, 15, 17, 19, 20, 21, 27, 32, 38, 39, 40, 41, 44, 46, 49, 50, 53, 54, 58, 59, 60,
61, 64, 65, 69, 70, 72, 74, 77, 78, 80, 81, 82, 84, 85, 86, 87, 88, 92, 95, 96, 99}
Va(Weather){'Sunny', 'Overcast', 'Rainy'}
Va(Temperature){'79', '80', '89', '86', '65', '73', '82', '78', '60', '87', '67', '68', '74', '70', '84', '63', '66', '77', '62', '88',
'83', '72', '90', '85', '76', '75', '61', '95', '69', '71', '59'}
Va(Wind){'Tue', 'True', 'False'}
Vd{'Yes', 'Tes', 'No'}
Va({'Sunny', 'Overcast', 'Rainy'}, {'79', '80', '89', '86', '65', '73', '82', '78', '60', '87', '67', '68', '74', '70', '84', '63',
'66', '77', '62', '88', '83', '72', '90', '85', '76', '75', '61', '95', '69', '71', '59'}, {'Tue', 'True', 'False'})
IND(A)[[0, 14], [1, 58], [2, 71], [3, 7], [4, 66], [5], [6, 48], [8], [9], [10], [11], [12], [13], [15], [16], [17], [18], [19], [20,
94], [21], [22], [23, 95], [24, 51], [25], [26], [27], [28], [29], [30, 44, 84], [31], [32], [33, 42, 53], [34, 35, 39], [36, 37], [38,
40, 81], [41], [43, 99], [45], [46, 47], [49], [50], [52, 60], [54], [55, 98], [56], [57], [59], [61], [62], [63], [64], [65], [67,
68], [69], [70], [72], [73], [74], [75], [76], [77, 90], [78], [79], [80], [82], [83, 89], [85], [86], [87], [88], [91], [92], [93],
[96], [97]]
Object's of LowerXA =>{0, 5, 9, 10, 11, 14, 15, 17, 19, 21, 27, 32, 38, 40, 41, 49, 50, 54, 59, 61, 64, 65, 69, 70, 72, 74, 78, 80, 81,
82, 85, 86, 87, 88, 92, 96}
Object's of UpperXA =>{0, 1, 2, 4, 5, 9, 10, 11, 14, 15, 17, 19, 20, 21, 23, 27, 30, 32, 33, 34, 35, 38, 39, 40, 41, 42, 43, 44, 46,
47, 49, 50, 52, 53, 54, 58, 59, 60, 61, 64, 65, 66, 69, 70, 71, 72, 74, 77, 78, 80, 81, 82, 84, 85, 86, 87, 88, 90, 92, 94, 95, 96, 99}
POS(X){0, 5, 9, 10, 11, 14, 15, 17, 19, 21, 27, 32, 38, 40, 41, 49, 50, 54, 59, 61, 64, 65, 69, 70, 72, 74, 78, 80, 81, 82, 85, 86, 87,
88, 92, 96}
BND(X){1, 2, 4, 20, 23, 30, 33, 34, 35, 39, 42, 43, 44, 46, 47, 52, 53, 58, 60, 66, 71, 77, 84, 90, 94, 95, 99}
NEG(X){3, 6, 7, 8, 12, 13, 16, 18, 22, 24, 25, 26, 28, 29, 31, 36, 37, 45, 48, 51, 55, 56, 57, 62, 63, 67, 68, 73, 75, 76, 79, 83, 89,
91, 93, 97, 98}
-----
Accuracy of Approximation: 0.571
Quality of classification: 73.0 %
Roughness: 27.0 %

```

Рисунок 18. Результат работы алгоритма на датасете "Golf" при поиске приближений, точности классификации, аппроксимации и неразличимости объектов

Теперь еще больше усложняем задачу. Рассмотрим датасет «Car evaluation» из репозитория калифорнийского университета UCI Machine Learning Repository. Данный набор насчитывает количество объектов в размере 1728 экземпляров, значения информационных атрибутов которых являются мультивариативными, то бишь разного типа. Все объекты описываются шестью информационными атрибутами: цена автомобиля (vhigh, high, med, low), состояние автомобиля (vhigh, high, med, low), количество дверей автомобиля (2, 3, 4, 5more), количество посадочных мест (2, 4, more), размер багажного отделения (small, med, big) и безопасность автомобиля (low, med, high). Решающим атрибутом является класс, присвоенный автомобилю, значения которого могут принимать вид «unacc», «acc», «good» и «vgood». Опять же, поскольку размер датасета достаточно внушительный, сформированные продукционные правила, а также вывод объектов нижней и верхней аппроксимации, и тех объектов, что попали в граничный регион не будут приведены из тех соображений, что их получается слишком много (в порядке 1000 экземпляров только лишь для одного из приближений). В остальном, результат работы для значения решающего атрибута равным «unacc» показан на рисунке 19.

