МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Інтелектуальна система оцінки якості навчального контенту випускової кафедри»

Завідувач випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Шелехов І.В.**

Студента групи ІН.м–92 **Суріна Д.О.**

СУМИ 2020

Сумський державний університет
(назва вузу)

Факультет *ЕЛІТ* Кафедра *Комп'ютерних наук.*
Спеціальність *«Інформатика»*

Затверджую:
зав.кафедрою _____
_____
"_____" _____ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Суріну Дмитрію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інтелектуальна система оцінки якості навчального контенту випускової кафедри.*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____

3. Вхідні данні до проекту (роботи) _____
_____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз предметної області 2) Технологія нечіткого логічного виводу Мамдані та дерева прийняття рішень ід3 3) Розробка інформаційного та програмного забезпечення системи*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
_____
_____
_____
_____
_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	*Аналіз предметної області*		
2.	*Технологія нечіткого логічного виводу Мамдані та дерева прийняття рішень id3*		
3.	*Розробка інформаційного та програмного забезпечення системи*		

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 64 стор., 13 рис., 2 додатки, 19 джерел.

Об’єкт дослідження — процес оцінки функціонального стану навчальних матеріалів для підвищення якості освіти.

Мета роботи — розробка інформаційного та програмного забезпечення системи нечіткого логічного виводу та дерева прийняття рішень для удосконалення навчальних матеріалів кафедри «Комп’ютерних наук» та адаптації їх до умов ринку праці.

Методи дослідження — теорія нечітких множин, дерева прийняття рішень.

Результати — Розроблено систему оцінки якості навчального контенту з використанням нечіткого логічного виводу Мамдані в рамках теорії нечітких множин та дерева прийняття рішень, а також її програмна реалізація. Розроблена система реалізована за допомогою середовища C#.

СИСТЕМА НЕЧІТКОГО ЛОГІЧНОГО ВИВОДУ, ФУНКЦІЯ ПРИНАЛЕЖНОСТІ, ДЕРЕВО ПРИЙНЯТТЯ РІШЕНЬ, ТЕОРІЯ НЕЧІТКИХ МНОЖИН, C#.

ЗМІСТ

ВСТУП

Навколо проблеми якості освіти ведуться нескінченні обговорення та суперечки. Боротьба за якість освіти висувається як провідне завдання в діяльності освітніх установ. Важливою особливістю сучасної системи освіти є існування інноваційної стратегії організації навчання, яка визначає якість освіти та імідж будь-якого навчального закладу. Під якістю освіти розуміється можливість навчального контенту відповідати заявленим нормам державного стандарту та соціального замовлення, а також його адаптації до життєвих умов соціально-економічного та науково-технічного прогресу [1].

В поняття якості освітнього процесу можуть входити фактори, що безпосередньо можуть впливати на ефективність протікання процесу, такі як:

– зміст навчального контенту;

– менеджмент освітнього процесу;

– навчально-методична та матеріально-технічна забезпеченість навчального процесу;

– технологія навчального процесу;

– якісний склад викладачів.

Відповідно можна виділити фактори, які в більшій мірі будуть визначати поняття якості освітнього процесу. Такими факторами виступають зміст навчального контенту та якісний склад викладачів.

В багатьох випадках на базі досвіду методичної роботи кваліфікованих викладачів, можна визначити, що галузь освіти має проблеми з адаптацією змісту та наповнення навчального контенту до вимог ринку праці.

Одними із ключових вимог діяльнісного підходу в навчанні студентів є чітко виражена направленість навчального процесу на розвиток навичок логічного мислення у учнів при прийнятті рішень пов’язаних з характером майбутньої професійної діяльності. Реалізація даної задачі в нових навчальних програмах потребує удосконалення існуючої системи контроля якості навчального контенту. До прогресивних методів контроля якості відносять модульно-рейтинговий метод як спосіб оцінки знань, вмінь та навичок.

Модульно-рейтингова технологія навчання – це така організація навчального процесу, в якій зміст навчання представляється в вигляді самостійних, закінчених модулів, які несуть в собі одночасно інформацію і методичні вказівки щодо її застосування, а оцінювання успішності здійснюється за допомогою рейтингової системи оцінювання знань. Основні принципи даного навчання полягають в мобільності, структуризації змісту навчання, динамічності, гнучкості, усвідомленні перспектив [3].

Отже, створення системи оцінки якості навчання є важливим стратегічним рішенням для проведення аналізу поточного функціонального стану навчального контенту, що дає змогу вчасно скоригувати навчальний матеріал для підвищення результатів. Ця система дозволяє встановити корисність, яку буде отримувати організація майбутнього працедавця від навчання випускників, допомагає визначити ефективність використання тої чи іншої форми навчання та актуальність тематики навчального матеріалу відносно ринку праці. Результат оцінки якості навчального контенту потребує аналізу та подальшого використання для розробки та удосконалення нових навчальних програм у майбутньому. Підвищення якості навчального контенту дає можливість відмовлятися від старих навчальних програм, які не мають такої високої результативності.

Аналітично-інформаційна система оцінки навчального контенту повинна бути розроблена з використанням інтелектуальної складової для забезпечення чіткого результату при оцінці функціонального стану.

Головним завданням роботи є формування в якості вхідних даних нечіткої множини та навчальної матриці для системи нечіткого логічного виводу, яка повинна містити у собі математично формалізовані нечіткі дані на основі оцінок змістовних модулів від роботодавців та випускників, та дерева прийняття рішень для оцінки якості навчання. Підготовка та аналіз математичних моделей отриманих за допомогою нечітких множин, функції приналежності, нечіткої бази знань, методів дефазифікації та дерева прийняття рішень, які будуть використовуватися в якості інтелектуальної складової в розроблюваній системі оцінки якості навчального контенту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження актуальності проблеми

Важливу роль у сучасних інформаційних системах освіти відіграє персоніфікація, яка реалізована шляхом регулярного тестування оцінки рівня підготовки, ступеня і швидкості освоєння матеріалу.

Підвищення ефективності керування є актуальною проблемою в умовах підвищуваної складності технологічного обладнання, процесів та систем. Для проектування систем керування складними об’єктами важливу роль відіграє вирішення задач побудови адекватних математичних або імітаційних моделей та синтезу алгоритмів керування, що забезпечують вирішення задач в умовах невизначеності [6]. Інтелектуальні системи прийняття рішень (ІСПР), знаходять більш широке використання при керуванні та формалізації складних об’єктів та систем. Одним з ефективних підходів, використовуваних при синтезі сучасних ІСПР, являється використання теорії нечітких множин та нечіткої логіки. Оперування нечіткими поняттями в теорії нечітких множин максимально наближає опис об’єкта керування до звичайної людської мови, що значно спрощує побудову моделі системи.

ІСПР, що побудовані на нечіткій логіці здатні вирішувати багато різноманітних задач керування в умовах невизначеності. При розробці даних систем доцільно проектувати програмний модуль, який реалізує алгоритм нечіткого логічного виводу під конкретне завдання, або використовувати готові програмні розробки, попередньо адаптуючи їх для вирішення поставленого завдання.

1.2 Аналіз аналогів

Аналіз існуючого програмного забезпечення для обробки нечіткої інформації дозволяє виділити дві основні групи ПО такого типу:

– спеціалізовані комерційні програмні продукти, що використовують апарат нечіткої логіки для вирішення задач в різних галузях. Такі продукти не призначені для взаємодії з іншими програмами у реальному часі (Matlab, Matematica, CubiCalc, RuleMaker, FuziCalc, fuzzyTECH та ін.)

– невеликі бібліотеки класів з обмеженими можливостями налаштування, або розробленими під конкретний програмний продукт (Free Fuzzy Logic Library, fuzzyCLIPS).

Система Matlab створена спеціально для проведення інженерних розрахунків та є вельми використовуваною у навчальному процесі вищих навчальних закладів. Математичний апарат, що в ній використовується, максимально наближений до сучасного математичного апарату інженера чи вченого та опирається на розрахунки з матрицями, векторами, комплексними числами. Графічне уявлення функціональних залежностей організовано в ній у формі, яку потребує інженерна документація. Мова програмування системи Matlab має просту структуру, вона складає лише декілька десятків операторів, натомість це компенсується великою кількістю процедур та функцій, склад яких є легкозрозумілим користувачеві з відповідною інженерною підготовкою. Matlab – є відкритою системою, це означає що практично усі процедури та функції доступні не тільки для використання, але й для коригування та модифікування. Вона може гнучко розширюватись за вимогою користувача створеними їм програмами та процедурами для рішення потрібних класів задач. Також вона має можливість використовувати практично усі розрахункові можливості системи у режимі потужного наукового калькулятору, що дає можливість складати власні алгоритми з метою багаторазового їх використання для досліджень. Ця можливість робить

систему незамінним інструментом для проведення наукових розрахункових досліджень. Відносно усіх вище перелічених особливостей можна констатувати, що система має великий спектр можливостей та по швидкості виконання задач вона опереджає багато інших схожих рішень [8].

Одним із лідируючих засобів розробки програмного забезпечення для розрахунків із застосуванням нечіткої логіки і нечітких нейронних мереж є пакет fuzzyTECH корпорації Inform Software Corporation, який досить успішно використовується фахівцями в Японії, Європі і США. FuzzyCLIPS – це нечітке логічне розширення оболонки експертної системи CLIPS (C Language Integrated Production System). Воно розширює можливості CLIPS, надаючи можливість нечіткого виводу, яка повністю інтегрована з фактами CLIPS та механізмом виводу, дозволяючи представляти та керувати нечіткими фактами та правилами. FuzzyCLIPS може мати справу з точними, нечіткими (або неточними) та комбінованими міркуваннями, дозволяючи вільно змішувати нечіткі і нормальні терміни в правилах і фактах експертної системи. В системі використовуються два основних неточних поняття: нечіткість і невизначеність. Дана бібліотека містить в собі всі необхідні засоби для розробки та аналізу створюваних нечітких систем, а також забезпечує зручну роботу на всіх стадіях проектування. Це все досягається завдяки використанню зручного графічного інтерфейсу користувача і технології «Point-and-Click». Наочне задання правил логічного висновку формату "ЯКЩО-ТО" в діалоговому режимі спілкування з користувачем дозволяє оптимізувати розроблювану систему, що в кінцевому рахунку призводить до підвищення її ефективності в цілому [2].

Перераховані продукти не володіють достатнім рівнем універсальності, щоб їх компоненти можна було б програмно вбудовувати при побудові власних розробок та гнучко вбудовувати в залежності від вимог системи.

1.3 Вибір засобів реалізації

Для вирішення задачі класифікації оцінки якості освіти за визначеною шкалою можна частково використати алгоритми кластер-аналізу. Під кластером зазвичай розуміється частина даних (в типовому випадку – підмножина об'єктів або підмножина змінних, або підмножина об'єктів, що характеризуються підмножиною змінних), яка виділяється з решти наявністю деякої однорідності її елементів. У найпростішому випадку мова йде про подібність елементів, в ідеальному випадку - про співпадаючі значення основних змінних чи іншого роду близькості, яка виражається у геометричній близькості відповідних об'єктів [12].

Процес класифікації об’єктів без використання навчальної множини називається автоматичною класифікацією або ж класифікацією без вчителя. Такий метод самонавчання є широко розповсюдженим в інтелектуальних системах, зокрема в експертних системах розпізнавання образів та класифікації. У даному випадку в алгоритмі дерева прийняття рішень буде використовуватись підхід розпізнавання даних за класифікованими навчальними матрицями, тобто навчання з “учителем”.

Кластерний аналіз застосовують, коли за статистичними даними необхідно розділити елементи вибірки на групи. Причому два елементи групи з однієї і тієї ж групи повинні бути «близькими» за сукупністю значень виміряних у них ознак, а два елементи з різних груп повинні бути «далекими» в тому ж сенсі. Методи кластерного аналізу можна розділити на дві групи:

- ієрархічні
- неієрархічні

Кожна з груп включає безліч підходів і алгоритмів. Використовуючи різні методи кластерного аналізу, можливо отримати різні рішення для одних і тих же даних. Це вважається нормальним явищем. Суть ієрархічної кластеризації полягає в послідовному об'єднанні менших кластерів в великі або поділі великих кластерів на менші. Група агломеративних методів

ієрархічної кластеризації характеризується послідовним об'єднанням вихідних елементів і відповідним зменшенням числа кластерів. На початку роботи алгоритму всі об'єкти є окремими кластерами. На першому кроці найбільш схожі об'єкти об'єднуються в кластер. На наступних кроках об'єднання триває до тих пір, поки всі об'єкти не будуть складати один кластер [17].

Для неієрархічної кластеризації використовується алгоритм k-середніх. Алгоритм k-середніх будує k кластерів, розташованих на можливо великих відстанях один від одного. Основний тип задач, які вирішує алгоритм k-середніх, - наявність припущень (гіпотез) щодо числа кластерів, при цьому вони повинні бути різні настільки, наскільки це можливо. Вибір числа k може базуватися на результатах попередніх досліджень, теоретичних міркуваннях або інтуїції. Загальна ідея алгоритму: заданий фіксоване число k кластерів спостереження зіставляються кластерам так, що середні в кластері (для всіх змінних) максимально можливо відрізняються один від одного [10].

В цілому проблема розпізнавання образів складається з двох частин: навчання та розпізнавання. Навчання здійснюється шляхом співвідношення окремих об'єктів із зазначенням їх приналежності того чи іншого класу. В результаті навчання система повинна набувати здатності реагувати однаковими реакціями на всі об'єкти одного класу і різними - на всі об'єкти різних класів. Дуже важливо щоб процес навчання завершився тільки шляхом показів кінцевого числа об'єктів без будь-яких інших підказок. В якості об'єктів навчання можуть бути або картинки, або інші візуальні зображення (літери), або різні явища зовнішнього світу, наприклад звуки, стану організму при медичному діагнозі, стан технічного об'єкта в системах управління і ін. Важливо, що в процесі навчання вказуються тільки самі об'єкти і їх приналежність класу. За навчанням відбувається процес розпізнавання нових об'єктів, який характеризує дії вже навченої системи. Автоматизація цих процедур і становить проблему навчання розпізнаванню образів [19].

