МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ АГРОТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ
ВІДДІЛЕННЯ «ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП'ЮТЕРНІ НАУКИ»
ЦИКЛОВА КОМІСІЯ «ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП'ЮТЕРНІ НАУКИ»

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи освітнього ступеня «фаховий молодший бакалавр»
за спеціальністю 122 «Комп’ютерні науки»
(освітня програма 122 «Комп’ютерні науки»)
на тему:

«Розробка програмного забезпечення для аналізу та оптимізації вибору в умовах невизначеності за допомогою класичних критеріїв»

Студента IV курсу П-42 групи
Матвійчука Дмитра Сергійовича
(прізвище, ім’я та по-батькові)

Керівник Можаровський Сергій Володимирович
(прізвище, ім’я та по-батькові)

Рецензент
(прізвище, ім’я та по-батькові)

Національна шкала

Кількість балів: Оцінка: ECTS

Члени комісії Ісаєв А.М.
(підпис) (прізвище та ініціали)

Габрійчук Н.І.
(підпис) (прізвище та ініціали)

Устименко Л.М.
(підпис) (прізвище та ініціали)

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ АГРОТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ
ВІДДІЛЕННЯ «ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП'ЮТЕРНІ НАУКИ»
ЦИКЛОВА КОМІСІЯ «ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП'ЮТЕРНІ НАУКИ»

«ЗАТВЕРДЖУЮ»
Голова циклової комісії «Інженерна інфраструктура та комп’ютерні науки»
_____Д.М.Палій
«06» листопада 2024 р.

ЗАВДАННЯ

на дипломну роботу

Здобувач фахової передвищої освіти: **МАТВІЙЧУК Дмитро Сергійович**

Керівник роботи: **МОЖАРОВСЬКИЙ Сергій Володимирович**

Тема роботи: **«Розробка програмного забезпечення для аналізу та оптимізації вибору в умовах невизначеності за допомогою класичних критеріїв»**,

затверджена наказом закладу вищої освіти від «03» грудня 2024 р., №496н.

Вихідні дані для роботи: **об'єктом дослідження є програмна реалізація процесу прийняття рішень в умовах невизначеності; предметом дослідження є методи аналізу рішень на основі класичних критеріїв: Гурвіца, мінімаксний критерій Вальда, Севіджа, Байєра-Лапласа.**

Консультанти з дипломної роботи із зазначенням розділів, що їх стосується:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	МОЖАРОВСЬКИЙ С. В.	12.12.2024	02.04.2025
2	МОЖАРОВСЬКИЙ С. В.	02.04.2025	14.05.2025
3	МОЖАРОВСЬКИЙ С. В.	14.05.2025	28.05.2025

Календарний план

№ з/п	Етап роботи	Термін виконання	Примітка
1	Формулювання мети та цілей роботи, визначення об’єкта та предмета дослідження, визначення завдань розробки	12.12.2024	Виконано
2	Планування розробки	29.01.2025	Виконано
3	Формулювання технічного завдання та огляд літератури	19.02.2025	Виконано
4	Проєктування структури	02.04.2025	Виконано
5	Написання програмного коду та відлагождення	23.04.2025	Виконано
6	Тестування та розгортання системи	14.05.2025	Виконано
7	Систематизація результатів розробки, написання та оформлення ПЗ	28.05.2025	Виконано
8	Попередній захист роботи	13.06.2025	Виконано

Здобувач фахової передвищої освіти _____ Дмитро МАТВІЙЧУК

Керівник _____ Сергій МОЖАРОВСЬКИЙ

РЕФЕРАТ

Записка: 71 стор., 33 рис., 6 табл., 1 додаток, 20 джерел.

Ключові слова: ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ПРОГРАМА, МАТЕМАТИЧНІ МЕТОДИ ДОСЛІДЖЕННЯ ОПЕРАЦІЙ, КЛАСИЧНІ КРИТЕРІЇ, ПРОЦЕС ПРИЙНЯТТЯ РІШЕНЬ В УМОВАХ НЕВИЗНАЧЕНОСТІ.

Об'єкт дослідження: класичні критерії прийняття рішень в умовах невизначеності.