```

Va(Buying){'med', 'high', 'low', 'vhigh'}
Va(Maint){'med', 'high', 'low', 'vhigh'}
Va(Doors){'3', '4', '2', '5more'}
Va(Persons){'4', '2', 'more'}
Va(Lug_boot){'small', 'med', 'big'}
Va(Safety){'med', 'high', 'low'}
Vd{'vgood', 'good', 'unacc', 'acc'}
Accuracy of Approximation: 0.811
Quality of classification: 86.458 %
Roughness: 33.335 %

```

Рисунок 19. Результат работы алгоритма на датасете "Car evaluation" при поиске точности классификации, аппроксимации и неразличимости объектов

Проведя вышеописанные эксперименты, можно смело переходить к анализу полученных результатов и их сравнению.

3.2 АНАЛИЗ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ

Проведя эксперименты над разработанным программным комплексом, запишем, для удобства, все полученные результаты в сводную таблицу для их последующей оценки (см. таблица 4).

Таблица 4. Сводная таблица результатов проведенных экспериментов

Датасет	Количество объектов	Точность аппроксимации	Точность классификации, %	Неразличимость объектов, %
Flu	50	0,232	34	66
Profit	52	0,6	75	25
Golf	100	0,571	73	27
Cars	1728	0,811	86,458	33,335

Исходя из приведенных выше результатов стоит отметить, что во всех, даже нетривиальных случаях, разработанный программный комплекс отработал в штатном режиме, не выдавая никаких сторонних ошибок, к которым не относятся ошибки со стороны пользователя. Эксперименты проводились с разным количеством объектов в датасете и типом информационных атрибутов, что сказывалось только лишь на времени обработки и вычислений алгоритма, которое в свою очередь зависит в том числе и от уровня мощности аппаратных ресурсов, на которых запускается программа. Но и оно уместается в пределах разумного, не достигая каких-то критических показателей. Предельно допустимым количеством информационных атрибутов является 10, а предельным количеством объектов в датасете – 10.500 тысяч. Также важным моментом является разметка данных в датасете, поскольку они все друг от друга отличаются. Программа не будет воспринимать датасет, выдавая ошибку, разметка атрибутов которого осуществлялась при помощи запятых, слешей и

прочих специальных символов. Разработанная программа воспринимает атрибуты, перечисление которых осуществляется строго через пробел.

На первый взгляд может показаться, что первый эксперимент, «Flu», выдался неудачным из-за низких показателей точности аппроксимации и процента точности классификации объектов, но это не совсем так. Данные показатели обуславливаются тем, что набор данных имел довольно много примеров выборки с нечеткостями. Собственно, эта цель и преследовалась. Важно было увидеть, как именно поведет себя программа, если ей на обработку дать датасет с очень противоречивой информацией, и справится ли она с поставленной задачей, поскольку, например, иной алгоритм, не умеющий работать с нечеткостями, просто бы откинул подобного рода объекты, никак их не воспринимая. Думаю, что поставленная цель, в данном случае, была достигнута, и программа успешно прошла предложенный ей тест-кейс.

Второй эксперимент, «Profit», выдался, я считаю, вполне удачным. В нем проверялось как программа отреагирует на то, если у информационного атрибута будет много всевозможных значений, а не 2-3, как это было ранее. В данном случае, таким атрибутом является «тип выпускаемого товара», который может принимать в себе вплоть до 6 значений. При неразличимости 25% объектов, точность аппроксимации и классификации получилась вполне достойной.

В третьем эксперименте, «Golf», был намерено искажен один из информационных атрибутов. Цель данного эксперимента заключалась в том, чтобы проверить, создаст ли алгоритм новое множество для искаженного атрибута или же нет. Но, опять же, как говорилось ранее, это можно посчитать и провалом, поскольку всё зависит от того, какие результаты мы ожидаем получить от работы. В настоящей ситуации, выявить искаженную информацию, имеющую шумы, не было основной целью тест-кейса. В остальном, результаты получились достойными.

Четвертый эксперимент, «Car evaluation», включал в себя несколько проверок. Во-первых, как программа примет и отреагирует на датасет внушающего размера. Во-вторых, что будет, если все информационные

атрибуты будут иметь не один и тот же тип данных, а будут отличаться друг от друга. Данная проверка была пройдена, и результаты превзошли все ожидания. При 33% неразличимых объектов, алгоритм точно классифицировал 86% из них и получил самое большое значение точности аппроксимации, среди остальных.

Неформально был проведен и еще один стресс-эксперимент, не описанный выше, в котором загружался датасет из 20 тысяч экземпляров объектов, описываемых 10-ю информационными атрибутами. Целью данного эксперимента являлось определение предельного количества объектов и атрибутов. Если про количество объектов можно сказать, что допустимо и еще больше, то к количеству атрибутов такое высказывание не применимо, поскольку оно жестко прописано в исходном коде программы. В итоге, как и ожидалось, увеличение размера датасета повлияло только лишь на время работы программы, поскольку иных манипуляций, кроме увеличения размера набора данных, не проводилось.

ЗАКЛЮЧЕНИЕ

Целью ВКР являлось изучение возможностей построения обобщенных понятий при использовании алгоритма RS1 с целью формирования модели поведения робота. При реализации проекта было выполнено следующее.

Была рассмотрена и изучена теория приближенных множеств. Был детально разобран алгоритм RS1, основой которого настоящая теория является. Алгоритм в подробностях представлен на псевдокоде. Даны примеры работы алгоритма.

Было выполнено сравнение возможностей алгоритма RS1 и ID3, дана оценка вычислительной сложности алгоритма RS1, рассмотрены его положительные и отрицательные стороны.

Обоснованы выбор среды моделирования, языка программирования. Дано поэтапное описание программных компонентов разработанного программного комплекса RST_RS1. Для созданного программного комплекса разработан интерфейс, наглядный и удобный в использовании. Для предотвращения неправильных действий пользователя создана система всплывающих подсказок и предупреждающих сообщений. В дальнейшем программный комплекс может быть успешно применен при использовании разных робототехнических систем и комплексов для формирования их поведения.

Работа программного приложения протестирована на различных наборах данных из известных коллекций данных. Каждый набор данных (датасет) детально описан, даны результаты, полученные алгоритмом для каждого набора в различных режимах работы. Сравнение полученных результатов с опубликованными в научной литературе позволило сделать вывод, что алгоритм работает корректно. Задание на ВКР выполнено в полном объеме.

Дальнейшей целью является усовершенствование разработанного программного комплекса для робототехнических систем посредством наложения еще одного алгоритма и практическим внедрением в конкретного робота.

СПИСОК ЛИТЕРАТУРЫ

1. Вагин В. Н., Головина Е. Ю., Загорянская А. А., Фомина М. В. «Достоверный и правдоподобный вывод в интеллектуальных системах», ООО Издательская фирма «Физико-математическая литература», 978-9221-0962-8, 2008
2. Zdzislaw I. Pawlak «Rough sets: Theoretical aspects of reasoning about data», Kluwer Academic Publishers, 978-0792314727, 1991
3. Лутц М. «Изучаем Python», ООО Издательская фирма «Диалектика», 978-5-907144-52-1, 2019
4. Дж. Вандер Плас, «Python для сложных задач. Наука о данных и машинное обучение», ООО Издательская фирма «Питер», 978-5-4461-0914-2, 2018
5. Уэс Маккини, «Python и анализ данных», ООО Издательская фирма «ДМК-Пресс», 978-5-97060-315-4, 2023
6. Ajith Abraham, Rafael Falcon, Rafael Bello, «Rough Set Theory: A True Landmark in Data Analysis», «Springer», 978-3540899204, 2009

ПРИЛОЖЕНИЕ

Программный модуль с реализацией алгоритма RS1:

```
class TWD:
    def __init__(self, DS=[],A=[]):
        self.DS = DS
        self.A = A

    def getU(self):
        return set([o for o in range(len(self.DS))])

    def getX(self,Vd):
        X = []
        for o in range(len(self.DS)):
            if self.DS[o][len(self.A)]==Vd:
                X.append(o)
        return set(X)

    def getVa(self,a=""):
        Va = []
        if a in self.A:
            Va = set([i[self.A.index(a)] for i in self.DS])
        elif a=="":
            for a in self.A:
                Va.append( set([i[self.A.index(a)] for i in self.DS]) )
        return Va

    def getVd(self):
        return set([i[len(self.A)] for i in self.DS])

    def getIND(self,P=[]):
        IND = []
        unique = []

        if P == []:
            P = self.A

        for o in range(len(self.DS)):
            Vo = []
            for a in self.A:
                if a in P:
                    Vo.append( self.DS[o][self.A.index(a)] )
            if Vo not in unique:
                unique.append(Vo)
                IND.append([o])
            elif Vo in unique:
                i = unique.index(Vo)
                if isinstance(IND[i], list):
```