Розглянемо методи розпізнавання образів у рамках статистичного підходу:

– лінгвістичний метод (якщо опис образів здійснюється за допомогою підобразів і їх співвідношень, то для конструювання системи розпізнавання використовують лінгвістичний або синтаксичний підхід з використанням принципу загальності властивостей. Основне припущення, яке робиться в цьому методі, ґрунтується на тому, що образи, які належать одному і тому ж класу, володіють рядом загальних властивостей або ознак, які відображають подібність таких образів. Ці загальні властивості можна частково ввести в пам'ять системи розпізнавання. Коли системі представити некласифікований образ, то вибирається набір ознак, що визначаються, причому останні інколи кодуються, а потім вони порівнюються з ознаками, закладеними в пам'яті системи розпізнавання. При використанні даного методу основна задача полягає у виділенні загальних властивостей за вибором образів, належність яких шуканому класу відома);

– математичний метод (в основу математичного підходу покладені правила класифікації, які формуються і виводяться в рамках визначеного математичного формалізму з допомогою принципів загальності властивостей і кластеризації. Коли образи деякого класу представляють собою вектори, компонентами яких є дійсні числа, то цей клас можна розглядати як кластер. Побудова системи розпізнавання, яка базується на реалізації даного принципу, визначається просторовим розміщенням окремих кластерів. Якщо кластери, що відповідають різним класам, рознесені далеко один від одного, то можна користуватися простими схемами розпізнавання, наприклад, класифікацією за принципом мінімальної відстані);

– евристичний метод (за основу евристичного підходу взяті інтуїція і досвід людини: в ньому використовуються принципи перерахування членів класу і загальні властивості. Зазвичай системи, побудовані такими методами, включають набір специфічних процедур, розроблених для конкретних задач

розпізнавання. Хоча евристичний підхід відіграє велику роль в розпізнаванні, небагато може бути сказано про загальні принципи синтезу, оскільки розв’язання кожної конкретної задачі потребує використання специфічних прийомів розробки. Це означає, що структура і якість евристичної системи в значній мірі визначається талантом та досвідом роботи тих, хто її розробляє).

Основний недолік більшості відомих методів автоматичної класифікації полягає в ігноруванні перетину класів розпізнавання, що має місце на практиці в задачах керування слабо формалізованими процесами з нечіткими даними. Тому останнім часом спостерігається збільшення уваги до розробки алгоритмів автоматичної класифікації у рамках теорії нечітких множин. Але при цьому все ще поза увагою досліджень залишається таке важливе питання, як оптимізація просторово-часових параметрів навчання інтелектуальної системи [5].

В основі цієї теорії лежить уявлення про те, що складові даної множини є елементи, які володіють загальною властивістю, можуть володіти цією властивістю в різному ступені і, відповідно, належати до даної множини з різним ступенем. При такому підході вислів "цей елемент належить даній множині" втрачає сенс, оскільки необхідно вказати "наскільки сильно" або з яким ступенем конкретний елемент задовільняє властивостям даної множини.

Таким чином є елементи, що задовільняють дану множину в повному, стовідсоткову або одиничному ступені, також є елементи, що не належать множині, іншими словами належать їй у нульовому ступені та ті які належать множині у деякому ступені тобто між 0 та 1.

Нехай є деяка множина X. Якщо задана деяка звичайна, чітка підмножина A множини X, $A \subset X$, тоді по кожному елементу X можливо сказати, належить він до A чи ні. Для будь-якої підмножини A множини X введемо термін «індикатор», або характеристичної функції. Ця функція визначається наступним чином:

$$\chi_A(x) = \begin{cases} 1, & x \in A, \\ 0, & x \notin A. \end{cases}$$

Якщо на множині X задана функція Ха, яка може приймати усього два значення 1 та 0, відповідно така функція у повній мірі визначає множину А наступним чином:

$$A = \{x \in X \mid \chi(x) = 1\}.$$

Відповідно, множину та її індикатор можливо ототожнити. Сама ідея нечітких множин полягає в тому щоб дозволити індикатору приймати не тільки значення 1 та 0, а й будь-які проміжні значення [9].

Системи, що засновані на нечітких множинах розроблені і успішно впроваджені в таких областях, як: управління технологічними процесами, управління транспортом, медична діагностика, технічна діагностика, фінансовий менеджмент, біржове прогнозування, розпізнавання образів. Спектр додатків дуже широкий - від відеокамер і побутових пральних машин до засобів наведення ракет ППО і управління бойовими вертольотами.

Практичний досвід розробки систем нечіткого логічного виводу свідчить, що терміни і вартість їх проектування значно менше, ніж при використанні традиційного математичного апарату, при цьому забезпечується необхідний рівень працездатності і прозорості моделей. Тому будемо використовувати цей підхід для розроблюваної системи.

Для реалізації програмного продукту інтелектуальних систем часто використовують мову програмування Python. По суті, машинне навчання - це технологія, яка допомагає додаткам на основі штучного інтелекту навчатися і видавати результати автоматично, без людського втручання.

Робота фахівця по машинному навчанню полягає в збиранні, систематизуванні та аналізі даних, а потім на основі отриманої інформації

створювати алгоритми для штучного інтелекту. Python найкраще підходить для виконання таких завдань, тому що він досить зрозумілий в порівнянні з іншими мовами. Більш того, у нього відмінна продуктивність при обробці даних [16]. Одна з основних причин, чому Python використовується для машинного навчання полягає в тому, що у нього є безліч фреймворків, які спрощують процес написання коду і скорочують час на розробку.

Python є інтерпретованою, початково об’єктно-орієнтованою мовою програмування. Одною із її особливостей є те, що інтерпретатор Python реалізований практично на всіх платформах і операційних системах.

Іншою особливістю є наявність великої кількості з’єднувальних до програми модулів, що забезпечують різні додаткові можливості:

Numerical Python – розширені математичні можливості, такі як маніпуляції з цілими векторами і матрицями;

Tkinter – побудова додатків з використанням графічного інтерфейсу користувача (GUI) на основі широко розповсюдженого на X-Windows Tk-інтерфейсу;

OpenGL – використання великої бібліотеки графічного моделювання дво- і тривимірних об’єктів Open Graphics Library.

В якості програмного забезпечення для реалізації системи буде використовуватись середа розробки Microsoft Visual Studio, інтерфейс програмування додатків API, що відповідає за графічний інтерфейс користувача, та мови програмування C# з використанням методів ООП.

Інтегроване середовище розробки Visual Studio - це стартовий майданчик для написання, налагодження і складання коду, а також публікації додатків. Інтегроване середовище розробки (IDE) являє собою багатофункціональну програму, яку можна використовувати для різних додатків програмного забезпечення. Крім стандартного редактора і стандартних засобів розробки, Visual Studio включає в себе компілятори, засоби автозавершення коду, графічні конструктори і багато інших функцій

для спрощення процесу розробки [15]. Середовище розробки програмних проектів є відкритою мовною середою. Це означає, що поряд з мовами програмування, включеними в середу фірмою Microsoft - Visual C ++ .Net (з керованими розширеннями), Visual C # .Net, Visual Basic .Net, - в середу можуть додаватися будь-які мови програмування, компілятори яких створюються іншими фірмами. Відкритість середовища не означає повної свободи. Всі розробники компіляторів при включенні нової мови в середу розробки повинні слідувати певним обмеженням. Головне обмеження, яке можна вважати і головним достоїнством, полягає в тому, що всі мови, що включаються в середу розробки Visual Studio .Net, повинні використовувати єдиний каркас - Framework .Net. Завдяки цьому досягаються багато бажані властивості: легкість використання компонентів, розроблених на різних мовах; можливість розробки кількох частин однієї програми на різних мовах; можливість безшовної налагодження такого додатка; можливість написати клас на одній мові, а його нащадків - на інших мовах. Єдиний каркас призводить до зближення мов програмування, дозволяючи разом з тим зберігати їх індивідуальність і наявні у них гідності. Подолання мовного бар'єру - одна з найважливіших завдань сучасного світу. Visual Studio .Net, завдяки єдиному каркасу, до певної міри вирішує цю задачу в світі програмістів [7].

Об'єктно-орієнтоване програмування – це такий підхід до написання програм, який ґрунтується на об'єктах, а не на функціях і процедурах. Ця модель ставить в центр уваги об'єкти, а не дії, дані, а не логіку. Об'єкт - реалізація класу. Все реалізації одного класу схожі один на одного, але можуть мати різні параметри і значення. Об'єкти можуть задіяти методи, специфічні для них. ООП сильно спрощує процес організації та створення структури програми. Окремі об'єкти, які можна міняти без впливу на інші частини програми, спрощують також і внесення в програму змін. Так як з плином часу програми стають все більшими, а їх підтримка все більш важкою, ці два

аспекти ООП стають все більш актуальними [13]. Наведемо основні концепції ООП:

- абстракція даних: подробиці внутрішньої логіки приховані від кінцевого користувача. Користувачеві не потрібно знати, як працюють ті чи інші класи і методи, щоб їх використовувати;
- спадкування: найпопулярніший принцип ООП. Спадкування дозволяє повторне використання коду - якщо якийсь клас вже має якусь логіку і функції, нам не потрібно переписувати все це заново для створення нового класу, ми можемо просто включити старий клас в новий, повністю;
- інкапсуляція: включення в клас об'єктів іншого класу, питання доступу до них, їх видимості
- поліморфізм: це властивість одних і тих же об'єктів і методів приймати різні форми.

Використання об’єктно-орієнтованої методології розробки програмного забезпечення надає розробнику значну перевагу на етапах його створення, відладки, впровадження та супроводу за рахунок декомпозиції системи на окремі об’єкти, відокремлення опису внутрішньої складової об’єктів від протоколів взаємодії між ними (інкапсуляція), побудови ієрархії об’єктів (наслідування), повторного використання коду, реалізації особливостей поведінки близьких по функціональності об’єктів (поліморфізм).

1.4 Постановка задачі

Метою створення системи оцінки якості навчального контенту є розроблення інформаційного та програмного забезпечення системи нечіткого логічного виводу у рамках теорії нечітких множин та розробка алгоритму для побудови дерева рішень на основі значень оцінок від роботодавців та випускників у вигляді навчальної матриці.

Розробимо користувацьку функцію приналежності за даною формулою:

$$\mu_A(x) = \frac{1}{1 + \left| \frac{x - m}{\sigma} \right|^{2a}}$$

Рисунок 2.1 – Формула користувацької функції приналежності

Надамо три набори значень параметрів центру (m) та ширини (σ) функції належності для подання трьох лінгвістичних значень у вигляді нечітких множин та відобразимо їх на графіку. Першим параметром буде значення оцінки змістовних модулів роботодавцями, а другим – випускниками. В якості лінгвістичних значень буде розумітися якість навчального контенту наприклад: контент отримав оцінку «відмінно», «добре» або «задовільно». Використаємо розроблену функцію приналежності в алгоритмі нечіткого логічного виводу Мамдані.

Побудуємо дерево прийняття рішень на основі матриці експертних даних. Для цього треба сформувати апаратний комплекс для взаємодії з вхідними даними, обробки даних та формування результату.

Матрицею називають прямокутну таблицю, заповнену числами. Найважливіші характеристики матриці - число рядків і число стовпців. Позначають матриці великими латинськими літерами. Самі числа називають елементами матриці і характеризують їх становищем в матриці, задаючи номер рядка і номер стовпця і записуючи їх у вигляді подвійного індексу, причому спочатку записують номер рядка, а потім стовпчика [4]. В якості навчальної матриці для побудови дерева рішень буде використовуватись

матриця A розміру M x N, що представляє собою прямокутну таблицю цілих чисел від 0 до 100, які містяться в M рядках та N стовпцях:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Де aij (i =1, …, m; j =1, …, n) – елементи матриці A. Перший індекс i – номер рядку, другий індекс j – номер стовпця, на перехресті яких знаходиться елемент aij.

Згідно з алгоритмом побудови дерева рішень за алгоритмом ІД3 спочатку потрібно програмно реалізувати метод вибору атрибуту розбиття множини значень у вузлі за алгоритмом теоретико-інформаційного критерію, який оснований на поняттях інформаційної ентропії.

$$H = -\sum_{i=1}^{n} \frac{N_i}{N} \log\left(\frac{N_i}{N}\right)$$

Рисунок 2.5 – Алгоритм теоретико-інформаційного критерію

Де n – число класів в вихідній підмножині, N_i – число множин значень i-го класу, N – загальне число множин в підмножині.

Таким чином, найкращим атрибутом розбиття буде той, який забезпечить максимальне зниження ентропії результуючої підмножини відносно батьківської, тобто ми повинні отримати максимальний приріст інформації. Найкращий атрибут потрібно визначити як рішення для вузла, для кожного значення атрибуту створити нащадок вузла. Також треба сформувати методи для вводу даних навчальних прикладів та відсортувати їх по листкам. Якщо навчальний приклад є ідеально класифікованим, то треба використати метод для зупину алгоритму.

Для досягнення поставленої мети необхідно в дипломній роботі вирішити такі завдання:

1) розробка інтерфейсу для взаємодії з користувачем;

2) вибір логіко-математичного алгоритму за яким буде відбуватися функціонування інтелектуальної складової системи нечіткого логічного виводу;

3) в якості функції належності для всіх вхідних та вихідних лінгвістичних змінних використати користувацьку функцію приналежності;

4) формування правил для нечіткої бази знань;

5) розробка та програмна реалізація алгоритму фазифікації вхідних даних для зведення правил із нечіткої бази знань та масиву вхідних змінних універсальної множини $\{d_1, d_2, \ldots, d_m\}$ до нечіткої множини на інтервалі $[\underline{y}, \bar{y}]$ в логіко-математичному алгоритмі;

6) розробка та програмна реалізація алгоритму агрегування;

7) розробка та застосування алгоритмів дефазифікації вихідної змінної y, яка відповідає вхідному вектору x^*;

8) побудова дерева рішень на основі матриці експертних даних;

9) перевірка працездатності розроблених алгоритмів;

2 ТЕХНОЛОГІЯ НЕЧІТКОГО ЛОГІЧНОГО ВИВОДУ МАМДАНІ ТА ДЕРЕВА ПРИЙНЯТТЯ РІШЕНЬ ІДЗ

2.1 Основні принципи нечіткого логічного виводу Мамдані

Нечіткий логічний вивід – це апроксимація залежностей «входи-вихід» на базі лінгвістичних виразів типу «ЯКЩО-ТО» та операцій над нечіткими множинами. Основні положення теорії нечітких множин приведемо на рисунку:

Рисунок 2.2 – Типова структура моделі на основі нечіткого логічного виводу

Фазифікатор – перетворює фіксований вектор впливаючих факторів X в вектор нечітких множин $\tilde{X}$, необхідних для виконання нечіткого логічного виводу.