Мета роботи: розробка програмного забезпечення для аналізу та оптимізації вибору в умовах невизначеності за допомогою класичних критеріїв.

Методи дослідження: критерії Вальда (мінімакс), Байєса-Лапласа, Севіджа та Гурвіца для обґрунтування вибору за умов невизначеності.

Результати: розроблено програмне забезпечення для аналізу та оптимізації вибору в умовах невизначеності за допомогою класичних критеріїв Байєса-Лапласа, Севіджа, Гурвіца та Вальда.

ABSTRACT

The work consists of an introduction, three main sections, and a conclusion. It is presented on 71 pages of printed text and includes 33 figures, 6 tables, 1 appendix, and 20 references.

The work is focused on the development of software for analyzing and optimizing decision-making under uncertainty using classical decision-making criteria, such as Wald’s (minimax), Bayes-Laplace, Savage, and Hurwicz. The program implements these methods to support justified choice in uncertain conditions and may be used for educational, scientific, or practical decision analysis.

KEY WORDS: SOFTWARE, PROGRAM, MATHEMATICAL METHODS OF OPERATIONS RESEARCH, CLASSICAL CRITERIA, DECISION MAKING PROCESS UNDER UNCERTAINTY.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

IDE – інтегроване середовище розробки

ПЗ – програмне забезпечення

S – критерій Севіджа

HW – критерій Гурвіца

BL – критерій Байєра-Лапласа

MM – мінімаксний критерій (Вальда)

ВСТУП

Актуальність теми. Складність і динамічність сучасного світу невпинно зростають, а отже, з'являється неабияка невизначеність у прийнятті рішень попри всі сфери людської діяльності: від підприємництва та фінансово-кредитної сфери до державного врядування, постачання, торгівлі і зрештою, повсякденного життя. Саме через невизначеність з'являються ситуації, де результати того чи іншого вибору здебільшого не передбачувані, адже залежать від ряду зовнішніх чинників, ймовірність появи яких часто невідома або оцінюється лише наближено. Як наслідок, ухвалення дієвих та обґрунтованих рішень набуває характеру надзвичайно важливої задачі, від якої напряму залежить успішність функціонування систем та досяжність встановлених цілей.

У рамках протидії зазначеним труднощам дедалі більше відчувається брак автоматизованих систем, спроможних підтримувати ухвалення рішень в контексті невизначеності. Використання класичних критеріїв, як-от критерій Вальда (мінімакс), Байєса-Лапласа, Севіджа та Гурвіца, є усталеним та вдалим інструментом для спрощення формалізації проблеми вибору. Втім, їх практичне запровадження потребує програмних засобів, котрі б спрощували процедуру вводу початкових даних, автоматизовували б складні математичні обрахунки й наочно відображали отримані результати.

Мета: створення програмного комплексу, сфокусованого навколо прийняття рішень за умов невизначеності завдяки використанню класичних критеріїв аналізу, які надаватимуть користувачам дієвий інструментарій для обґрунтованого вибору та оптимізації альтернативних стратегій.

Об’єкт дослідження: класичні критерії прийняття рішень в контексті невизначеності та їхня програмна реалізація.

Предмет дослідження: застосування критеріїв мінімаксу (критерій Вальда), Байєса-Лапласа, Севіджа та Гурвіца для обґрунтування вибору за умов невизначеності, зіставлення їх ефективності при різній кількості станів та альтернатив, ймовірностях, критерії оптимізму/песимізму, а також порівняння їх доцільності.

Методи дослідження:

– опрацювання наукових праць, публікацій, статей, посібників та інших авторитетних джерел з теорії прийняття рішень та класичних критеріїв з аналізу в контексті невизначеності, виявлення їхніх сильних й слабких аспектів в різних областях їхнього застосування;

– проєктування та розробка програмного забезпечення для аналізу та оптимізації вибору в умовах невизначеності за допомогою класичних критеріїв, вибір мови програмування ($C\#$) та інструментів розробки (.NET Framework), написання програмного коду, ретельне тестування та налагодження для забезпечення його коректності та стабільності;

– проведення тестових експериментів з різною кількістю станів та альтернатив, матриці виграшів, ймовірності станів, коефіцієнтом Гурвіца (оптимізму/песимізму);

– написання методичних рекомендацій для роботи з програмою, графічна інтерпретація розрахунків (гістограма), яка підвищує наочність та зручність інтепретації оптимальних альтернатив для кожного з критеріїв.