```

        IND[i].append(o)
    else:
        IND[i]=[IND[i]]
        IND[i].append(o)
    return IND

def getLowerAX(self,X,IND):
    lowerAX = []
    for i in range(len(IND)):
        if set(IND[i]).issubset(X):
            for n in range(len(IND[i])):
                lowerAX.append(IND[i][n])
    return set(lowerAX)

def getUpperAX(self,X,IND):
    upperAX = []
    for i in range(len(IND)):
        if len(set(IND[i]).intersection(set(X)))>0:
            for n in range(len(IND[i])):
                upperAX.append(IND[i][n])
    return set(upperAX)

def getPOSX(self,X,IND):
    return self.getLowerAX(X,IND)

def getBNDX(self,X,IND):
    return self.getUpperAX(X,IND) - self.getLowerAX(X,IND)

def getNEGX(self,X,IND):
    return self.getU()-(self.getPOSX(X,IND).union(self.getBNDX(X,IND)))

def getRules(self,region,P=[]):
    if P == []:
        P = self.A
    rules = ""
    count = 0
    for i in region:
        count+=1
        rule = ""
        for n in range(len(self.A)):
            if not rule:
                rule += " IF " + " ( " + P[n] + " == " + str(self.DS[i][n]) + "
) "
            else:
                if n > 0:
                    rule += " & "
                rule += " ( " + P[n] + " == " + str(self.DS[i][n]) + " ) "
        if rule not in rules:
            rules += "\n [" +str(count)+"] " + rule
    return rules

```

```

def precision(self,X,IND):
    return len(self.getLowerAX(X,IND))/len(self.getUpperAX(X,IND))

def quality(self,X,IND):
    return
(len(self.getPOSX(X,IND))+len(self.getNEGX(X,IND)))/len(self.getU())*100

def roughness(self,X,IND):
    return (len(self.getBNDX(X,IND))/len(self.getU()))*100

def comb(self,inp, temp=[], ans=[]):
    for i in range(len(inp)):
        current = inp[i]
        remaining = inp[i + 1:]
        temp.append(current)
        ans.append(list(temp))
        self.comb(remaining, temp, ans)
        temp.pop()
    return ans

def getReduct(self):
    ind = self.getIND()
    red = []
    comb = self.comb(self.A)
    for a in comb:
        if self.getIND(a) == ind and a != self.A:
            red.append(a)
    return red

```

Программный модуль с реализацией интерфейса для приложения (Golf):

```

# Imports
import sys
from twd import TWD
import PySimpleGUI as sg
import os
import pandas as pd

TITLE = "RS1 algorithm based on RST"

# File for exporting results
EXPORT_FILE = ''

# Reading data from a text file
def read_text_from_file(file_path):
    try:
        with open(file_path) as f:
            rows = []
            for line in f:

```

```

        row = line.split()
        row[0] = str(row[0])
        row[1] = str(row[1])
        row[2] = str(row[2])
        row[3] = str(row[3])
        row[4] = str(row[4])
        row[5] = str(row[5])
        rows.append(row)
    return rows
except:
    print(f'Error reading file {file_path}')
    return ''

# GUI implementation
def main_window():
    sg.theme('SandyBeach')
    layout = [
        [sg.Text(TITLE, justification='center', font=("Helvetica", 24))],
        [sg.Text('_', * 92)],
        [sg.Text('Choose the file (.txt)', font=("Helvetica", 16))],
        [sg.InputText(size=(45, 1), key='file_path'), sg.FileBrowse('Open folder'),
sg.Submit('Load data')],
        [sg.Text('<Buying, Maint, Doors, Persons, Lug_boot, Safety>, decision
value: Class(unacc/acc/good/vgood)')],
        [sg.Multiline(size=(90, 10), key='text')],
        [sg.Text('_', * 92)],
        [sg.Text('Please enter the value of the decisive attribute of dataset: '),
sg.InputText(size=(42, 1), key='decision')],
        [sg.Button('Run the algorithm', button_color='ForestGreen', size = (80,
2))],
        [sg.Output(size=(90, 15), key='-OUTPUT-')],
        [sg.Text('_', * 92)],
        [sg.Text('Enter file name to export results: '), sg.InputText(size=(38, 1),
key = 'export_file'), sg.Button('Export', size=(19, 1))],
        [sg.Cancel('Exit', button_color='OrangeRed', size = (80, 2))]
    ]

# Implementation of the program in response to user actions
window = sg.Window(TITLE, layout)
text = []
df_flag = False
export_text = ''
decision_value = ''
while True:
    event, values = window.read(timeout=400)
    # Implementation of closing program
    if event in (None, 'Exit', sg.WIN_CLOSED):
        sg.popup_ok("The program was closed\nSee you!")
        window.close()
        return None

```

```

# Implementation of loading data into the program
if event == 'Load data':
    df_flag = False
    window['text'].update('')
    window['-OUTPUT-'].update('')
    file_path = values.get('file_path')
    # Selecting the file path
    if not file_path:
        sg.popup_error('Select file path')
        continue
    if file_path.endswith('.txt'):
        text = read_text_from_file(file_path)
        tmp = ''
        for i in range (len(text)):
            tmp = tmp + str(text[i]) + "\n"
        window['text'].update(tmp)
        print("Dataset received")
    else:
        sg.popup_error('Only .txt format is supported!')