Нечітка база знань представляє собою інформацію про залежність Y = f(X) у вигляді лінгвістичних правил типу «ЯКЩО-ТО».

Машина нечіткого логічного виводу – на основі правил бази знань визначає значення вихідної змінної у вигляді нечіткої множини $\tilde{Y}$, що відповідає нечітким значенням вхідних змінних $\tilde{X}$ [11].

Дефазифікатор – перетворює вихідну нечітку множину $\tilde{Y}$ в чітке число Y.

Функцією приналежності (membership function) називається функція, яка дозволяє обчислити ступінь приналежності довільного елемента універсальної множини до нечіткої множини.

Лінгвістичною змінною (linguistic variable) називається змінна, значеннями якої можуть бути слова або словосполучення деякої природної або штучної мови.

Нечіткою множиною *(fuzzy set)* $\tilde{A}$ на універсальній множині U називається сукупність пар ($\mu_A(u), u$), де $\mu_A(u)$ - ступінь приналежності елемента $u \in U$ до нечіткій множині $\tilde{A}$. Ступінь приналежності - це число з діапазону [0, 1]. Чим вище ступінь приналежності, тим більшою мірою елемент універсальної множини відповідає властивостям нечіткої множини. Якщо універсальна множина складається з кінцевої кількості елементів $U = \{u_1, u_2, \dots u_k\}$, тоді нечітка множина $\tilde{A}$ записується у вигляді $\tilde{A} = \sum_{i=1}^{k} \mu_A(u_i)/u_i$ знаки Σ та $\int$ в цій формулі означають сукупність пар $\mu_A(u)$ і U.

Термом (term) називається будь-який елемент терм-множини. В теорії нечітких множин терм формалізується нечіткою множиною за допомогою функції приналежності.

Використаємо для розроблюваної системи нечіткого логічного виводу алгоритм Мамдані.

$$\bigcup_{p=1}^{k_j} \left(\bigcap_{i=1}^{n} x_i = a_{i,jp} \ \text{ с весом } w_{jp} \right) \rightarrow y = d_j, \quad j = \overline{1,m}$$

Рисунок 2.3 – Формула алгоритму Мамдані

Даний алгоритм виконує нечіткий логічний висновок на основі значень нечіткої бази знань, відповідно розробимо її з урахуванням поточних вимог до поставленої задачі. Одна з яких вимагає, щоб значення вхідних і вихідних змінних були задані нечіткими множинами.

В якості функції належності для всіх вхідних та вихідних лінгвістичних змінних використаємо користувацьку функцію приналежності.

Введемо наступні позначення, що необхідні для опису роботи алгоритму Мамдані, в яких значення вхідних і вихідних змінних задані нечіткими множинами:

$\mu_{jp}(x_j)$ – функція приналежності входу x_j нечіткому терму $a_{i,jp}$, тобто

$$a_{i,jp} = \int_{\underline{x}_i}^{\bar{x}_i} \mu_{jp}(x_i)/x_i, \quad x_j \in \left[\underline{x}_j, \bar{x}_j\right]$$

$\mu_{d_j}(y)$ - функція приналежності виходу y нечіткому терму d_j, т.е.

$$d_j = \int_{\underline{y}}^{\bar{y}} \mu_{d_j}(y)/y, \quad y \in \left[\underline{y}, \bar{y}\right]$$

Ступені приналежності вхідного вектора $X^* = \left(x_1^*, x_2^*, \ldots x_n^*\right)$ нечітким термам d_j із бази знань розраховується наступним чином:

$$\mu_{d_j}\left(X^*\right) = \bigvee_{p=\overline{1,k_j}} w_{jp} \cdot \bigwedge_{i=\overline{1,n}} \left[\mu_{jp}\left(x_i^*\right)\right], \; j = \overline{1,m}$$

де $\vee(\wedge)$ - операція з s-норми (t-норми), тобто з множини реалізацій логічної операції АБО (І). Найбільш часто використовуються наступні реалізації: для операції АБО - знаходження максимуму і для операції І - знаходження мінімуму.

В результаті отримуємо таку нечітку множину, відповідно вхідному вектору X^*:

$$\tilde{y} = \frac{\mu_{d_1}(X^*)}{d_1} + \frac{\mu_{d_2}(X^*)}{d_2} + \ldots + \frac{\mu_{d_m}(X^*)}{d_m}$$

Особливістю цієї нечіткої множини є те, що універсальною множиною для нього є терм-множина вихідних змінних y. Такі нечіткі множини називаються нечіткими множинами другого порядку.

Для переходу від нечіткої множини, заданої на універсальній множині нечітких термів $\{d_1, d_2, \ldots, d_m\}$ до нечіткої множини на інтервалі $[\underline{y}, \bar{y}]$ необхідно: 1) "зрізати" функції приналежності $\mu_{d_j}(y)$ на рівні $\mu_{d_j}(X^*)$;
2) об'єднати (агрегувати) отримані нечіткі множини. Математично це записується в такий спосіб:

$$\tilde{y} = \underset{j=1,m}{\underline{agg}}\left(\int_{\underline{y}}^{\bar{y}} \min\left(\mu_{d_j}(X^*),\ \mu_{d_j}(y)\right) / y \right)$$

де agg - агрегування нечітких множин, яке найбільш часто реалізується операцією знаходження максимуму.

Чітке значення виходу y, відповідне вхідному вектору X^* визначається в результаті дефазифікації нечіткої множини $\tilde{y}$. Найбільш часто застосовується дефазифікація за методом центра ваги:

$$y = \frac{\int_{\underline{y}}^{\bar{y}} y \cdot \mu_{\tilde{y}}(y)dy}{\int_{\underline{y}}^{\bar{y}} \mu_{\tilde{y}}(y)dy}$$

Рисунок 2.4 – Формула методу дефазифікації за методом центра ваги

2.2 Основні принципи дерев прийняття рішень та концепція алгоритму побудови

Розглянемо поняття дерева прийняття рішень та алгоритми їх побудови. Дерева прийняття рішень – це прості прогностичні моделі, які співставляють вхідні атрибути с цільовим значенням з використанням простих умовних правил. Дерева зазвичай використовують в проблемах, рішення яких повинні бути легко зрозумілими або пояснювані людьми, наприклад в комп'ютерній діагностиці [14]. Мета всього процесу побудови дерева прийняття рішень - створити модель, по якій можна було б класифікувати випадки і вирішувати, які значення може приймати цільова функція, маючи на вході кілька змінних. Дерева прийняття рішень дають прямий інтуїтивний спосіб класифікації нового набору вхідних параметрів тестового екземпляру за допомогою набору простих правил. Із-за цієї характеристики дерева прийняття рішень широко використовуються в ситуаціях, коли інтерпретація отриманих результатів та процесу міркувань має вирішальне значення, наприклад в автоматизованій діагностиці. Дерева прийняття рішень можна просто намалювати вручну на основі будь-яких попередніх знань, які може мати автор. Однак їх реальна сила становиться явною, коли дерева навчаються автоматично, використовуючи деякий алгоритм навчання. Найбільш примітними та класичними прикладами для навчання дерева прийняття рішень є алгоритми ІД3 та С4.5. Обидва вони є прикладами жадібних алгоритмів, що виконують пошук локального оптимуму в надії створити найбільш спільне дерево. Алгоритм ІД3 дозволяє працювати тільки з дискретною цільовою змінною, тому дерева рішень, що побудовані з використанням даного алгоритму є класифікуючими.

Жадібний алгоритм — простий і прямолінійний евристичний алгоритм, який приймає найкраще рішення, виходячи з наявних на кожному етапі даних не зважаючи на можливі наслідки, сподіваючись урешті-решт отримати

оптимальний розв'язок. Легкий в реалізації і часто дуже ефективний за часом виконання.

Процес побудови дерев рішень полягає в послідовному, рекурсивному розбитті навчальної множини на підмножини з застосуванням вирішальних правил в вузлах. Процес розбиття триває до тих пір, поки всі вузли в кінці всіх гілок НЕ будуть оголошені листям. Оголошення вузла листом може статися природним чином (коли він буде містити єдиний об'єкт, або об'єкти тільки одного класу), або після досягнення деякої умови зупинки, що задається користувачем (наприклад, мінімально допустиму кількість прикладів у вузлі або максимальна глибина дерева) [18].

Основні етапи побудови дерев прийняття рішень:

1. Вибір атрибута, за яким буде проводитися розбиття в даному вузлі.

2. Вибір критерію зупинки навчання.

3. Вибір методу відсікання гілок.

4. Оцінка точності побудованого дерева.

При формуванні правила для розбиття в черговому вузлі дерева необхідно вибрати атрибут, за яким це буде зроблено. Загальне правило для цього можна сформулювати наступним чином: обраний атрибут повинен розбити безліч спостережень у вузлі так, щоб результуючі підмножини містили приклади з однаковими мітками класу, або були максимально наближені до цього, тобто кількість об'єктів з інших класів «домішок» в кожному з цих множин було якомога менше. Для цього були обрані різні критерії, найбільш популярними з яких стали теоретико-інформаційний та статистичний.

Такі алгоритми засновані на принципі «бритви Окками», обираючи більш прості та менші дерева в надії, що такі більш дрібні дерева можуть зберегти більшу силу узагальнення. Ця перевага формалізується за допомогою специфіки критеріїв вибору гіпотез, ґрунтуючись на прирості інформації.

Приріст інформації вимірює приріст отриманої інформації при використанні кожного із атрибутів в якості наступного фактора для розгляду під час прийняття рішення. Приріст інформації можна визначити за наступною формулою:

$$Gain(S,A) \equiv Entropy(S) - \sum_{v \in V(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Entropy(S) \equiv -\sum_{i=1}^{c} p_i \log_2(p_i)$$

Рисунок 1.1 – Формула приросту інформації

де S – набір навчальних прикладів; A – атрибут розбиття множини значень; | S | - кількість елементів в S, |Sv| - кількість елементів в Sv; v – всі можливі значення атрибуту.

Незважаючи на те, що дерева рішень покладаються на Бритву Оккама, під час керування навчанням, ні алгоритм ID3, ні C4.5 не гарантують отримання меншого або більш загального дерева. Це відбувається тому, що їх навчання засноване виключно на евристиці, а не на умовах дійсної оптимальності.

3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Проектування та розробка програмного забезпечення алгоритму Мамдані

Модель ієрархії класів розроблених з використанням методології ООП, що відображає ключові сутності використовувані при виконанні нечіткого логічного виводу та побудові дерева прийняття рішень:

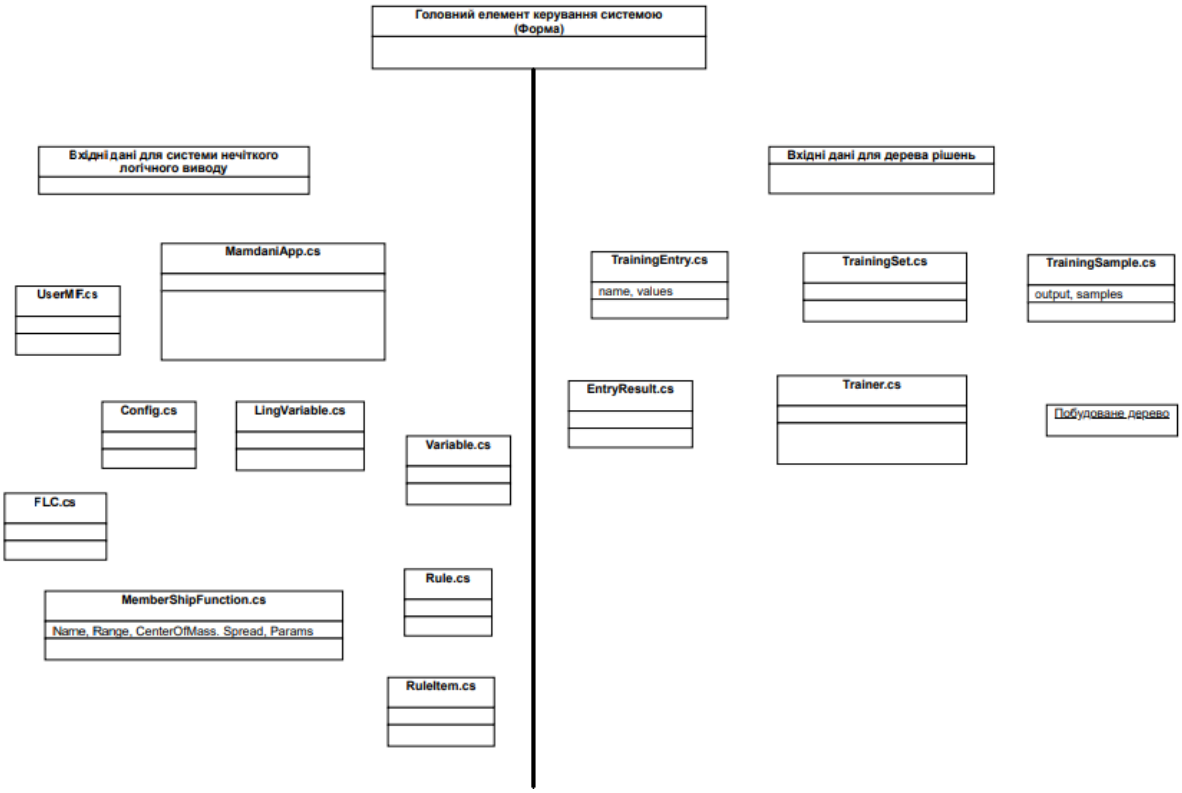


Рисунок 3.1 – Модель розроблених класів

Розглянемо основні особливості розроблених класів:

Головний клас керування системою Form має методи:

– контролю кількості та наявності правил, вхідних та вихідних змінних в системі;

– динамічного реагування та перерахунку результатів відносно доданих або видалених значень вхідних змінних, або правил;

– завантаження власних вхідних даних в системі нечіткого логічного виводу;

– додавання власних вхідних/вихідних змінних до поточної конфігурації та вивід їх значень на екран;

– видалення власних вхідних/вихідних змінних;

– додавання власних функцій приналежності до поточної конфігурації та їх графічна реалізація в системі;

– видалення функції приналежності

LingVariable – клас, що використовується для створення вхідних (InputVariables) та вихідних (OutputVariables) лінгвістичних змінних. В якості параметрів він приймає назву створеної лінгвістичної змінної та визначає її тип. Використовується для задання спектру розподілу значень в яких будуть знаходитись розраховані функції приналежності. Також має вбудований метод пошуку цих функцій приналежності в батьківському класі MemberShipFunction за ім’ям та додавання в них вхідних даних з класу MamdaniApp для їх розрахунку.