Структура дипломної роботи: постановка задачі, аналіз подібних комерційних продуктів, обґрунтування вибору комп’ютерних технологій, проєктування головного вікна, реалізацію критеріїв мінімаксу (критерій Вальда), Байєса-Лапласа, Севіджа та Гурвіца, тестування та опис його подальшого практичного використання.

РОЗДІЛ 1. ОГЛЯД ТА АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

1.1. Постановка задачі розробки програмного продукту

Ключовим етапом роботи є розробка програмного забезпечення для аналізу та оптимізації вибору в умовах невизначеності з використанням класичних критеріїв Вальда (мінімаксного), Севіджа, Гурвіца та Байєса-Лапласа. Отриманий програмний продукт має стати надійним інструментом для осіб, які приймають рішення, дозволяючи їм орієнтуватися в складних сценаріях, результати яких не є до кінця спрогнозованими.

Перед написанням програми потрібно зважити деякі аспекти.

По-перше, вибір класичних критеріїв: необхідно обґрунтувати вибір критеріїв Вальда (мінімальний), Севіджа, Гурвіца та Байєса-Лапласа, які будуть реалізовані в програмному продукті. Важливо проаналізувати обмеження, що накладаються кожним критерієм, і визначити, з якими типами даних вони найкраще працюють. Приміром, критерій Вальда є песимістичним, обираючи альтернативу, яка максимізує мінімальний виграш, тоді як критерій Гурвіца дозволяє коригувати рівень оптимізму/песимізму. Критерій Байєса-Лапласа потребує ймовірностей для кожного стану, тому він підходить для сценаріїв, де значення ймовірностей є відомими або можуть бути обрахованими.

По-друге, обробка та аналіз введених даних, матриця виграшів: можливість вводити числові значення для альтернатив і станів, обробка помилок та некоректного ведення чисел, перевірка діапазону для таблиці ймовірностей (сумарно 1), якщо всі значення в матриці однакові — чи потрібне попередження.

По-третє, взаємодія з користувачем. Динамічна зміна розмірності (кількість станів/альтернатив) без перезавантаження, підтримка копіювання/вставки з Excel, реалізація підказок для користувача (наприклад, «сума ймовірностей має дорівнювати 1»).

По-четверте, візуалізація результатів, тестування та документація. Зрозумілий гайд із прикладами (наприклад, як інтерпретувати критерій Севіджа), коментарі в коді для математичних формул, текстові звіти з виділенням

ключових даних, можливість друку звітів. Окрім текстового представлення розрахунків, у програмі слід передбачити графічне оформлення отриманих результатів (у вигляді гістограми), яка дозволить оперативно зіставляти значення, одержані за різними критеріями, і наочно оцінювати кращі розрахунки.

Програмний продукт повинен якнайкраще відповідати потребам користувачів у прийнятті рішень в умовах невизначеності, пропонуючи не лише розрахунки, але й інструменти для розуміння та представлення цих розрахунків. А це означає, що система має бути не просто калькулятором, а інтегрованою платформою від введення даних до аналізу та представлення кінцевих результатів.

1.2. Аналіз та порівняння подібних додатків

Попри те, що розроблюваний програмний продукт зосереджений на реалізації класичних критеріїв Вальда, Севіджа, Гурвіца та Байєса-Лапласа для оптимізації вибору, слід наголосити, що це частина ширшої екосистеми подібних програм, у кожній з яких є свої особливості та переваги.

Комерційний пакет бізнес-аналітики *SAS Enterprise Miner* вирізняється як сильний лідер у сегменті аналітики для управління активами та пасивами, а також є визнаним лідером у категорії інтегрованих рішень для трансфертного ціноутворення, управління ризиком ліквідності, оптимізації капіталу та балансу, хеджування та управління ризиками, а також фінансового планування та бюджетування (див.рис.1.1) [1, 18]. Він, як правило, включає не тільки класичні критерії, а й більш складні алгоритми машинного навчання, імітаційне моделювання (Монте-Карло), аналіз чутливості, а також інтеграцію з великими базами даних. Водночас, частина рішень використовує більш математичний та симуляційний підхід. До основних недоліків варто віднести дороговизну та складність освоєння, що іноді стає проблемою для пересічного користувача чи освітньої організації [2, 19].