# Implementation of algorithm RS1
if event == 'Run the algorithm':
    decision_value = values.get('decision')
    if (len(decision_value)) == 0:
        sg.popup_error('You didn\'t specify a decisive attribute')
        continue
    if text != []:
        window['-OUTPUT-'].update('')
        Test_dt = TWD(text, ["Buying", "Maint", "Doors", "Persons",
"Lug_boot", "Safety"])
        output_res = ''
        print("Please standby\nThe RS1 algorithm is being launched ...")
        print('-' * 150)
        output_res += "U => " + str(Test_dt.getU()) + "\n"
        output_res += "X={x| Class(x)=" + str(f"<{decision_value}>") + "}"
+ " => " + str(Test_dt.getX(decision_value)) + "\n"
        output_res += "Va(Buying)" + str(Test_dt.getVa("Buying")) + "\n"
        output_res += "Va(Maint)" + str(Test_dt.getVa("Maint")) + "\n"
        output_res += "Va(Doors)" + str(Test_dt.getVa("Doors")) + "\n"
        output_res += "Va(Persons)" + str(Test_dt.getVa("Persons")) + "\n"
        output_res += "Va(Lug_boot)" + str(Test_dt.getVa("Lug_boot")) +
"\n"
        output_res += "Va(Safety)" + str(Test_dt.getVa("Safety")) + "\n"
        output_res += "Vd" + str(Test_dt.getVd()) + "\n"
        output_res += "Va" + str(Test_dt.getVa()) + "\n"
        output_res += "IND(A)" + str(Test_dt.getIND()) + "\n"
        output_res += "Object\'s of LowerXA => " +
str(Test_dt.getLowerAX(Test_dt.getX(decision_value), Test_dt.getIND())) + "\n"
        output_res += "Object\'s of UpperXA => " +
str(Test_dt.getUpperAX(Test_dt.getX(decision_value), Test_dt.getIND())) + "\n"

```

```

        output_res += "POS(X)" +
str(Test_dt.getPOSX(Test_dt.getX(decision_value),Test_dt.getIND())) + "\n"
        output_res += "BND(X)" +
str(Test_dt.getBNDX(Test_dt.getX(decision_value),Test_dt.getIND())) + "\n"
        output_res += "NEG(X)" +
str(Test_dt.getNEGX(Test_dt.getX(decision_value),Test_dt.getIND())) + "\n"
        output_res += "Accuracy of Approximation: " +
str(Test_dt.precision(Test_dt.getX(decision_value),Test_dt.getIND())) + "\n"
        output_res += "Quality of classification: " +
str(Test_dt.quality(Test_dt.getX(decision_value),Test_dt.getIND())) + " %" + "\n"
        output_res += "Roughness: " +
str(Test_dt.roughness(Test_dt.getX(decision_value),Test_dt.getIND())) + " %" + "\n"
        output_res += '-' * 150 + "\n"
        output_res += "" + "\n"
        output_res += "Production rules for positive region for decision
value: " + str(f"<{decision_value}>") + "\n"
        output_res +=
str(Test_dt.getRules(Test_dt.getPOSX(Test_dt.getX(decision_value),Test_dt.getIND()
)) + "\n"
        output_res += "\nProduction rules for negative region for decision
value: " + str(f"<{decision_value}>") + "\n"
        output_res +=
str(Test_dt.getRules(Test_dt.getNEGX(Test_dt.getX(decision_value),Test_dt.getIND()
)) + "\n"
        output_res += "\nProduction rules for boundry region for decision
value: " + str(f"<{decision_value}>") + "\n"
        output_res +=
str(Test_dt.getRules(Test_dt.getBNDX(Test_dt.getX(decision_value),Test_dt.getIND()
)) + "\n"
        output_res += "" + "\n"
        output_res += "Reduction:" + str(Test_dt.getReduct())
        print(output_res)
    else:
        sg.popup_error('Enter input data!')
        continue

    if event == 'Export':
        if output_res == '':
            sg.popup_error('First, load/process the dataset')
            continue
        try:
            with open(values.get('export_file'), 'w') as f:
                f.write(output_res)
        except:
            sg.popup_error(f'Error saving to file {EXPORT_FILE}')
        else:
            sg.popup_ok(f"The result is saved in " + values.get('export_file'))

if __name__ == "__main__":
    # ConsoleTest()

```

```

main_window()
sys.exit()

```

Программный модуль с реализацией интерфейса для приложения (Flu):

```

# Imports
import sys
from twd import TWD
import PySimpleGUI as sg
import os
import pandas as pd

TITLE = "RS1 algorithm based on RST"

# File for exporting results
EXPORT_FILE = ''

#Input the dataset from a input.txt file
def ConsoleTest():
    flag = False
    while not flag:
        file_path = input("Please enter the complete correct file path of the
dataset: \n")
        ch = (input("Press 'Y' to confirm \nPress 'N' to enter file path again
\n")).upper()
        if ch == 'Y':
            flag = True
        elif ch == 'N':
            flag = False

    f = open(file_path, 'r', encoding='utf-8')
    rows = []
    for line in f:
        row = line.split()
        row[0] = str(row[0])
        row[1] = str(row[1])
        row[2] = str(row[2])
        rows.append(row)
    print (rows)

# Column titles
Test_dt = TWD(rows,["Headache","Muscle Pain","Temperature"])

# Print results
print("U => ",Test_dt.getU())
print("X={x| Flu(x)=y} =>",Test_dt.getX("y"))
print("Va(Headache)",Test_dt.getVa("Headache"))
print("Va(Muscle Pain)",Test_dt.getVa("Muscle Pain"))
print("Va(Temperature)",Test_dt.getVa("Temperature"))
print("Vd",Test_dt.getVd())

```

```

print("Va",Test_dt.getVa())
print("IND(A)",Test_dt.getIND())
print("IND(Headache)",Test_dt.getIND(["Headache"]))
print("IND(Headache,Muscle Pain)",Test_dt.getIND(["Headache","Muscle Pain"]))
print("IND(Muscle Pain)",Test_dt.getIND(["Muscle Pain"]))
print("IND(Temperature)",Test_dt.getIND(["Temperature"]))
print("IND(Headache,Temperature)",Test_dt.getIND(["Headache","Temperature"]))
print("lowerXA =>",Test_dt.getLowerAX(Test_dt.getX("y"),Test_dt.getIND()))
print("upperXA =>",Test_dt.getUpperAX(Test_dt.getX("y"),Test_dt.getIND()))
print("POS(X)",Test_dt.getPOSX(Test_dt.getX("y"),Test_dt.getIND()))
print("BND(X)",Test_dt.getBNDX(Test_dt.getX("y"),Test_dt.getIND()))
print("NEG(X)",Test_dt.getNEGX(Test_dt.getX("y"),Test_dt.getIND()))
print("Precision of Approximation:
",Test_dt.precision(Test_dt.getX("y"),Test_dt.getIND()))
print("Quality of Approximation:
",Test_dt.quality(Test_dt.getX("y"),Test_dt.getIND()))
print("Roughness: ",Test_dt.roughness(Test_dt.getX("y"),Test_dt.getIND()))
print("")
print("Production rules for positive region:")
print(Test_dt.getRules(Test_dt.getPOSX(Test_dt.getX("y"),Test_dt.getIND())))
print("\nProduction rules for negative region:")
print(Test_dt.getRules(Test_dt.getNEGX(Test_dt.getX("y"),Test_dt.getIND())))
print("\nProduction rules for boundry region:")
print(Test_dt.getRules(Test_dt.getBNDX(Test_dt.getX("y"),Test_dt.getIND())))
print("")
print("Reduction:",Test_dt.getReduct())