Variable – клас, що дозволяє графічно реалізовувати поточні розраховані функції приналежності для лінгвістичних змінних. Цей клас також включає методи для редагування значень вхідних та вихідних лінгвістичних змінних таких як назва або спектр розподілу.

UserMF – в якості вхідних даних для розрахунку користувацької функцій приналежності клас UserMF приймає такі параметри як назва функції приналежності, параметр a - центр функції належності, параметр b - ширина функції належності. Кількість вхідних параметрів для користувацької функції належності була визначена двома значеннями: оцінка роботодавця та випускника. Метод getOutput даного класу - приймає результат, що розраховується за формулою користувацької функції приналежності класа UserMF. Ці дані будуть використовуватися для розрахунку методами імплікації, фазифікації та дефазифікації за центром ваги.

MamdaniApp – визначає список вхідних та вихідних лінгвістичних змінних та список використовуваних правил в базі знань. У даному класі були сформовані усі необхідні вхідні дані у вигляді нечітких множин для розрахунку підсумкового значення ступеня приналежності для трьох лінгвістичних змінних. Відповідно можна визначити до якої якості освіти належить будь-який набір параметрів оцінок від 0 до 100. Таким чином можна виділити значення параметрів які б в більшому ступені задовольняли одне з трьох значень лінгвістичних змінних.

Rule – формує кінцеве значення вирішальних правил із лінгвістичних значень вхідних та вихідних змінних в базі знань за допомогою метода з'єднання вхідних та вихідних правил операндом "And".

RuleItem – використовується для формування списку вхідних або вихідних правил. В якості значень приймає лінгвістичні правила.

Config – визначає за яким методом буде проводитись розрахунок результатів по алгоритму Мамдані: метод імплікації нечітких висловлювань, який реалізується знаходженням мінімуму функції належності та метод дефазифікації центра ваги за яким буде розраховане чітке значення Y на базі значень вихідної нечіткої множини.

FLC – використовує дані всіх класів для проведення процедур імплікації, фазифікації, дефазифікації та повертає кінцеве значення функції приналежності для поточної лінгвістичної змінної.

Також був розроблений клас NewVariable для створення власних вхідних та вихідних лінгвістичних змінних для трикутних функцій належності під час виконання розрахунків. Даний клас буде приймати назву змінної, значення спектру розподілу та тип змінної, та повертати їх в загальний список.

Відповідно до наданих даних та умов реалізації була сформована база знань з 9 вхідних правил та 9 вихідних правил "ЯКЩО-ТО" на основі якої було виконано зведення масиву вхідних змінних універсальної множини $\{d_1, d_2, \ldots, d_m\}$ до нечіткої множини на інтервалі $[\underline{y}, \bar{y}]$:

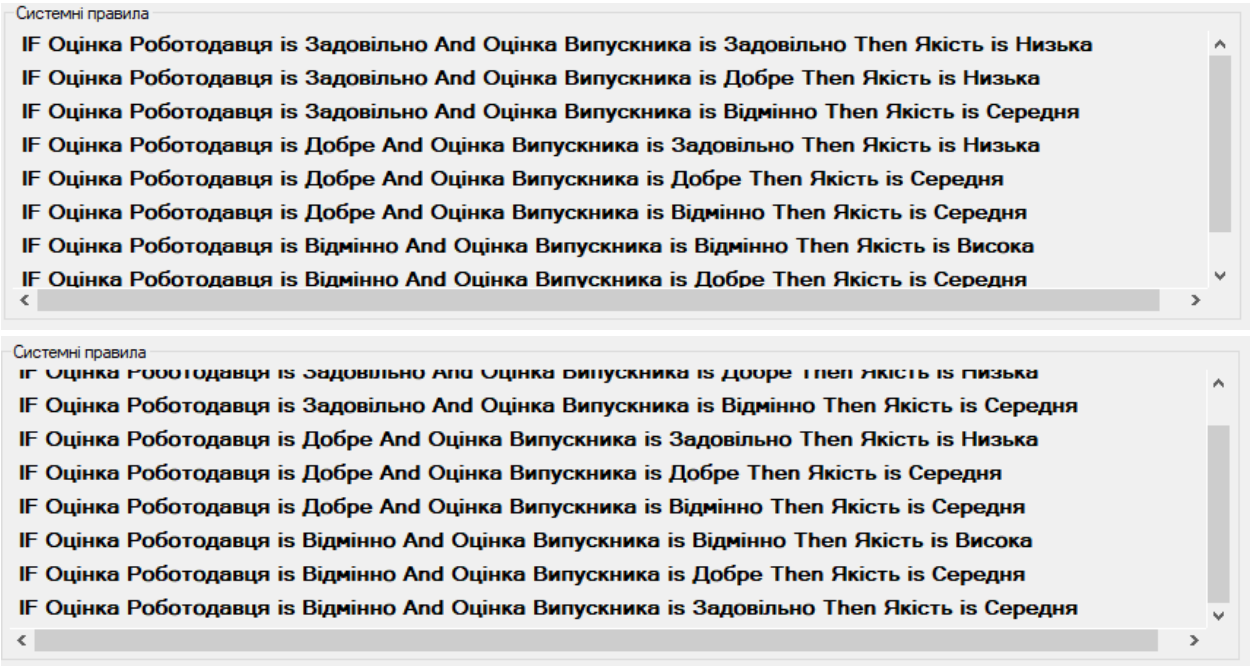


Рисунок 3.2 – Формування бази знань

3.2 Опис користувацького інтерфейсу

Інтерфейс даного проекту являє собою комплексну систему з використанням таких елементів:

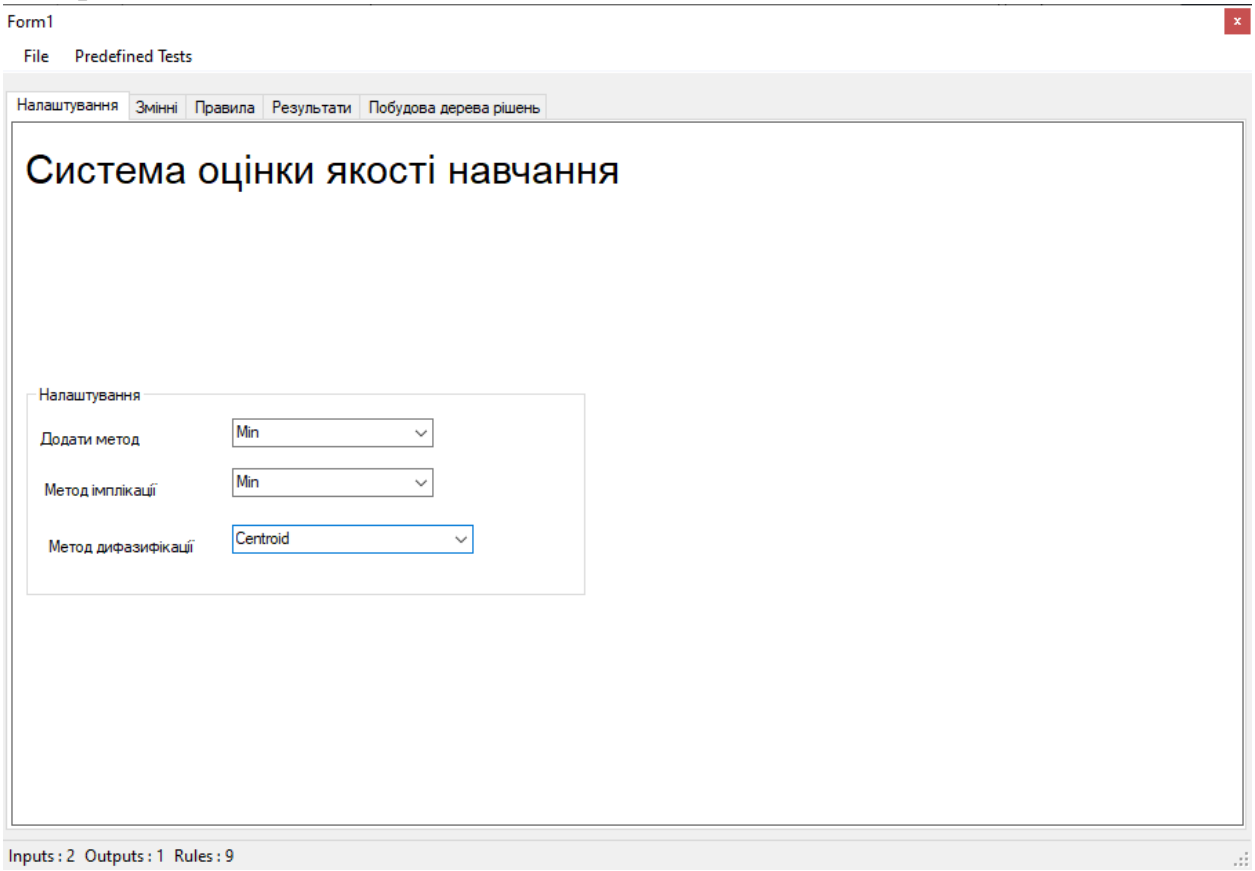


Рисунок 3.1 – Початкове вікно вибору конфігурації

Дана форма передбачає меню для вибору власних конфігурацій для надання даних для успішної взаємодії з алгоритмом нечіткого логічного виводу і вибір методу імплікації вхідних значень та дефазифікації вихідних значень. Після завантаження власної конфігурації відбувається повна очистка вхідних даних, що знаходились в системі раніше та завантаження нових для проведення розрахунків. Основними класами що беруть участь в формуванні даної форми будуть Config, метод myPresetSetting() класу Form для завантаження даних конфігурації за якими будуть проводитись розрахунки та метод reloadStrip() для перевірки кількості вхідних/вихідних змінних та правил в системі.

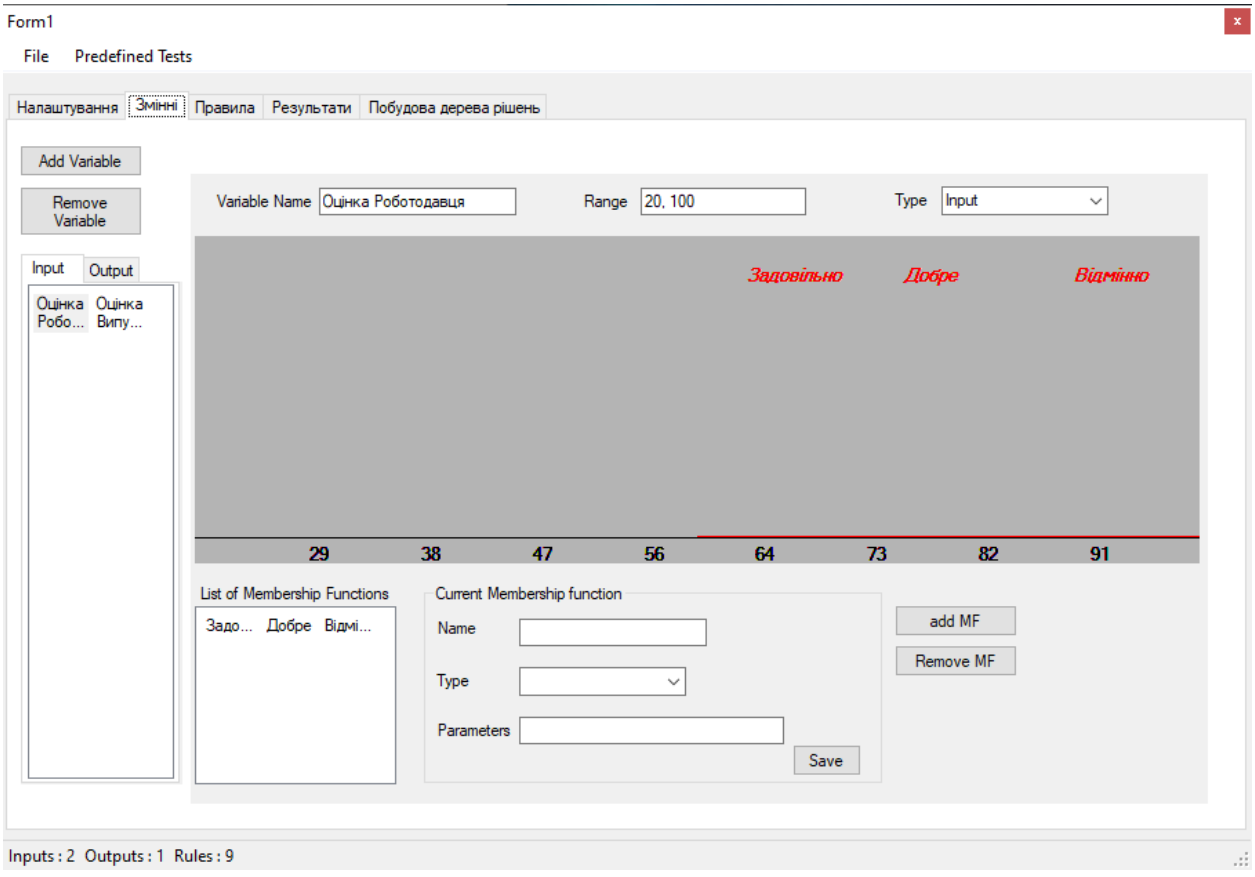


Рисунок 3.2 – Початкове вікно вибору конфігурації

Дана форма відображає на графіку розраховане значення функції приналежності для трьох лінгвістичних змінних у вигляді нечітких множин, що перетинаються. Має методи для додавання або видалення інших змінних

відносно загального розрахунку. Основними класами що беруть участь в формуванні даних цієї форми будуть MamdaniApp, UserMF, LingVariable та Variable.

3.3 Проектування та розробка дерева прийняття рішень ІД3

Побудова дерева рішень буде відбуватися на основі масиву експертних даних тому потрібно для алгоритму побудови дерева розробити методи для розподілу їх на рівні частини та сформувати з них тренувальні данні.