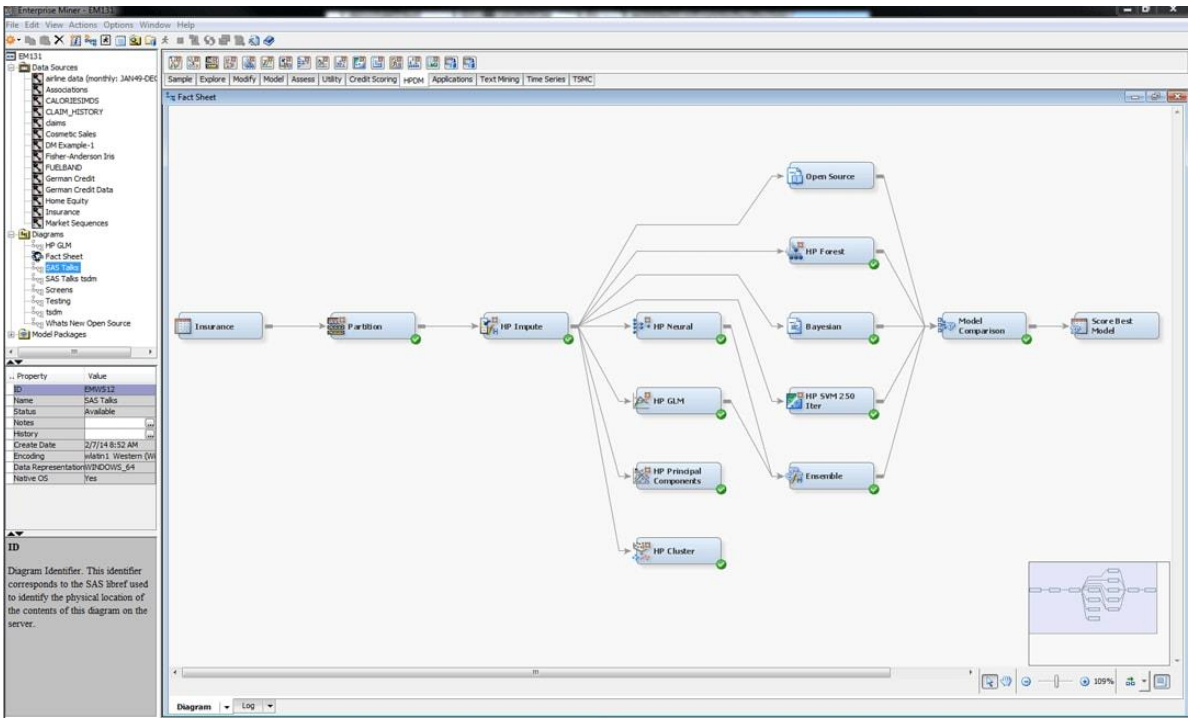


Рисунок 1.1 – Платформа SAS Enterprise Miner

Комерційний пакет бізнес-аналітики SPSS Decision Trees від IBM дає змогу визначати окремі групи, знаходити залежності між ними та передбачати подальші тенденції розвитку подій [3, 17].