# Export results into result.txt file
result_file = open("result.txt", "w", encoding='utf-8')
result_file.write("==== OBJECTS OF LOWER REGION =====\n")
result_file.write(str(Test_dt.getLowerAX(Test_dt.getX("y"),Test_dt.getIND())))
result_file.write("\n\n==== OBJECTS OF UPPER REGION =====\n")
result_file.write(str(Test_dt.getUpperAX(Test_dt.getX("y"),Test_dt.getIND())))
result_file.write("\n\n==== PRECISION OF APPROXIMATION =====\n")
result_file.write("M = (Lower(U) / Upper(U)) = ")
result_file.write(str(Test_dt.precision(Test_dt.getX("y"),Test_dt.getIND())))
result_file.write("\n\n==== PRODUCTION RULES FOR POSITIVE REGION =====")
result_file.write(Test_dt.getRules(Test_dt.getPOSX(Test_dt.getX("y"),Test_dt.ge
tIND()))))
result_file.write("\n\n==== PRODUCTION RULEES FOR NEGARIVE REGION =====")
result_file.write(Test_dt.getRules(Test_dt.getNEGX(Test_dt.getX("y"),Test_dt.ge
tIND()))))
result_file.write("\n\n==== PRODUCTION RULES FOR BOUNDRY REGION =====")
result_file.write(Test_dt.getRules(Test_dt.getBNDX(Test_dt.getX("y"),Test_dt.ge
tIND()))))
result_file.close()

# Reading data from a text file
def read_text_from_file(file_path):
    try:

```

```

        with open(file_path) as f:
            rows = []
            for line in f:
                row = line.split()
                row[0] = str(row[0])
                row[1] = str(row[1])
                row[2] = str(row[2])
                rows.append(row)
            return rows
    except:
        print(f'Error reading file {file_path}')
        return ''

# GUI implementation
def main_window():
    sg.theme('SandyBeach')
    layout = [
        [sg.Text(TITLE, justification='center', font=("Helvetica", 24))],
        [sg.Text('_', * 92)],
        [sg.Text('Choose the file (.txt)', font=("Helvetica", 16))],
        [sg.InputText(size=(45, 1), key='file_path'), sg.FileBrowse('Open folder'),
sg.Submit('Load data')],
        [sg.Text('<Headache, Muscle Pain, Temperature>, decision value:
Flu(yes/no)')],
        [sg.Multiline(size=(90, 10), key='text')],
        [sg.Text('_', * 92)],
        [sg.Text('Please enter the value of the decisive attribute of dataset: '),
sg.InputText(size=(42, 1), key='decision')],
        [sg.Button('Run the algorithm', button_color='ForestGreen', size = (80,
2))],
        [sg.Output(size=(90, 15), key='-OUTPUT-')],
        [sg.Text('_', * 92)],
        [sg.Text('Enter file name to export results: '), sg.InputText(size=(38, 1),
key = 'export_file'), sg.Button('Export', size=(19, 1))],
        [sg.Cancel('Exit', button_color='OrangeRed', size = (80, 2))]
    ]

# Implementation of the program in response to user actions
window = sg.Window(TITLE, layout)
text = []
df_flag = False
export_text = ''
decision_value = ''
while True:
    event, values = window.read(timeout=400)
    # Implementation of closing program
    if event in (None, 'Exit', sg.WIN_CLOSED):
        sg.popup_ok("The program was closed\nSee you!")
        window.close()
        return None

```



```

# Implementation of loading data into the program
if event == 'Load data':
    df_flag = False
    window['text'].update('')
    window['-OUTPUT-'].update('')
    file_path = values.get('file_path')
    # Selecting the file path
    if not file_path:
        sg.popup_error('Select file path')
        continue
    if file_path.endswith('.txt'):
        text = read_text_from_file(file_path)
        tmp = ''
        for i in range (len(text)):
            tmp = tmp + str(text[i]) + "\n"
        window['text'].update(tmp)
        print("Dataset received")
    else:
        sg.popup_error('Only .txt format is supported!')

# Implementation of algorithm RS1
if event == 'Run the algorithm':
    decision_value = values.get('decision')
    if (len(decision_value)) == 0:
        sg.popup_error('You didn\'t specify a decisive attribute')
        continue
    if text != []:
        window['-OUTPUT-'].update('')
        Test_dt = TWD(text, ["Headache", "Muscle Pain", "Temperature"])
        output_res = ''
        print("Please standby\nThe RS1 algorithm is being launched ...")
        print('-' * 150)
        output_res += "U => " + str(Test_dt.getU()) + "\n"
        output_res += "X={x| Flu(x)=" + str(f"<{decision_value}>") + "}" +
" => " + str(Test_dt.getX(decision_value)) + "\n"
        output_res += "Va(Headache)" + str(Test_dt.getVa("Headache")) +
"\n"
        output_res += "Va(Muscle Pain)" + str(Test_dt.getVa("Muscle Pain"))
+ "\n"
        output_res += "Va(Temperature)" + str(Test_dt.getVa("Temperature"))
+ "\n"
        output_res += "Vd" + str(Test_dt.getVd()) + "\n"
        output_res += "Va" + str(Test_dt.getVa()) + "\n"
        output_res += "IND(A)" + str(Test_dt.getIND()) + "\n"
        output_res += "Object\'s of LowerXA => " +
str(Test_dt.getLowerAX(Test_dt.getX(decision_value), Test_dt.getIND())) + "\n"
        output_res += "Object\'s of UpperXA => " +
str(Test_dt.getUpperAX(Test_dt.getX(decision_value), Test_dt.getIND())) + "\n"
        output_res += "POS(X)" +
str(Test_dt.getPOSX(Test_dt.getX(decision_value), Test_dt.getIND())) + "\n"

```