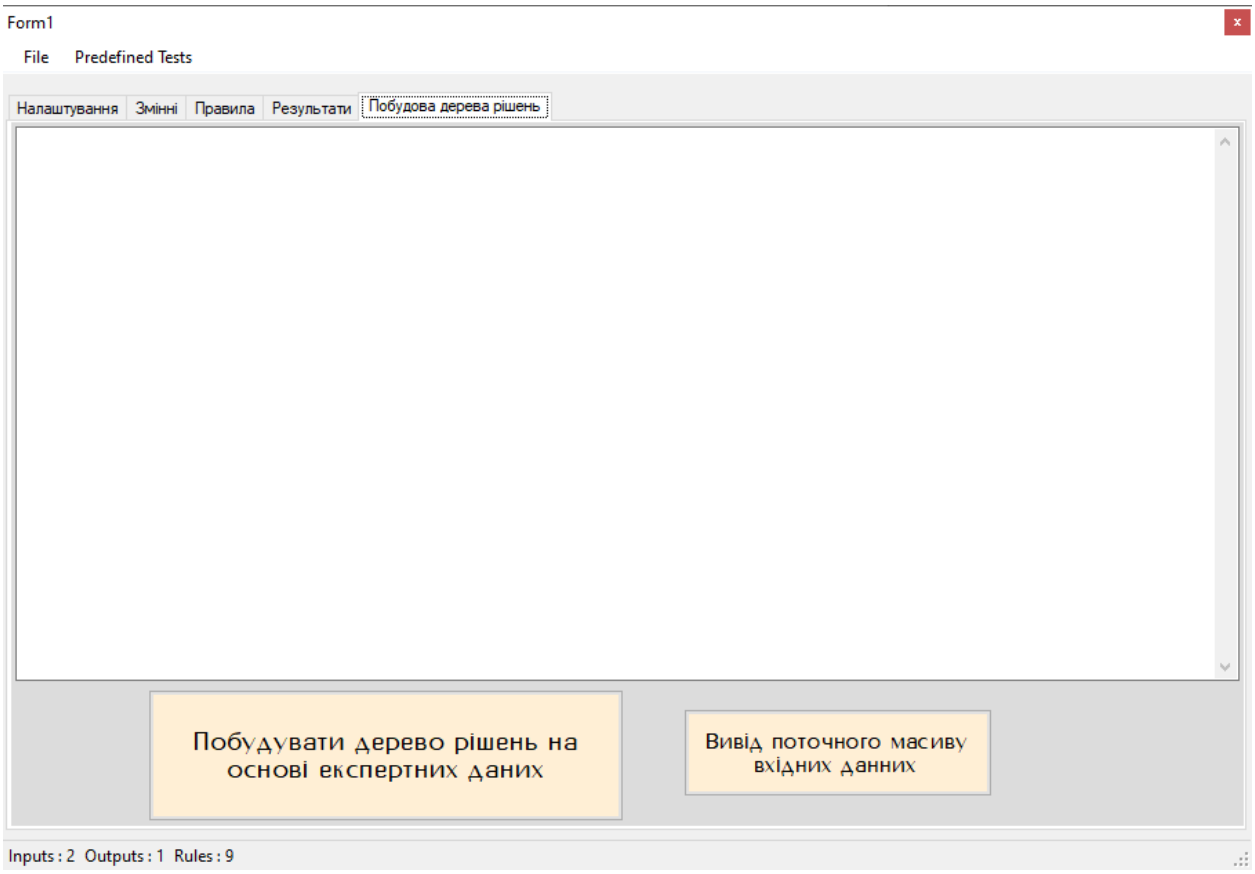


Рисунок 3.1 – Вікно результатів побудованого дерева рішень

Відповідно до алгоритму побудови дерева також потрібен метод, який буде приймати дані для визначення приналежності об’єктів з навчальної множини до того чи іншого класу. Тобто потрібно сформувати клас який б брав значення із навчальної множини та співставляв їх з класифікаційними даними і відповідно б отримував тренувальну множину значень для розрахунку теоретико-інформаційного критерію. Тобто потрібно сформувати

алгоритм пошуку з найкращим значенням атрибуту розбиття, яке забезпечить максимальне зниження ентропії результуючої підмножини відносно батьківської. Найкращий атрибут потрібно визначити як рішення для вузла, для кожного значення атрибуту створити нащадок вузла. Отже зважаючи на вимоги побудова дерева рішень буде відбуватися зверху вниз від кореневого вузла до його листків та рішенням задачі класифікації буде останній вузол дерева рішень, що класифікований як лист. В нашому випадку класифікаційними даними будуть ступені якості освіти та їх значення.

Також в системі була вирішена проблема адаптації числових даних матриці до потрібних лінгвістичних висловлювань для роботи з тренувальною множиною шляхом перетворення поточного числового масиву даних на масив рядкових об’єктів. Дана система також має методи для порядкового виводу значень експертного масиву даних для зручності включення або виключення цих даних в алгоритмі побудови дерева.

Проведемо огляд класів сформованих для побудови дерева рішень:

TrainingEntry.cs – екземпляр цього класу створює набір даних за якими буде проведена класифікація приймає в якості параметрів назву класу та класифікаційні дані

Структура класу:

```
public sealed class TrainingEntry
   {
      public string Name { get; set; }

      public string[] Values { get; set; }

      public TrainingEntry(string name, string[] values)
      {
         this.Name = name;
         this.Values = values;
      }
   }
```

TrainingSample.cs – приймає в якості параметрів множину значень тренувальних даних

Структура класу:

```
public sealed class TrainingSample
   {
      public bool Output { get; private set; }

      public string[] Samples { get; private set; }

      public TrainingSample(bool output, params string[] samples)
      {
         this.Output = output;
         this.Samples = samples;
      }
   }
```

TrainingSet.cs – формує тренувальну множину для її подальшого розподілу найкращим атрибутом

Структура класу:

```
public sealed class TrainingSet
   {
      public string OutputName { get; private set; }

      public TrainingEntry[] Entries { get; private set; }

      public List<TrainingSample> Samples { get; private set; }

      private bool m_canAddSample = true;

      public TrainingSet(TrainingSet other)
      {
         this.OutputName = other.OutputName;
         this.Entries = other.Entries;
         this.Samples = other.Samples;
      }

      public TrainingSet(TrainingSet other, List<TrainingSample> samples)
      {
         this.OutputName = other.OutputName;
         this.Entries = other.Entries;
         this.Samples = samples;
      }

      public TrainingSet(string outputName, params TrainingEntry[] entries)
      {
```

```
            this.Samples = new List<TrainingSample>();
            this.OutputName = outputName;
            this.Entries = entries;
        }

        public int GetDataCount()
        {
            return this.Samples.Count;
        }

        public int GetC1()
        {
            return (from n in this.Samples where n.Output == true select
n).Count();
        }

        public int GetC2()
        {
            return (from n in this.Samples where n.Output == false select
n).Count();
        }

        public double GetP1()
        {
            return (double)this.GetC1() / (double)this.GetDataCount();
        }

        public double GetP2()
        {
            return (double)this.GetC2() / (double)this.GetDataCount();
        }

        public string GetSamplesAsString()
        {

            StringBuilder sb = new StringBuilder();

            for (int i = 0; i < this.Samples.Count; i++)
            {

                TrainingSample current = this.Samples[i];

                sb.AppendLine(string.Join(",", current.Samples));
```

```
            }

            return sb.ToString();
        }

        public void AddSample(TrainingSample sample)
        {
            if (this.m_canAddSample)
            {
                if (!this.Samples.Contains(sample))
                {
                    this.Samples.Add(sample);
                }
                else
                {
                    throw new Exception("Вже містить значення");
                }
            }
            else
            {
                throw new Exception("Навчальні дані заблоковано");
            }
        }

        public void Lock()
        {
            this.m_canAddSample = false;
        }

    }
```

Trainer.cs – проводить навчання тренувальної множини, формує дерево рішень з використанням всіх етапів його побудови та виводить результат в форму.

3.4 Аналіз отриманих результатів

Для перевірки роботи системи Мамдані візьмемо наприклад набір параметрів [65, 85], введемо їх в поле «змінні» та отримаємо результат.

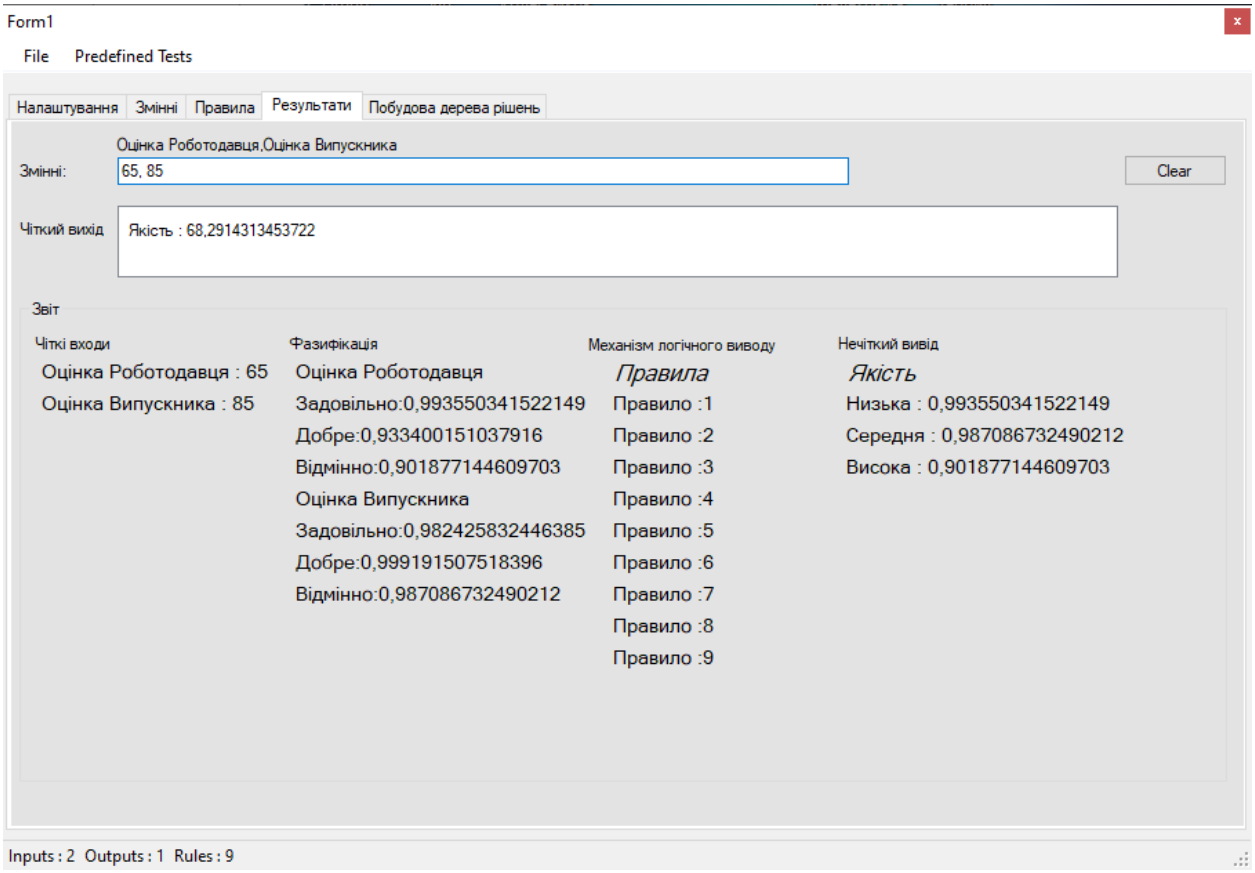
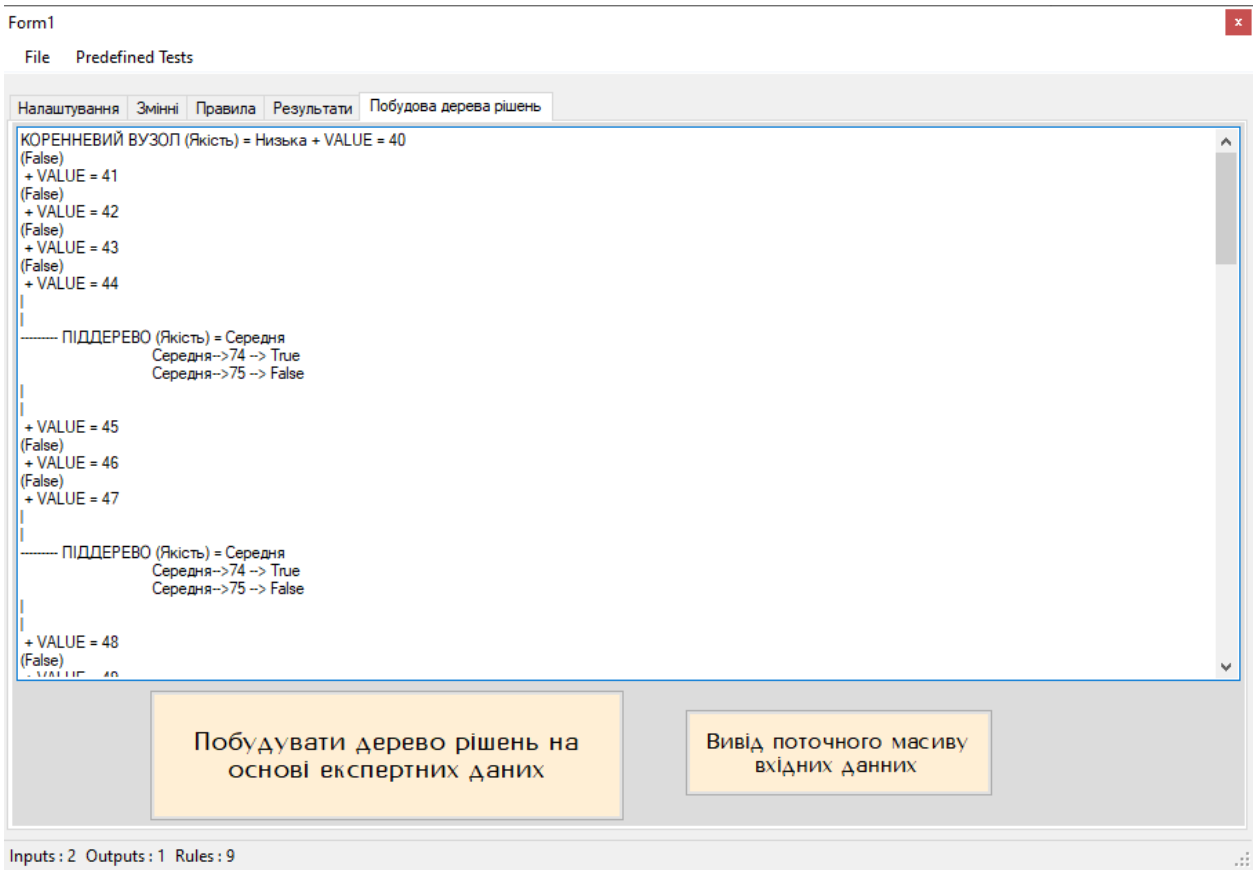


Рисунок 3.1 – Результат роботи програми «Система логічного виводу Мамдані»

За результатами фазифікації значення функції приналежності для «Оцінки роботодавця» найбільш близьке до одиниці в лінгвістичній змінній – «Задовільно», а для «Оцінки випускника» – «Добре». Відповідно ступень якості навчання в нечіткому виводі буде класифікуватись як «Низька».