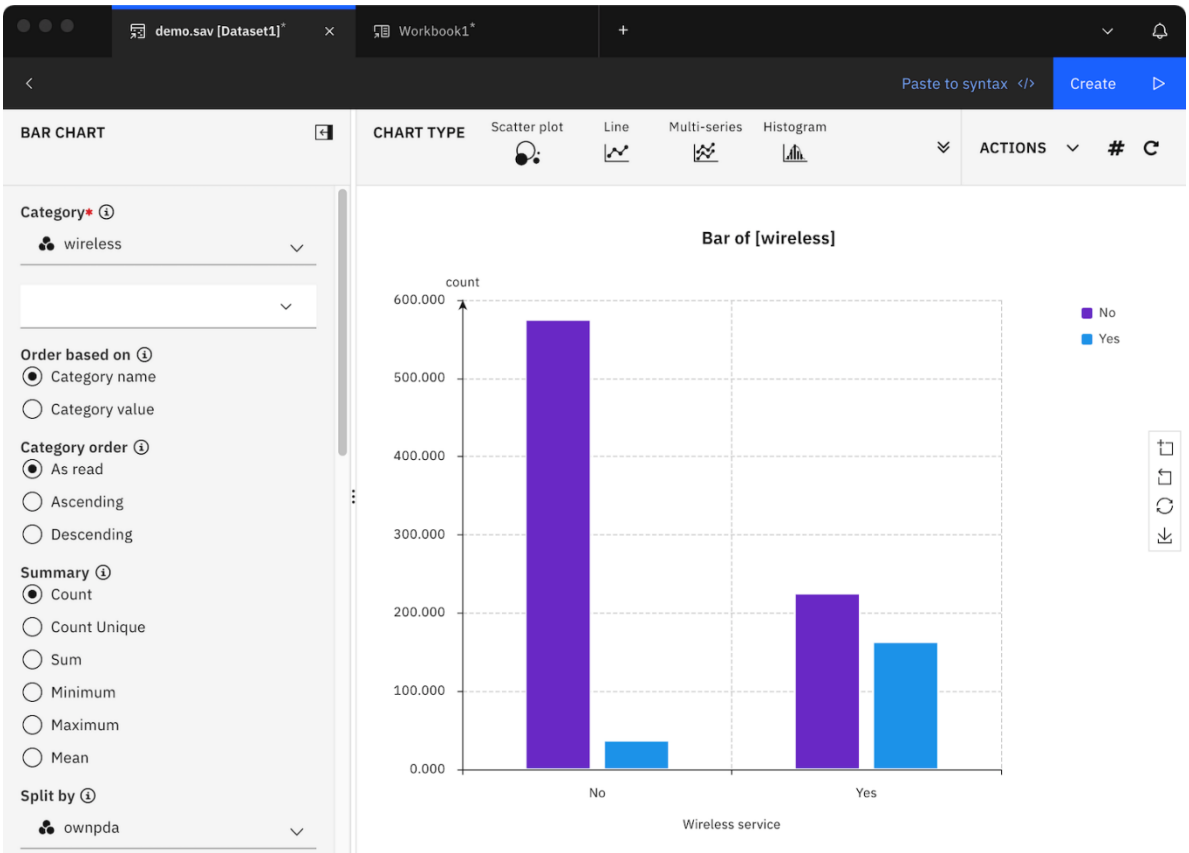


Рисунок 1.2 – IBM SPSS Statistics (вкладка Beta)

SPSS Statistics пропонує універсальний набір протестованих інструментів для управління даними, статистичних і графічних процедур у простому у використанні пакеті. Розширені API-інтерфейси Python і R дають змогу користуватися ресурсами з відкритим вихідним кодом і безперешкодно вбудовувати їх у SPSS [4, 16].

При складності у розумінні програми, можна скористатись віртуальними рекомендаціями, де покроково показано сам процес аналізу і побудови рішення (див.рис.1.3).