```

        output_res += "BND(X)" +
str(Test_dt.getBNDX(Test_dt.getX(decision_value),Test_dt.getIND())) + "\n"
        output_res += "NEG(X)" +
str(Test_dt.getNEGX(Test_dt.getX(decision_value),Test_dt.getIND())) + "\n"
        output_res += "Accuracy of Approximation: " +
str(Test_dt.precision(Test_dt.getX(decision_value),Test_dt.getIND())) + "\n"
        output_res += "Quality of classification: " +
str(Test_dt.quality(Test_dt.getX(decision_value),Test_dt.getIND())) + " %" + "\n"
        output_res += "Roughness: " +
str(Test_dt.roughness(Test_dt.getX(decision_value),Test_dt.getIND())) + " %" + "\n"
        output_res += '-' * 150 + "\n"
        output_res += "" + "\n"
        output_res += "Production rules for positive region for decision
value: " + str(f"<{decision_value}>") + "\n"
        output_res +=
str(Test_dt.getRules(Test_dt.getPOSX(Test_dt.getX(decision_value),Test_dt.getIND()
)) + "\n"
        output_res += "\nProduction rules for negative region for decision
value: " + str(f"<{decision_value}>") + "\n"
        output_res +=
str(Test_dt.getRules(Test_dt.getNEGX(Test_dt.getX(decision_value),Test_dt.getIND()
)) + "\n"
        output_res += "\nProduction rules for boundry region for decision
value: " + str(f"<{decision_value}>") + "\n"
        output_res +=
str(Test_dt.getRules(Test_dt.getBNDX(Test_dt.getX(decision_value),Test_dt.getIND()
)) + "\n"
        output_res += "" + "\n"
        output_res += "Reduction:" + str(Test_dt.getReduct())
        print(output_res)
    else:
        sg.popup_error('Enter input data!')
        continue

    if event == 'Export':
        if output_res == '':
            sg.popup_error('First, load/process the dataset')
            continue
        try:
            with open(values.get('export_file'), 'w') as f:
                f.write(output_res)
        except:
            sg.popup_error(f'Error saving to file {EXPORT_FILE}')
        else:
            sg.popup_ok(f"The result is saved in " + values.get('export_file'))

if __name__ == "__main__":
    #ConsoleTest()
    main_window()
    sys.exit()

```

Программный модуль с реализацией интерфейса для приложения (Profit):

```
# Imports
import sys
from twd import TWD
import PySimpleGUI as sg
import os
import pandas as pd

TITLE = "RS1 algorithm based on RST"

# File for exporting results
EXPORT_FILE = ''

#Input the dataset from a input.txt file
def ConsoleTest():
    flag = False
    while not flag:
        file_path = input("Please enter the complete correct file path of the
dataset: \n")
        ch = (input("Press 'Y' to confirm \nPress 'N' to enter file path again
\n")).upper()
        if ch == 'Y':
            flag = True
        elif ch == 'N':
            flag = False

    f = open(file_path, 'r', encoding='utf-8')
    rows = []
    for line in f:
        row = line.split()
        row[0] = str(row[0])
        row[1] = str(row[1])
        row[2] = str(row[2])
        rows.append(row)
    print (rows)

# Column titles
Test_dt = TWD(rows, ["Duration", "Competition", "Type", "Crisis"])

# Print results
print("U => ", Test_dt.getU())
print("X={x| Recruit(x)=} =>", Test_dt.getX("Accept"))
print("Va(Duration)", Test_dt.getVa("Duration"))
print("Va(Competition)", Test_dt.getVa("Competition"))
print("Va(Type)", Test_dt.getVa("Type"))
print("Va(Crisis)", Test_dt.getVa("Crisis"))
print("Vd", Test_dt.getVd())
print("Va", Test_dt.getVa())
print("IND(A)", Test_dt.getIND())
```

```

print("IND(Duration)",Test_dt.getIND(["Duration"]))
print("IND(Duration,Competition)",Test_dt.getIND(["Duration","Competition"]))
print("IND(Competition)",Test_dt.getIND(["Competition"]))
print("IND(Type)",Test_dt.getIND(["Type"]))
print("IND(Duration,Type)",Test_dt.getIND(["Duration","Type"]))
print("lowerXA =>",Test_dt.getLowerAX(Test_dt.getX("y"),Test_dt.getIND()))
print("upperXA =>",Test_dt.getUpperAX(Test_dt.getX("y"),Test_dt.getIND()))
print("POS(X)",Test_dt.getPOSX(Test_dt.getX("y"),Test_dt.getIND()))
print("BND(X)",Test_dt.getBNDX(Test_dt.getX("y"),Test_dt.getIND()))
print("NEG(X)",Test_dt.getNEGX(Test_dt.getX("y"),Test_dt.getIND()))
print("Precision of Approximation:
",Test_dt.precision(Test_dt.getX("y"),Test_dt.getIND()))
    #print("Quality of Approximation:
",Test_dt.quality(Test_dt.getX("y"),Test_dt.getIND()))
    #print("Roughness: ",Test_dt.roughness(Test_dt.getX("y"),Test_dt.getIND()))
    print("")
    print("Production rules for positive region:")
    print(Test_dt.getRules(Test_dt.getPOSX(Test_dt.getX("y"),Test_dt.getIND()))
    print("\nProduction rules for negative region:")
    print(Test_dt.getRules(Test_dt.getNEGX(Test_dt.getX("y"),Test_dt.getIND()))
    print("\nProduction rules for boundry region:")
    print(Test_dt.getRules(Test_dt.getBNDX(Test_dt.getX("y"),Test_dt.getIND()))
    print("")
    print("Reduction:",Test_dt.getReduct())

# Export results into result.txt file
result_file = open("result.txt", "w", encoding='utf-8')
result_file.write("==== OBJECTS OF LOWER REGION =====\n")
result_file.write(str(Test_dt.getLowerAX(Test_dt.getX("y"),Test_dt.getIND()))
result_file.write("\n\n==== OBJECTS OF UPPER REGION =====\n")
result_file.write(str(Test_dt.getUpperAX(Test_dt.getX("y"),Test_dt.getIND()))
result_file.write("\n\n==== PRECISION OF APPROXIMATION =====\n")
result_file.write("M = (Lower(U) / Upper(U)) = ")
result_file.write(str(Test_dt.precision(Test_dt.getX("y"),Test_dt.getIND()))
result_file.write("\n\n==== PRODUCTION RULES FOR POSITIVE REGION =====")
result_file.write(Test_dt.getRules(Test_dt.getPOSX(Test_dt.getX("y"),Test_dt.ge
tIND()))
result_file.write("\n\n==== PRODUCTION RULEES FOR NEGARIVE REGION =====")
result_file.write(Test_dt.getRules(Test_dt.getNEGX(Test_dt.getX("y"),Test_dt.ge
tIND()))
result_file.write("\n\n==== PRODUCTION RULES FOR BOUNDRY REGION =====")
result_file.write(Test_dt.getRules(Test_dt.getBNDX(Test_dt.getX("y"),Test_dt.ge
tIND()))
result_file.close()

# Reading data from a text file
def read_text_from_file(file_path):
    try:
        with open(file_path) as f:
            rows = []

```

```

        for line in f:
            row = line.split()
            row[0] = str(row[0])
            row[1] = str(row[1])
            row[2] = str(row[2])
            rows.append(row)
        return rows
    except:
        print(f'Error reading file {file_path}')
        return ''