Далі перевіримо алгоритм побудови дерева прийняття рішень.



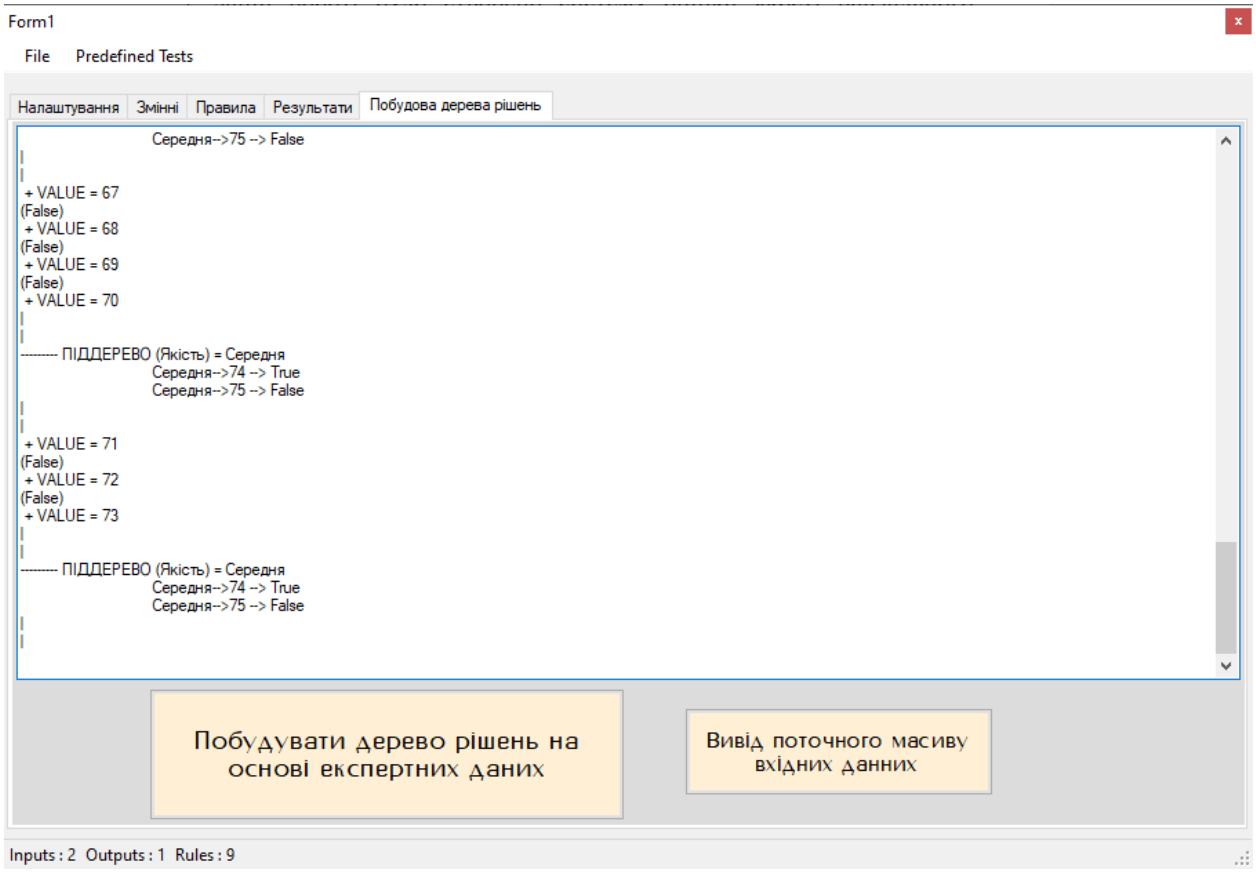


Рисунок 3.2-3.3 – Результат побудови дерева прийняття рішень

Кінцевим вирішальним значенням якості навчання буде останній сформований вузол дерева прийняття рішень, що був оголошений листком з мінімальною ентропією та має значення True, тобто вузол буде містити у собі об’єкти тільки одного класифікатора після розбиття найкращим атрибутом. Відповідно до наданих даних якість навчального контенту була визначена як середня.

ВИСНОВОК

У даній роботі було створено систему оцінки якості навчального контенту з використанням алгоритмів нечіткого логічного виводу та дерева прийняття рішень, що виводить результат у вигляді значень функцій приналежності для лінгвістичних змінних та вирішального листка, який визначає рішення для кожної підмножини об’єктів, що задовільняють всім правилам гілки, яка закінчується даним листком. В якості основного методу реалізації інтелектуальної системи оцінки якості навчального контенту був обраний алгоритм автоматичної класифікації Мамдані у рамках теорії нечітких множин, а для реалізації дерева рішень, алгоритм ітераційного дихотомізатора (ІД3).

СПИСОК ЛІТЕРАТУРИ

1. Балута Т. П. Priorities of information technologies in higher education. Гілея: науковий вісник. – К.: «Видавництво «Гілея», 2019. – Вип. 141 (2). Ч. 2. Філософські науки. – 172 с.

2. Боровик С. Ю. Инструментальные средства проектирования и отладки нечетких логических систем. Пакет fuzzytech. – Самара: 2000. – 77 с.

3. Гудкова В. С., Ячинова С. Н. Модульно-рейтинговая система как средство повышения качества обучения. – Молодой ученый, 2015. – № 8 (88). С. 910-912.

4. Кононюк А.Е. К65 Дискретно-непрерывная математика. Матрицы. К.5.Ч.1. К.4: "Освіта України", 2011. – 612 с.

5. Осіпов В.А., Довбиш А.С. Звіт про науково-дослідну роботу «Розроблення науково-методичних основ та інформаційних засобів проектування здатних самонавчатися адаптивних систем керування технологічними процесами». – Суми: СумДУ, 2009. – 106 с.

6. Кондратенко Г. В. Синтез нечетких регуляторов на основе объектно-ориентированных технологий / Г. В. Кондратенко, Ю. П. Кондратенко, К. В. Мухортова // Автоматика. Автоматизація. Електротехнічні комплекси та системи. - 2005. - № 1. - С. 140-147. - Режим доступу: http://nbuv.gov.ua/UJRN/aaeks_2005_1_26

7. Культин, Н. Б. Основы программирования в Microsoft Visual C# 2010. — СПб.: БХВ6. Петербург, 2011. — 364 с. – Режим доступа: http://znanium.com/bookread.php?book=351294

8. Лазарев Юрий Федорович. Начала программирования в среде matlab: Учебное пособие. К.: НТУУ "КПИ", 2003. 424 с.

9. Леонов Н.Н. Методы нечеткой классификации в социологических исследованиях // Социологический альманах. Вып. 1. – 2010. – С. 147 – 155.

10. Шатовська Т. Б., Каменєва І. В. Комбінований ієрархічний підхід до кластеризації документів, Вісник Вінницького політехнічного інституту, № 1,

с. 47-50. Лис. 2010. Режим доступу до статті: https://visnyk.vntu.edu.ua/index.php/visnyk/article/view/696

11. Штовба С.Д. Идентификация нелинейных зависимостей с помощью нечеткого логического вывода в системе MATLAB // Exponenta Pro: Математика в приложениях. – 2003. – №2. – С. 9–15.

12. Ястребов Ю. Ю. Використання кластерного аналізу для оцінювання ефективності розміщення фінансових ресурсів сільськогосподарських підприємств // Економіка: реалії часу. Науковий журнал. 2018. № 2 (36). С. 65-71. – Режим доступу до журналу: https://economics.opu.ua/files/archive/2018/No2/65.pdf. DOI: 10.5281/zenodo.1308168.

13. «Diving in OOP (Day 1) : Polymorphism and Inheritance (Early Binding/Compile Time Polymorphism)». Режим доступу до статті: https://www.codeproject.com/Articles/771455/Diving-in-OOP-Day-Polymorphism-and-Inheritance-Ear

14. Esmeir, S. & Markovitch, S., 2007. Anytime Learning of Decision Trees. J. Mach. Learn. Res., May, Volume 8, pp. 891-933.

15. https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide Дата звернення: 25.10.2020 р.

16. https://hackernoon.com/why-is-python-used-for-machine-learning-773aaaadd0ea Дата звернення: 01.11.2020 р.

17. https://intuit.ru/studies/courses/6/6/lecture/182 Дата звернення: 07.11.2020 р.

18. https://loginom.ru/blog/decision-tree-p1 Дата звернення: 15.11.2020 р.

19. https://wiki.tntu.edu.ua/Розпізнавання_образів Дата звернення: 21.11.2020 р.

ДОДАТОК А

MamdaniApp.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using FuzzyLogicController;
using FuzzyLogicController.FLC;
using FuzzyLogicController.RuleEngine;
using StringTok;
using FuzzyLogicController.MFs;

namespace FuzzyLogicUI
{
   public static class MamdaniApp
   {
      public static List<LingVariable> InputVariables;
      public static List<LingVariable> OutputVariables;
      public static List<Rule> Rules;
      public static FLC FuzzyControl;
      public static Config Configuration;

      public static void Initalize()
      {
         InputVariables = new List<LingVariable>();
         OutputVariables = new List<LingVariable>();
         Rules = new List<Rule>();
         Configuration = new Config(ImpMethod.Prod, ConnMethod.Min);
         Configuration.DefuzzificationType = DefuzzifcationType.ModifiedHeight;
         FuzzyControl = new FLC(Configuration);
      }

      public static void myPresetSetting()
      {
         LingVariable x1 = new LingVariable("Оцінка Роботодавця", VarType.Input);
         x1.setRange(20, 100);
         x1.addMF(new UserMF("Задовільно", 60, 73));
         x1.addMF(new UserMF("Добре", 69, 89));
         x1.addMF(new UserMF("Відмінно", 85, 100));

         LingVariable x2 = new LingVariable("Оцінка Випускника", VarType.Input);
         x2.setRange(20, 100);
         x2.addMF(new UserMF("Задовільно", 60, 73));
         x2.addMF(new UserMF("Добре", 69, 89));
         x2.addMF(new UserMF("Відмінно", 85, 100));

         MamdaniApp.InputVariables.Add(x1);
         MamdaniApp.InputVariables.Add(x2);

         //output
         LingVariable Quality = new LingVariable("Якість", VarType.Output);
```

```
        Quality.setRange(40, 100);
        Quality.addMF(new UserMF("Низька", 50, 60));
        Quality.addMF(new UserMF("Середня", 70, 80));
        Quality.addMF(new UserMF("Висока", 90, 100));

        MamdaniApp.OutputVariables.Add(Quality);

        //Rules
        List<RuleItem> rule1in = new List<RuleItem>();
        List<RuleItem> rule1out = new List<RuleItem>();
        List<RuleItem> rule2in = new List<RuleItem>();
        List<RuleItem> rule2out = new List<RuleItem>();
        List<RuleItem> rule3in = new List<RuleItem>();
        List<RuleItem> rule3out = new List<RuleItem>();
        List<RuleItem> rule4in = new List<RuleItem>();
        List<RuleItem> rule4out = new List<RuleItem>();
        List<RuleItem> rule5in = new List<RuleItem>();
        List<RuleItem> rule5out = new List<RuleItem>();
        List<RuleItem> rule6in = new List<RuleItem>();
        List<RuleItem> rule6out = new List<RuleItem>();
        List<RuleItem> rule7in = new List<RuleItem>();
        List<RuleItem> rule7out = new List<RuleItem>();
        List<RuleItem> rule8in = new List<RuleItem>();
        List<RuleItem> rule8out = new List<RuleItem>();
        List<RuleItem> rule9in = new List<RuleItem>();
        List<RuleItem> rule9out = new List<RuleItem>();

        rule1in.AddRange(new RuleItem[2] { new RuleItem("Оцінка Роботодавця",
"Задовільно"), new RuleItem("Оцінка Випускника", "Задовільно") });
        rule1out.AddRange(new RuleItem[1] { new RuleItem("Якість", "Низька") });

        rule2in.AddRange(new RuleItem[2] { new RuleItem("Оцінка Роботодавця",
"Задовільно"), new RuleItem("Оцінка Випускника", "Добре") });
        rule2out.AddRange(new RuleItem[1] { new RuleItem("Якість", "Низька") });

        rule3in.AddRange(new RuleItem[2] { new RuleItem("Оцінка Роботодавця",
"Задовільно"), new RuleItem("Оцінка Випускника", "Відмінно") });
        rule3out.AddRange(new RuleItem[1] { new RuleItem("Якість", "Середня") });

        rule4in.AddRange(new RuleItem[2] { new RuleItem("Оцінка Роботодавця", "Добре"),
new RuleItem("Оцінка Випускника", "Задовільно") });
        rule4out.AddRange(new RuleItem[1] { new RuleItem("Якість", "Низька") });

        rule5in.AddRange(new RuleItem[2] { new RuleItem("Оцінка Роботодавця", "Добре"),
new RuleItem("Оцінка Випускника", "Добре") });
        rule5out.AddRange(new RuleItem[1] { new RuleItem("Якість", "Середня") });

        rule6in.AddRange(new RuleItem[2] { new RuleItem("Оцінка Роботодавця", "Добре"),
new RuleItem("Оцінка Випускника", "Відмінно") });
        rule6out.AddRange(new RuleItem[1] { new RuleItem("Якість", "Середня") });
```

```
        rule7in.AddRange(new RuleItem[2] { new RuleItem("Оцінка Роботодавця",
"Відмінно"), new RuleItem("Оцінка Випускника", "Відмінно") });
        rule7out.AddRange(new RuleItem[1] { new RuleItem("Якість", "Висока") });

        rule8in.AddRange(new RuleItem[2] { new RuleItem("Оцінка Роботодавця",
"Відмінно"), new RuleItem("Оцінка Випускника", "Добре") });
        rule8out.AddRange(new RuleItem[1] { new RuleItem("Якість", "Середня") });

        rule9in.AddRange(new RuleItem[2] { new RuleItem("Оцінка Роботодавця",
"Відмінно"), new RuleItem("Оцінка Випускника", "Задовільно") });
        rule9out.AddRange(new RuleItem[1] { new RuleItem("Якість", "Середня") });