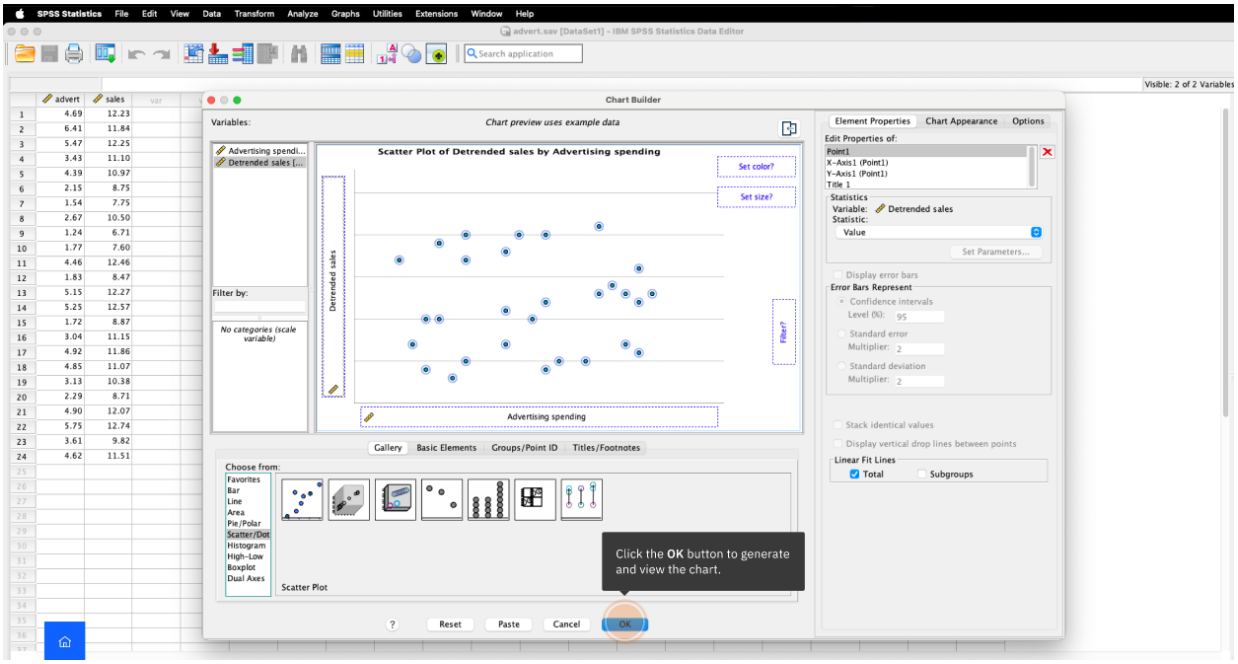


Рисунок 1.3 – Довідка з використання

Є можливість скористатись демо-версією програми, але щоб одержати повноцінний продукт, потрібно заплатити 87.67 євро за авторизованого користувача.

До безкоштовних та найпростіших засобів для аналізу та оптимізації вибору в умовах невизначеності за допомогою класичних критеріїв Вальда, Севіджа, Гурвіца, Байєра-Лапласа можна віднести Microsoft Excel.

Оскільки, в переважній більшості користувачі, які зіштовхуються з необхідністю аналізу даних та прийнятті рішень, володіють елементарними уміннями та навичками роботи з таблицями Excel. На рис.1.4 показана програмна

реалізація прийняття рішень в умовах ризику за допомогою класичних критеріїв Вальда, Байєса-Лапласа, Севіджа та Гурвіца.

Вибрати найкраще рішення, користуючись мінімаксним (MM) і BL-критеріями

	α_1	α_2	α_3	α_4	α_5	$f_{ir} = \min_j f_{ij}$	Z_{mm}	$\sum f_{ij} \cdot g_j$	$Z_{BL} = \max_i f_i$
X_1	29	23	34	56	66	23	25	38.98	48.72
X_2	56	25	62	30	56	25		48.72	
X_3	37	53	31	24	38	24		33.98	

Вибрати найкраще рішення, користуючись S-критерієм

	α_1	α_2	α_3	α_4	α_5	$f_{ir} = \max_i \Delta_{ij}$	$Z_s = \min_i f_{ir}$
X_1	27	30	28	0	0	30	28
X_2	0	28	0	26	10	28	
X_3	19	0	31	32	28	32	
max	56	53	62	56	66		

Вибрати найкраще рішення, користуючись HW-критерієм (Гурвіца)

$$f_{ir} = \left(C \cdot \min_j a_{ij} + (1 - C) \max_j a_{ij}\right)$$

$$Z_{HW} = \max_i \left(C \min_j a_{ij} + (1 - C) \max_j a_{ij}\right)$$

	α_1	α_2	α_3	α_4	α_5	f_{ir}	Z_{HW}
X_1	29	23	34	56	66	32	32
X_2	56	25	62	30	56	32	
X_3	37	53	31	24	38	30	

Рисунок 1.4 – Приклад роботи з Microsoft Excel

До послуг розробників та дослідників безліч різноманітних бібліотек на таких мовах програмування, як Python (SciPy, NumPy для виконання математичних операцій, pandas для маніпулювання даними, matplotlib/seaborn для створення візуалізацій), R (з низкою пакетів для ухвалення рішень) або C# (де для написання дипломної роботи використовуватимуться System.Windows.Forms.DataVisualization.Charting, ClosedXML.Excel, iTextSharp.text.pdf для візуалізації, експорту та друку). Зазначені інструментарії є базовими для створення власних рішень, надаючи можливість максимально гнучко адаптувати функціонал до конкретних потреб користувача [5, 20].