# GUI implementation
def main_window():
    sg.theme('SandyBeach')
    layout = [
        [sg.Text(TITLE, justification='center', font=("Helvetica", 24))],
        [sg.Text('_', * 92)],
        [sg.Text('Choose the file (.txt)', font=("Helvetica", 16))],
        [sg.InputText(size=(45, 1), key='file_path'), sg.FileBrowse('Open folder'),
sg.Submit('Load data')],
        [sg.Text('<Duration, Competition, Type, Crisis>, decision value:
Prosit(Raise/Fall)')],
        [sg.Multiline(size=(90, 10), key='text')],
        [sg.Text('_', * 92)],
        [sg.Text('Please enter the value of the decisive attribute of dataset: '),
sg.InputText(size=(42, 1), key='decision')],
        [sg.Button('Run the algorithm', button_color='ForestGreen', size = (80,
2))],
        [sg.Output(size=(90, 15), key='-OUTPUT-')],
        [sg.Text('_', * 92)],
        [sg.Text('Enter file name to export results: '), sg.InputText(size=(38, 1),
key = 'export_file'), sg.Button('Export', size=(19, 1))],
        [sg.Cancel('Exit', button_color='OrangeRed', size = (80, 2))]
    ]

# Implementation of the program in response to user actions
window = sg.Window(TITLE, layout)
text = []
df_flag = False
export_text = ''
decision_value = ''
while True:
    event, values = window.read(timeout=400)
    # Implementation of closing program
    if event in (None, 'Exit', sg.WIN_CLOSED):
        sg.popup_ok("The program was closed\nSee you!")
        window.close()
        return None
    # Implementation of loading data into the program
    if event == 'Load data':

```

```

df_flag = False
window['text'].update('')
window['-OUTPUT-'].update('')
file_path = values.get('file_path')
# Selecting the file path
if not file_path:
    sg.popup_error('Select file path')
    continue
if file_path.endswith('.txt'):
    text = read_text_from_file(file_path)
    tmp = ''
    for i in range (len(text)):
        tmp = tmp + str(text[i]) + "\n"
    window['text'].update(tmp)
    print("Dataset received")
else:
    sg.popup_error('Only .txt format is supported!')

# Implementation of algorithm RS1
if event == 'Run the algorithm':
    decision_value = values.get('decision')
    if (len(decision_value)) == 0:
        sg.popup_error('You didn\'t specify a decisive attribute')
        continue
    if text != []:
        window['-OUTPUT-'].update('')
        Test_dt = TWD(text, ["Duration", "Competition", "Type", "Crisis"])
        output_res = ''
        print("Please standby\nThe RS1 algorithm is being launched ...")
        print('-' * 150)
        output_res += "U => " + str(Test_dt.getU()) + "\n"
        output_res += "X={x| Profit(x)=" + str(f"<{decision_value}>") + "}"
+ " => " + str(Test_dt.getX(decision_value)) + "\n"
        output_res += "Va(Duration)" + str(Test_dt.getVa("Duration")) +
"\n"
        output_res += "Va(Competition)" + str(Test_dt.getVa("Competition"))
+ "\n"
        output_res += "Va(Type)" + str(Test_dt.getVa("Type")) + "\n"
        output_res += "Va(Crisis)" + str(Test_dt.getVa("Crisis")) + "\n"
        output_res += "Vd" + str(Test_dt.getVd()) + "\n"
        output_res += "Va" + str(Test_dt.getVa()) + "\n"
        output_res += "IND(A)" + str(Test_dt.getIND()) + "\n"
        output_res += "Object\'s of LowerXA => " +
str(Test_dt.getLowerAX(Test_dt.getX(decision_value), Test_dt.getIND())) + "\n"
        output_res += "Object\'s of UpperXA => " +
str(Test_dt.getUpperAX(Test_dt.getX(decision_value), Test_dt.getIND())) + "\n"
        output_res += "POS(X)" +
str(Test_dt.getPOSX(Test_dt.getX(decision_value), Test_dt.getIND())) + "\n"
        output_res += "NEG(X)" +
str(Test_dt.getNEGX(Test_dt.getX(decision_value), Test_dt.getIND())) + "\n"

```

```

        output_res += "BND(X)" +
str(Test_dt.getBNDX(Test_dt.getX(decision_value),Test_dt.getIND())) + "\n"
        output_res += "Accuracy of Approximation: " +
str(Test_dt.precision(Test_dt.getX(decision_value),Test_dt.getIND())) + "\n"
        output_res += "Quality of classification: " +
str(Test_dt.quality(Test_dt.getX(decision_value),Test_dt.getIND())) + " %" + "\n"
        output_res += "Roughness: " +
str(Test_dt.roughness(Test_dt.getX(decision_value),Test_dt.getIND())) + " %" + "\n"
        output_res += '-' * 150 + "\n"
        output_res += "" + "\n"
        output_res += "Production rules for positive region for decision
value: " + str(f"<{decision_value}>") + "\n"
        output_res +=
str(Test_dt.getRules(Test_dt.getPOSX(Test_dt.getX(decision_value),Test_dt.getIND()
)) + "\n"
        output_res += "\nProduction rules for negative region for decision
value: " + str(f"<{decision_value}>") + "\n"
        output_res +=
str(Test_dt.getRules(Test_dt.getNEGX(Test_dt.getX(decision_value),Test_dt.getIND()
)) + "\n"
        output_res += "\nProduction rules for boundry region for decision
value: " + str(f"<{decision_value}>") + "\n"
        output_res +=
str(Test_dt.getRules(Test_dt.getBNDX(Test_dt.getX(decision_value),Test_dt.getIND()
)) + "\n"
        output_res += "" + "\n"
        output_res += "Reduction:" + str(Test_dt.getReduct())
        print(output_res)
    else:
        sg.popup_error('Enter input data!')
        continue

    if event == 'Export':
        if output_res == '':
            sg.popup_error('First, load/process the dataset')
            continue
        try:
            with open(values.get('export_file'), 'w') as f:
                f.write(output_res)
        except:
            sg.popup_error(f'Error saving to file {EXPORT_FILE}')
        else:
            sg.popup_ok(f"The result is saved in " + values.get('export_file'))

if __name__ == "__main__":
    # ConsoleTest()
    main_window()
    sys.exit()

```

Программный модуль с реализацией интерфейса для приложения (Car evaluation):

```
# Imports
import sys
from twd import TWD
import PySimpleGUI as sg
import os
import pandas as pd

TITLE = "RS1 algorithm based on RST"

# File for exporting results
EXPORT_FILE = ''

# Reading data from a text file
def read_text_from_file(file_path):
    try:
        with open(file_path) as f:
            rows = []
            for line in f:
                row = line.split()
                row[0] = str(row[0])
                row[1] = str(row[1])
                row[2] = str(row[2])
                row[3] = str(row[3])
                row[4] = str(row[4])
                row[5] = str(row[5])
                rows.append(row)
            return rows
    except:
        print(f'Error reading file {file_path}')
        return ''