        List<Rule> rules = new List<Rule>();
        MamdaniApp.Rules.Add(new Rule(rule1in, rule1out, Connector.And));
        MamdaniApp.Rules.Add(new Rule(rule2in, rule2out, Connector.And));
        MamdaniApp.Rules.Add(new Rule(rule3in, rule3out, Connector.And));
        MamdaniApp.Rules.Add(new Rule(rule4in, rule4out, Connector.And));
        MamdaniApp.Rules.Add(new Rule(rule5in, rule5out, Connector.And));
        MamdaniApp.Rules.Add(new Rule(rule6in, rule6out, Connector.And));
        MamdaniApp.Rules.Add(new Rule(rule7in, rule7out, Connector.And));
        MamdaniApp.Rules.Add(new Rule(rule8in, rule8out, Connector.And));
        MamdaniApp.Rules.Add(new Rule(rule9in, rule9out, Connector.And));
    }

    public static LingVariable getVariablebyName(List<LingVariable> variable, String name)
    {
        return variable.Find(delegate(LingVariable n) { return n.Name == name; });
    }

    public static List<double> tokString(String Text)
    {
        List<double> list = new List<double>();
        StringTokenizer tok = new StringTokenizer(Text);
        tok.NewDelim(new char[] { ',' });

        String token;
        do
        {
            token = tok.NextToken();
            list.Add(Convert.ToDouble(token));

        } while (tok.HasMoreTokens());

        return list;
    }

  }
}
```

UserMF.cs

```
using System;
using System.Collections.Generic;
```

```
using System.Text;

namespace FuzzyLogicController.MFs
{
        public class UserMF : MemberShipFunction
        {
                public UserMF(String Name, double a, double b)
                        : base()
                {
                        this.Name = Name;
                        Params.Add(a); Params.Add(b);
                        range.Add(Params[0]);
                        range.Add(Params[Params.Count - 1]);

                        this.CalculateCenterOfMass();
                        this.calculateSpread();
                }

                public override double getOutput(double Xvalue)
                {
                        double step = 2.5;
                        return 1 / (1 + Math.Pow(Math.Abs((Xvalue - Params[1]) / Params[0]),
step));
                }
        }
}
```

FLS.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using FuzzyLogicController.MFs;
using FuzzyLogicController.RuleEngine;

namespace FuzzyLogicController.FLC
{
   public class FLC
   {

      Config _configuration;

      public FLC(Config conf)
      {
         _configuration = conf;
      }

      private double Centroid(List<FuzzyNumber> nums, LingVariable variable)
      {
         double stepN = 17;
         double start = variable.Range[0];
         double end = variable.Range[1];
```

```
            double step = (end - start) / (stepN - 1);
            double uppersum = 0;
            double lowersum = 0;

            for (int i = 0; i < nums.Count; i++)
            {
                MemberShipFunction mf = variable.getMFbyName(nums[i].MemberShipName);
                for (double j = start; j < end; j = step + j)
                {
                    double value = mf.getOutput(j);
                    if (value > 0)
                    {
                        if (value < nums[i].FuzzyValue)
                        {
                            uppersum = value * j + uppersum;
                            lowersum = value + lowersum;
                        }
                        else
                        {
                            uppersum = nums[i].FuzzyValue * j + uppersum;
                            lowersum = nums[i].FuzzyValue + lowersum;
                        }
                    }
                }
            }
            return (uppersum / lowersum);
        }

        private double getImplication(double value1, double value2)
        {
            if (_configuration.Implication == ImpMethod.Min)
            {
                if (value1 < value2) { return value1; }
                else { return value2; }
            }
            else if (_configuration.Implication == ImpMethod.Prod)
            {
                return value1 * value2;
            }
            return 0;
        }

        private double MaxHeightDeFuzzifcation(List<FuzzyNumber> nums, LingVariable
variable)
        {
            double lowvalue = 0;
            double highvalue = 0;

            for (int i = 0; i < nums.Count; i++)
            {
                MemberShipFunction mf = variable.getMFbyName(nums[i].MemberShipName);
```

```
            highvalue = ((mf.CenterOfMass * getImplication(nums[i].FuzzyValue,mf.getOutput(mf.CenterOfMass))) / (mf.Spread*mf.Spread)) + highvalue;
            lowvalue = (getImplication(nums[i].FuzzyValue, mf.getOutput(mf.CenterOfMass)) / (mf.Spread * mf.Spread)) + lowvalue;
        }
        return (highvalue / lowvalue);
    }

    public List<FuzzyNumber> Fuzzification(double value, LingVariable variable)
    {
        List<FuzzyNumber> set = new List<FuzzyNumber>();
        List<MemberShipFunction> MFs = variable.MFs;
        for (int i = 0; i < variable.MFs.Count; i++)
        {
            double v = MFs[i].getOutput(value);
            if (v > 0)
            {
                set.Add(new FuzzyNumber(MFs[i].Name, v));
            }
        }
        return set;
    }

    public double DeFuzzification(List<FuzzySet> Sets, LingVariable variable)
    {
        double value = 0;
        for (int i = 0; i < Sets.Count; i++)
        {
            if (Sets[i].Variable == variable.Name)
            {
                if (_configuration.DefuzzificationType == DefuzzifcationType.ModifiedHeight)
                {
                    value = MaxHeightDeFuzzifcation(Sets[i].Set, variable);
                }
                else if (_configuration.DefuzzificationType == DefuzzifcationType.Centroid)
                {
                    value = Centroid(Sets[i].Set, variable);
                }
            }
        }
        return value;
    }

  }
}
```

ДОДАТОК Б

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Linq;
using System.IO;

namespace FuzzyLogicUI
{
   public partial class Form1 : Form
   {
      public static string ids;
      public static string[][] results;
      public Form1()
      {
         InitializeComponent();
      }

      private void Form1_Load(object sender, EventArgs e)
      {
         if (MamdaniApp.InputVariables.Count > 0)
         {
            variable1.Current = MamdaniApp.InputVariables[0];
            variable1.pop();
            popVar();
         }
         reloadStrip();
         configurationUI1.loadData();
      }

      private void reloadStrip()
      {
         toolStripStatusLabel1.Text = "Inputs : " + MamdaniApp.InputVariables.Count.ToString();
         toolStripStatusLabel2.Text = "Outputs : " +
MamdaniApp.OutputVariables.Count.ToString();
         toolStripStatusLabel3.Text = "Rules : " + MamdaniApp.Rules.Count.ToString();
      }

      private void popVar()
      {
         for (int i = 0; i < MamdaniApp.InputVariables.Count; i++)
         {
            if (InputView.Items.ContainsKey(MamdaniApp.InputVariables[i].Name) == false)
            {
```

```
                InputView.Items.Add(MamdaniApp.InputVariables[i].Name,
MamdaniApp.InputVariables[i].Name, 1);
            }
        }

        for (int i = 0; i < MamdaniApp.OutputVariables.Count; i++)
        {
            if (OutputView.Items.ContainsKey(MamdaniApp.OutputVariables[i].Name) == false)
            {
                OutputView.Items.Add(MamdaniApp.OutputVariables[i].Name,
MamdaniApp.OutputVariables[i].Name, 1);
            }
        }
    }

    private void AddVariable_Click(object sender, EventArgs e)
    {
        NewVariable var = new NewVariable();
        var.ParentTab = InOutTabs;
        var.ShowDialog();
        popVar();
    }

    private void InputView_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (InputView.SelectedIndices.Count > 0)
        {
            int index = InputView.SelectedIndices[0];
            variable1.ClearVariable();
            variable1.Current = MamdaniApp.InputVariables[index];
            variable1.pop();
        }
    }

    private void OutputView_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (OutputView.SelectedIndices.Count > 0)
        {
            int index = OutputView.SelectedIndices[0];
            variable1.ClearVariable();
            variable1.Current = MamdaniApp.OutputVariables[index];
            variable1.pop();
        }
    }

    private void MainPanel_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (MainPanel.SelectedTab.Text.Equals("Правила"))
        {
            rule1.loadVariables();
            rule1.loadRules();
        }
```

```
    else if (MainPanel.SelectedTab.Text.Equals("Резултати"))
    {
        reultsUI1.loadVariables();
    }

        reloadStrip();

}

private void myPresetMenu_Click(object sender, EventArgs e)
{
    MamdaniApp.Initalize();
    variable1.ClearVariable();
    InputView.Clear();
    OutputView.Clear();
    MamdaniApp.myPresetSetting();
    if (MamdaniApp.InputVariables.Count > 0)
    {
        variable1.Current = MamdaniApp.InputVariables[0];
        variable1.pop();
        popVar();
    }
}

private void closeToolStripMenuItem_Click(object sender, EventArgs e)
{
    MamdaniApp.Initalize();
    variable1.ClearVariable();
    InputView.Clear();
    OutputView.Clear();
    reultsUI1.loadVariables();
}

private void closeToolStripMenuItem1_Click(object sender, EventArgs e)
{
    this.Dispose();
}

private void button1_Click(object sender, EventArgs e)
{
    if (InOutTabs.SelectedTab.Text == "Input")
    {
        MamdaniApp.InputVariables.RemoveAt(InputView.SelectedIndices[0]);
        InputView.Items.RemoveAt(InputView.SelectedIndices[0]);
    }
    else
    {
        MamdaniApp.OutputVariables.RemoveAt(InputView.SelectedIndices[0]);
        OutputView.Items.RemoveAt(InputView.SelectedIndices[0]);
    }
}
```

```
private void button2_Click(object sender, EventArgs e)
{
    TrainingEntry entry1 = new TrainingEntry("Низька", new string[] { "40", "41" });
    TrainingEntry entry2 = new TrainingEntry("Низька", new string[] { "42", "43" });
    TrainingEntry entry3 = new TrainingEntry("Низька", new string[] { "44", "45" });
    TrainingEntry entry4 = new TrainingEntry("Низька", new string[] { "46", "47" });
    TrainingEntry entry5 = new TrainingEntry("Низька", new string[] { "48", "49" });
    TrainingEntry entry6 = new TrainingEntry("Низька", new string[] { "50", "51" });
    TrainingEntry entry7 = new TrainingEntry("Низька", new string[] { "52", "53" });
    TrainingEntry entry8 = new TrainingEntry("Низька", new string[] { "54", "55" });
    TrainingEntry entry9 = new TrainingEntry("Низька", new string[] { "56", "57" });
    TrainingEntry entry10 = new TrainingEntry("Низька", new string[] { "58", "59" });
    TrainingEntry entry11 = new TrainingEntry("Низька", new string[] { "60", "61" });
    TrainingEntry entry12 = new TrainingEntry("Низька", new string[] { "62", "63" });
    TrainingEntry entry13 = new TrainingEntry("Низька", new string[] { "64", "65" });
    TrainingEntry entry14 = new TrainingEntry("Низька", new string[] { "66", "67" });
    TrainingEntry entry15 = new TrainingEntry("Низька", new string[] { "68", "69" });
    TrainingEntry entry16 = new TrainingEntry("Низька", new string[] { "70", "71" });
    TrainingEntry entry17 = new TrainingEntry("Низька", new string[] { "72", "73" });

    TrainingEntry entry18 = new TrainingEntry("Середня", new string[] { "74", "75" });
    TrainingEntry entry19 = new TrainingEntry("Середня", new string[] { "76", "77" });
    TrainingEntry entry20 = new TrainingEntry("Середня", new string[] { "78", "79" });
    TrainingEntry entry21 = new TrainingEntry("Середня", new string[] { "80", "81" });
    TrainingEntry entry22 = new TrainingEntry("Середня", new string[] { "82", "83" });
    TrainingEntry entry23 = new TrainingEntry("Середня", new string[] { "84", "85" });
    TrainingEntry entry24 = new TrainingEntry("Середня", new string[] { "86", "87" });
    TrainingEntry entry25 = new TrainingEntry("Середня", new string[] { "88", "89" });

    TrainingEntry entry26 = new TrainingEntry("Висока", new string[] { "90", "91" });
    TrainingEntry entry27 = new TrainingEntry("Висока", new string[] { "92", "93" });
    TrainingEntry entry28 = new TrainingEntry("Висока", new string[] { "94", "95" });
    TrainingEntry entry29 = new TrainingEntry("Висока", new string[] { "96", "97" });
    TrainingEntry entry30 = new TrainingEntry("Висока", new string[] { "98", "99" });
    TrainingEntry entry31 = new TrainingEntry("Висока", new string[] { "100" });

    TrainingSet set = new TrainingSet("Якість", entry1, entry2, entry3, entry4, entry5, entry6,
    entry7, entry8, entry9, entry10, entry11, entry12, entry13, entry14, entry15,
    entry16, entry17, entry18, entry19, entry20, entry21, entry22, entry23, entry24, entry25,
    entry26, entry27, entry28, entry29, entry30, entry31);
    string[] s = File.ReadAllLines("C:/DiplomMatsDate/expertData.txt");

    int[][] array = new int[s.Length][];

    for (int i = 0; i < array.Length; i++)
    {
        string[] str = s[i].Trim().Split('\t');

        array[i] = new int[str.Length];
        for (int j = 0; j < str.Length; j++)
        {
            array[i][j] = int.Parse(str[j].Trim());
```

```
    }
}

//перетворення числового масиву в масив лінгвістичних даних
int[,] convertArray = To2D(array);
int[] flat = convertArray.Cast<int>().ToArray();
ids = String.Join(" ", flat.Select(p => p.ToString()).ToArray());
string[] lingMass = ids.Split(' ');

int val = 0;
var query = from str in lingMass
            let num = val++
            group str by num / 50 into g
            select g.ToArray();
results = query.ToArray();

/*
for (int i = 0; i < results[0].Length; i++) {
   string[] chunckRow1 += results[i].GetValue;
}
*/
//ArraySegment<String> arraySegment = new ArraySegment<String>(lingMass, 1, 1);