1.3. Вибір та обґрунтування технологій розробки програмного продукту

Для написання програмного коду дипломної роботи «Розробка програмного забезпечення для аналізу та оптимізації вибору в умовах невизначеності з використанням класичних критеріїв» мною обрано Visual Studio Community як інтегроване середовище розробки, C# як мову програмування та .NET Framework як базову платформу. Такий вибір був обумовлений їхньою винятковою продуктивністю, високою адаптивністю та різноманітною функціональністю, котра дозволяє створювати ефективні та зручні для користувача десктопні додатки [6]. Visual Studio, як провідне середовище розробки, надає потужні інструменти для проєктування інтерфейсу, відладки коду та керування проєктами, що суттєво прискорює процес написання додатків.

Завдяки такому поєднанню технологій будуть ефективно реалізовані алгоритми обчислення за критеріями Вальда (мінімаксний), Гурвіца, Севіджа та Байєса-Лапласа. Додатково це забезпечить розробку дружнього та інтуїтивно зрозумілого графічного інтерфейсу користувача, що має ключове значення для програми з підтримки прийняття рішень. Не менш важливою перевагою обраних технологій є здатність легко вбудовувати компоненти для взаємодії з популярними форматами даних, такими як Microsoft Excel (завдяки бібліотеці ClosedXML) та PDF (за допомогою iTextSharp), що дасть змогу здійснювати зручний імпорт та експорт даних, а разом з тим генерувати детальні звіти [7].

Висновки до розділу 1

У першому розділі проведено ретельний аналіз наявних методів та програмних засобів підтримки прийняття рішень в умовах невизначеності. Розглянуто основні комерційні та вільно поширювані програмні продукти, такі як SAS Enterprise Miner, IBM SPSS та Microsoft Excel, що дало змогу визначити їх переваги, недоліки та обмеження з точки зору гнучкості, доступності та адаптації до конкретних потреб користувача.

На основі порівняльного аналізу обґрунтовано доцільність створення власного програмного продукту з використанням класичних критеріїв - Вальда, Севіджа, Гурвіца та Байєса-Лапласа, які є фундаментальними інструментами аналізу альтернатив в умовах невизначеності. Окреслено ключові функціональні вимоги до системи: зручність введення даних, перевірка коректності ймовірностей, візуалізація результатів та їх експорт у популярні формати.

Крім того, обґрунтовано вибір мови програмування C# та середовища розробки Visual Studio, завдяки яким забезпечується належна продуктивність, доступ до широкого спектру бібліотек, зручність побудови графічного інтерфейсу та підтримка інтеграції з форматами Excel та PDF.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональні можливості та використання програмного продукту

Критерій прийняття рішень є функцією, що відображає уподобання особи, яка приймає рішення, та визначає алгоритм для вибору найкращого варіанту. У ситуаціях з неповними даними, кожне рішення приймається з урахуванням кількісних параметрів ситуації [10, 11]. Критерії можуть застосовуватися послідовно, а після розрахунку їх значень серед кількох варіантів доводиться неформально вибрати остаточне рішення. По-перше, це сприяє глибокому розумінню внутрішніх зв'язків проблеми прийняття рішень, а по-друге, зменшує вплив особистих суб'єктивних чинників.

Критерієм крайнього песимізму виступає критерій Вальда (інші назви: мінімаксний, максимінний). Статистик вважає, що «природа» завдаватиме максимального удару, тому він має зіграти так, щоб забезпечити собі гарантований результат та обрати таку чисту стратегію, при якій найменший виграш буде максимальним з усіх можливих (серед мінімальних значень вибирається максимум).