# GUI implementation
def main_window():
    sg.theme('SandyBeach')
    layout = [
        [sg.Text(TITLE, justification='center', font=("Helvetica", 24))],
        [sg.Text('_' * 92)],
        [sg.Text('Choose the file (.txt)', font=("Helvetica", 16))],
        [sg.InputText(size=(45, 1), key='file_path'), sg.FileBrowse('Open folder'),
sg.Submit('Load data')],
        [sg.Text('<Buying, Maint, Doors, Persons, Lug_boot, Safety>, decision
value: Class(unacc/acc/good/vgood)')],
        [sg.Multiline(size=(90, 10), key='text')],
        [sg.Text('_' * 92)],
        [sg.Text('Please enter the value of the decisive attribute of dataset: '),
sg.InputText(size=(42, 1), key='decision')],
```



```

[sg.Button('Run the algorithm', button_color='ForestGreen', size = (80,
2))],
[sg.Output(size=(90, 15), key='-OUTPUT-')],
[sg.Text('_' * 92)],
[sg.Text('Enter file name to export results: '), sg.InputText(size=(38, 1),
key = 'export_file'), sg.Button('Export', size=(19, 1))],
[sg.Cancel('Exit', button_color='OrangeRed', size = (80, 2))]
]

# Implementation of the program in response to user actions
window = sg.Window(TITLE, layout)
text = []
df_flag = False
export_text = ''
decision_value = ''
while True:
    event, values = window.read(timeout=400)
    # Implementation of closing program
    if event in (None, 'Exit', sg.WIN_CLOSED):
        sg.popup_ok("The program was closed\nSee you!")
        window.close()
        return None
    # Implementation of loading data into the program
    if event == 'Load data':
        df_flag = False
        window['text'].update('')
        window['-OUTPUT-'].update('')
        file_path = values.get('file_path')
        # Selecting the file path
        if not file_path:
            sg.popup_error('Select file path')
            continue
        if file_path.endswith('.txt'):
            text = read_text_from_file(file_path)
            tmp = ''
            for i in range (len(text)):
                tmp = tmp + str(text[i]) + "\n"
            window['text'].update(tmp)
            print("Dataset received")
        else:
            sg.popup_error('Only .txt format is supported!')

    # Implementation of algorithm RS1
    if event == 'Run the algorithm':
        decision_value = values.get('decision')
        if (len(decision_value)) == 0:
            sg.popup_error('You didn\'t specify a decisive attribute')
            continue
        if text != []:
            window['-OUTPUT-'].update('')

```

```

        Test_dt = TWD(text,["Buying","Maint","Doors","Persons",
"Lug_boot","Safety"])
        output_res = ''
        print("Please standby\nThe RS1 algorithm is being launched ...")
        print('-' * 150)
        output_res += "U => " + str(Test_dt.getU()) + "\n"
        output_res += "X={x| Class(x)=" + str(f"<{decision_value}>") + "}"
+ " => " + str(Test_dt.getX(decision_value)) + "\n"
        output_res += "Va(Buying)" + str(Test_dt.getVa("Buying")) + "\n"
        output_res += "Va(Maint)" + str(Test_dt.getVa("Maint")) + "\n"
        output_res += "Va(Doors)" + str(Test_dt.getVa("Doors")) + "\n"
        output_res += "Va(Persons)" + str(Test_dt.getVa("Persons")) + "\n"
        output_res += "Va(Lug_boot)" + str(Test_dt.getVa("Lug_boot")) +
"\n"

        output_res += "Va(Safety)" + str(Test_dt.getVa("Safety")) + "\n"
        output_res += "Vd" + str(Test_dt.getVd()) + "\n"
        output_res += "Va" + str(Test_dt.getVa()) + "\n"
        output_res += "IND(A)" + str(Test_dt.getIND()) + "\n"
        output_res += "Object\'s of LowerXA => " +
str(Test_dt.getLowerAX(Test_dt.getX(decision_value),Test_dt.getIND())) + "\n"
        output_res += "Object\'s of UpperXA => " +
str(Test_dt.getUpperAX(Test_dt.getX(decision_value),Test_dt.getIND())) + "\n"
        output_res += "POS(X)" +
str(Test_dt.getPOSX(Test_dt.getX(decision_value),Test_dt.getIND())) + "\n"
        output_res += "BND(X)" +
str(Test_dt.getBNDX(Test_dt.getX(decision_value),Test_dt.getIND())) + "\n"
        output_res += "NEG(X)" +
str(Test_dt.getNEGX(Test_dt.getX(decision_value),Test_dt.getIND())) + "\n"
        output_res += "Accuracy of Approximation: " +
str(Test_dt.precision(Test_dt.getX(decision_value),Test_dt.getIND())) + "\n"
        output_res += "Quality of classification: " +
str(Test_dt.quality(Test_dt.getX(decision_value),Test_dt.getIND())) + " %" + "\n"
        output_res += "Roughness: " +
str(Test_dt.roughness(Test_dt.getX(decision_value),Test_dt.getIND())) + " %" + "\n"
        output_res += '-' * 150 + "\n"
        output_res += "" + "\n"
        output_res += "Production rules for positive region for decision
value: " + str(f"<{decision_value}>") + "\n"
        output_res +=
str(Test_dt.getRules(Test_dt.getPOSX(Test_dt.getX(decision_value),Test_dt.getIND()
)) + "\n"

        output_res += "\nProduction rules for negative region for decision
value: " + str(f"<{decision_value}>") + "\n"
        output_res +=
str(Test_dt.getRules(Test_dt.getNEGX(Test_dt.getX(decision_value),Test_dt.getIND()
)) + "\n"

        output_res += "\nProduction rules for boundry region for decision
value: " + str(f"<{decision_value}>") + "\n"

```

```

        output_res +=
str(Test_dt.getRules(Test_dt.getBNDX(Test_dt.getX(decision_value),Test_dt.getIND())
)) + "\n"
        output_res += "" + "\n"
        output_res += "Reduction:" + str(Test_dt.getReduct())
        print(output_res)
    else:
        sg.popup_error('Enter input data!')
        continue

    if event == 'Export':
        if output_res == '':
            sg.popup_error('First, load/process the dataset')
            continue
        try:
            with open(values.get('export_file'), 'w') as f:
                f.write(output_res)
        except:
            sg.popup_error(f'Error saving to file {EXPORT_FILE}')
        else:
            sg.popup_ok(f"The result is saved in " + values.get('export_file'))

if __name__ == "__main__":
    # ConsoleTest()
    main_window()
    sys.exit()

```