// Формуємо тренувальні дані
set.AddSample(new TrainingSample(true, results[0]));
set.AddSample(new TrainingSample(false, results[1]));
set.AddSample(new TrainingSample(true, results[2]));
set.AddSample(new TrainingSample(false, results[3]));
set.AddSample(new TrainingSample(true, results[4]));
set.AddSample(new TrainingSample(false, results[5]));
set.AddSample(new TrainingSample(true, results[6]));
set.AddSample(new TrainingSample(false, results[7]));
set.AddSample(new TrainingSample(true, results[8]));
set.AddSample(new TrainingSample(false, results[9]));
set.AddSample(new TrainingSample(true, results[10]));
set.AddSample(new TrainingSample(false, results[11]));
set.AddSample(new TrainingSample(true, results[12]));
set.AddSample(new TrainingSample(false, results[13]));
set.AddSample(new TrainingSample(true, results[14]));
set.AddSample(new TrainingSample(false, results[15]));
set.AddSample(new TrainingSample(true, results[16]));
set.AddSample(new TrainingSample(false, results[17]));
set.AddSample(new TrainingSample(true, results[18]));
set.AddSample(new TrainingSample(false, results[19]));
set.AddSample(new TrainingSample(true, results[20]));
set.AddSample(new TrainingSample(false, results[21]));
set.AddSample(new TrainingSample(true, results[22]));
set.AddSample(new TrainingSample(false, results[23]));
set.AddSample(new TrainingSample(true, results[24]));
set.AddSample(new TrainingSample(false, results[25]));
set.AddSample(new TrainingSample(true, results[26]));
set.AddSample(new TrainingSample(false, results[27]));
```

```
    set.AddSample(new TrainingSample(true, results[28]));
    set.AddSample(new TrainingSample(false, results[29]));
    set.AddSample(new TrainingSample(true, results[30]));
    set.AddSample(new TrainingSample(false, results[31]));
    set.AddSample(new TrainingSample(true, results[32]));
    set.AddSample(new TrainingSample(false, results[33]));
    set.AddSample(new TrainingSample(true, results[34]));
    set.AddSample(new TrainingSample(false, results[35]));
    set.AddSample(new TrainingSample(true, results[36]));
    set.AddSample(new TrainingSample(false, results[37]));
    set.AddSample(new TrainingSample(true, results[38]));
    set.AddSample(new TrainingSample(false, results[39]));
    set.Lock();

    Trainer trainer = new Trainer(set);
    trainer.TrainID3(set, textBox1);
}
static T[,] To2D<T>(T[][] arr)
{
    try
    {
        int FirstDim = arr.Length;
        int SecDim = arr.GroupBy(row => row.Length).Single().Key;

        var result = new T[FirstDim, SecDim];
        for (int i = 0; i < FirstDim; ++i)
            for (int j = 0; j < SecDim; ++j)
                result[i, j] = arr[i][j];

        return result;
    }
    catch (InvalidOperationException)
    {
        throw new InvalidOperationException("Errors");
    }
}
private void button3_Click(object sender, EventArgs e)
{
    //вивід поточного масиву вхідних данних на екран
    if (results != null)
    {
        for (int j = 0; j < results[0].Length; j++)
        {
            textBox1.Text += results[0][j] + " ";
            textBox1.Text += results[1][j] + " ";
            textBox1.Text += results[2][j] + " ";
            textBox1.Text += results[3][j] + " ";
            textBox1.Text += results[4][j] + " ";
            textBox1.Text += results[5][j] + " ";
            textBox1.Text += results[6][j] + " ";
            textBox1.Text += results[7][j] + " ";
            textBox1.Text += results[8][j] + " ";
```

```
                textBox1.Text += results[9][j] + " ";
                textBox1.Text += results[10][j] + " ";
                textBox1.Text += results[11][j] + " ";
                textBox1.Text += results[12][j] + " ";
                textBox1.Text += results[13][j] + " ";
                textBox1.Text += results[14][j] + " ";
                textBox1.Text += results[15][j] + " ";
                textBox1.Text += results[16][j] + " ";
                textBox1.Text += results[17][j] + " ";
                textBox1.Text += results[18][j] + " ";
                textBox1.Text += results[19][j] + " ";
                textBox1.Text += results[20][j] + " ";
                textBox1.Text += results[21][j] + " ";
                textBox1.Text += results[22][j] + " ";
                textBox1.Text += results[23][j] + " ";
                textBox1.Text += results[24][j] + " ";
                textBox1.Text += results[25][j] + " ";
                textBox1.Text += results[26][j] + " ";
                textBox1.Text += results[27][j] + " ";
                textBox1.Text += results[28][j] + " ";
                textBox1.Text += results[29][j] + " ";
                textBox1.Text += results[30][j] + " ";
                textBox1.Text += results[31][j] + " ";
                textBox1.Text += results[32][j] + " ";
                textBox1.Text += results[33][j] + " ";
                textBox1.Text += results[34][j] + " ";
                textBox1.Text += results[35][j] + " ";
                textBox1.Text += results[36][j] + " ";
                textBox1.Text += results[37][j] + " ";
                textBox1.Text += results[38][j] + " ";
                textBox1.Text += results[39][j] + " ";
            }
        }
    }
  }
}
```

EntryResult.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace FuzzyLogicUI
{
   public sealed class EntryResult
   {
      public string SampleName { get; set; }

      public double EarningScale { get; set; }

      public EntryResult(string sampleName)
```

```
        {
            this.SampleName = sampleName;
        }
    }
}
```

Trainer.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace FuzzyLogicUI
{
    public sealed class Trainer
    {

        private TrainingSet m_set;

        private List<EntryResult> m_results;

        public Trainer(TrainingSet set)
        {
            this.m_set = set;
            this.m_results = new List<EntryResult>();
        }

        public void TrainID3(TrainingSet set, TextBox textBox1)
        {
            string master = this.CalculateHead(set);
            textBox1.Text = "";
            textBox1.Text += "КОРЕННЕВИЙ ВУЗОЛ (Якість) = " + master;

            StringBuilder sb = new StringBuilder(1000);
            for (int i = 0; i < set.Entries.Length; i++)
            {

                TrainingEntry current = set.Entries[i];

                if (current.Name.Equals(master, StringComparison.InvariantCultureIgnoreCase))
                {
                    for (int j = 0; j < current.Values.Length; j++)
                    {
                        string currentName = current.Values[j];

                        TrainingSet currentExcept = this.GetExceptBranch(set, currentName);
                        string subBranch = CalculateHead(currentExcept);
                        string subName = currentName;

                        sb.AppendLine(" + VALUE = " + subName);

                        if (!subBranch.Equals(master))
```

```
                {
                    sb.AppendLine("|");
                    sb.AppendLine("|");
                    sb.AppendLine("--------- ПІДДЕРЕВО (Якість) = " + subBranch + "\n");

                    PrintSubBranch(currentExcept, sb, subBranch);
                    //this.TrainID3(currentExcept);

                    sb.AppendLine("|");
                    sb.AppendLine("|");
                }
                else
                {
                    sb.AppendLine("(" + CalculateResult(currentExcept, subName) + ")");
                }
                //textBox1.Text += "name = " + currentName + "\t\t sub = " + subName + " -- scale
= " + currentScale;
            }
        }
    }
    textBox1.Text += sb.ToString();
}

public void PrintSubBranch(TrainingSet set, StringBuilder sb, string attibrute)
{
    int index = GetEntryHeadIDFor(set, attibrute);
    for (int i = 0; i < this.m_set.Entries[index].Values.Length; i++)
    {
        string current = this.m_set.Entries[index].Values[i];

        TrainingSet h = GetExceptBranch(set, current);
        sb.AppendLine("\t\t" + attibrute + "-->" + current + " --> " + CalculateResult(h,
current));
    }
}

public int GetEntryHeadIDFor(TrainingSet set, string entry)
{
    for (int i = 0; i < this.m_set.Entries.Length; i++)
    {
        if                                          (this.m_set.Entries[i].Name.Equals(entry,
StringComparison.InvariantCultureIgnoreCase))
        {
            return i;
        }
    }
    return -1;
}

public int GetEntryIDFor(TrainingSet set, string entry)
{
    for (int i = 0; i < this.m_set.Entries.Length; i++)
```

```
        {
            for (int j = 0; j < this.m_set.Entries[i].Values.Length; j++)
            {
                if (this.m_set.Entries[i].Values[j].Equals(entry,
StringComparison.InvariantCultureIgnoreCase))
                {
                    return i;
                }
            }
        }
        return -1;
    }

    public double GetBranchEarningScale(TrainingSet set, int index)
    {

        int totalEntries = set.Entries.Length;

        int totalOutCount = set.Entries[index].Values.Length;

        string[] entryValues = set.Entries[index].Values;

        Dictionary<string, int> outCountsTrue = new Dictionary<string, int>(totalOutCount);
        Dictionary<string, int> outCountsFalse = new Dictionary<string, int>(totalOutCount);

        for (int i = 0; i < set.Samples.Count; i++)
        {
            string currentSample = set.Samples[i].Samples[index];

            bool including = entryValues.Any(s => s.Equals(currentSample,
StringComparison.InvariantCultureIgnoreCase));

            if (including)
            {
                if (set.Samples[i].Output)
                {
                    if (!outCountsTrue.ContainsKey(currentSample))
                    {
                        outCountsTrue.Add(currentSample, 1);
                    }
                    else
                    {
                        outCountsTrue[currentSample]++;
                    }
                }
                else
                {
                    if (!outCountsFalse.ContainsKey(currentSample))
                    {
                        outCountsFalse.Add(currentSample, 1);
                    }
                    else
```

```
                {
                    outCountsFalse[currentSample]++;
                }
            }
        }

    }

    double totalH = 0.0d;

    if (outCountsTrue.Count > 0 && outCountsFalse.Count == 0)
    {
        return 1.0d;
    }
    else if (outCountsTrue.Count == 0 && outCountsFalse.Count > 0)
    {
        return 0.0d;
    }

    foreach (KeyValuePair<string, int> kvp in outCountsTrue)
    {

        int countTrue = 0;
        int countFalse = 0;

        if (outCountsTrue.ContainsKey(kvp.Key))
        {
            countTrue = outCountsTrue[kvp.Key];
        }

        if (outCountsFalse.ContainsKey(kvp.Key))
        {
            countFalse = outCountsFalse[kvp.Key];
        }

        int countTotal = countTrue + countFalse;


        double currentH = 0.0d;

        if (countTrue == 0 && countFalse > 0)
        {
            currentH = this.CalculateH(countFalse / (double)countTotal, countFalse /
(double)countTotal);
        }
        else if (countTrue > 0 && countFalse > 0)
        {
            currentH = this.CalculateH(countTrue / (double)countTotal, countFalse /
(double)countTotal);
        }
        else if (countTrue > 0 && countFalse == 0)
        {
```

```
            currentH = this.CalculateH(countTrue / (double)countTotal, countTrue / (double)countTotal);
        }

        totalH += (double)((double)((double)countTotal / (double)set.GetDataCount()) * currentH);

        //textBox1.Text += "Key = {0}, Value = {1}, TRUE = {2}, FALSE = {3}, H = {4}" + kvp.Key + kvp.Value + countTrue + countFalse + currentH;
      }

      return this.CalculateH(set) - totalH;
    }

    public TrainingSet GetExceptBranch(TrainingSet set, string attibrute)
    {

      List<TrainingSample> samples = new List<TrainingSample>(set.Samples);

      samples.RemoveAll(x => !x.Samples.Contains(attibrute));

      return new TrainingSet(set, samples);
    }


    public double CalculateH(TrainingSet set)
    {
      return this.CalculateH(set.GetP1(), set.GetP2());
    }

    public double CalculateH(double p1, double p2)
    {
      double P1Value = p1 * Math.Log(p1, 2);
      double P2Value = p2 * Math.Log(p2, 2);

      return -(P1Value + P2Value);
    }

    public string CalculateHead(TrainingSet set)
    {

      List<double> earningScales = new List<double>(set.Entries.Length);
      double highest = 0.0d;
      string highestName = string.Empty;

      for (int i = 0; i < set.Entries.Length; i++)
      {
        double scale = this.GetBranchEarningScale(set, i);
        if (i == 0)
        {
          highest = scale;
          highestName = set.Entries[0].Name;
```

```
            }

            if (highest < scale)
            {
               highest = scale;
               highestName = set.Entries[i].Name;
            }

            EntryResult result = new EntryResult(set.Entries[i].Name);
            result.EarningScale = scale;
            this.m_results.Add(result);
            earningScales.Add(scale);
         }

         return highestName;
      }

      public bool CalculateResult(TrainingSet set, string attibrute)
      {

         int index = GetEntryIDFor(set, attibrute);
         int totalEntries = set.Entries.Length;
         int totalOutCount = set.Entries[index].Values.Length;
         string[] entryValues = set.Entries[index].Values;
         string name = set.Entries[index].Name;
         double scale = this.GetBranchEarningScale(set, index);

         Dictionary<string, int> outCountsTrue = new Dictionary<string, int>(totalOutCount);
         Dictionary<string, int> outCountsFalse = new Dictionary<string, int>(totalOutCount);

         for (int i = 0; i < set.Samples.Count; i++)
         {
            string currentSample = set.Samples[i].Samples[index];

            bool including = entryValues.Any(s => s.Equals(currentSample,
StringComparison.InvariantCultureIgnoreCase));

            if (including)
            {
               if (set.Samples[i].Output)
               {

                  if (!outCountsTrue.ContainsKey(currentSample))
                  {
                     outCountsTrue.Add(currentSample, 1);
                  }
                  else
                  {
                     outCountsTrue[currentSample]++;
                  }
               }
               else
```

```
                {
                    if (!outCountsFalse.ContainsKey(currentSample))
                    {
                        outCountsFalse.Add(currentSample, 1);
                    }
                    else
                    {
                        outCountsFalse[currentSample]++;
                    }
                }
            }
        }


        if (!outCountsTrue.ContainsKey(attibrute) && outCountsFalse.ContainsKey(attibrute))
        {
            return false;
        }
        else if (outCountsTrue.ContainsKey(attibrute) && !outCountsFalse.ContainsKey(attibrute))
        {
            return true;
        }

        return false;
    }

    public void PrintEntryResults(Form1 obj)
    {
        for (int i = 0; i < this.m_results.Count; i++)
        {
            obj.textBox1.Text += "NAME = {0}, SCALE = {1}" + this.m_results[i].SampleName + this.m_results[i].EarningScale + "\n";
        }
    }
    public void PrintTrainingSet(TrainingSet set, Form1 obj)
    {
        for (int i = 0; i < set.Samples.Count; i++)
        {
            for (int j = 0; j < set.Samples[i].Samples.Length; j++)
            {
                obj.textBox1.Text += "NAME = {0}" + set.Samples[i].Samples[j] + "\n";
            }
        }
    }

  }
}
```