В таблиці 2.1 наведено приклад знаходження критерію Вальда.

Таблиця 2.1

Критерій Вальда (мінімакс)

	α_1	α_2	α_3	α_4	α_5	$f_{ir} = \min_j f_{ij}$	Z_{mm}
X_1	104	103	125	125	126	103	111
X_2	171	108	104	128	133	104	
X_3	125	114	113	111	141	111	

Як видно з таблиці, найкраще значення одержали для *третої альтернативи*.

Критерій Байєра-Лапласа (інші назви: критерій математичного сподівання, критерій максимуму середнього виграшу) сприяється на припущення, що вже

відомі ймовірності настання можливих станів зовнішнього середовища. Обовязкова вимога, яка накладається, сума ймовірностей дорівнює 1. Є випадки розвязання, де береться середнє значення ймовірності (1 поділити на кількість альтернатив), або для кожної альтернативи задається своє конкретне значення.

Візьмемо той самой приклад, який показано в таблиці 2.1 та знайдемо найкращу альтернативу за критерієм Байєса-Лапласа.

Таблиця 2.2

Критерій Байєса-Лапласа

	α_1	α_2	α_3	α_4	α_5	$\sum f_{ij} \cdot p_j$	$Z_{BL} = \max_i f_i$
X_1	104	103	125	125	126	116.82	132.66
X_2	171	108	104	128	133	132.66	
X_3	125	114	113	111	141	120.04	
p_j	0.23	0.16	0.12	0.32	0.17		

Згідно критерію Байєса-Лапласа кращою буде *друга альтернатива*.

Критерій Севіджа шукає можливість пом'якшити концепцію мінімаксного критерію, замінюючи матрицю платежів (прибутків або збитків) на матрицю збитків [9]. Від кожного елемента j стовпця матриці рішень f_{ij} віднімається максимальний результат $\max_j f_{ij}$ відповідного стовпця. Різниці формують матрицю залишків Δ_{ij}. До неї додається стовпець з найбільшою різницею. Обираються ті варіанти, у рядках яких зустрічаються найменші значення для даного стовпця.

Знайдемо найкращу альтернативу згідно критерію Севіджа (див.табл.2.3). Щоб не дублювати дані, матрицю станів візьмемо з таблиці 2.1 або 2.2.

Після проведених розрахунків, можна зробити висновок, що найкращою буде *друга альтернатива*.

Таблиця 2.3

Критерій Севіджа

	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	$f_{ir} = \max_i \Delta_{ij}$	$Z_s = \min_i f_{ir}$
X_1	67	11	0	3	15	67	21
X_2	0	6	21	0	8	21	
X_3	46	0	12	17	0	46	
max_j	171	114	125	128	141		

Критерій песимізму-оптимізму Гурвіца пропонує орієнтуватися на певне середнє значення, яке відображає стан між крайнім песимізмом і необмеженим оптимізмом. Відповідно, критерій визначає альтернативу з максимальним середнім результатом, водночас неявно припускаючи, наскільки ймовірно, що кожен з можливих станів середовища зустрічається з однаковою ймовірністю. З формальної точки зору його можна представити у вигляді наступних формул [8]:

$$f_{ir} = \left(C \cdot \min_j a_{ij} + (1 - C) \max_j a_{ij} \right) \quad (1)$$

$$Z_{HW} = \max_i \left(C \min_j a_{ij} + (1 - C) \max_j a_{ij} \right) \quad (2)$$

де C – це коефіцієнт песимізму, що коливається в межах від 0 до 1. Його значення залежить від оцінки ситуації особою, яка приймає рішення. У разі, коли обирається оптимістичний підхід, то $C > 0.5$, тоді як при песимістичній оцінці $C < 0.5$.

Таблиця 2.4

Критерій Гурвіца (оптимістичний підхід)

	α_1	α_2	α_3	α_4	α_5	f_{ir}	Z_{HW}
X_1	104	103	125	125	126	103	111
X_2	171	108	104	128	133	104	
X_3	125	114	113	111	141	111	

Третя альтернатива найкраща.

В таблицях 2.4 та 2.5 показано знаходження найкращої альтернативи згідно критерію Гурвіца при оптимістичному підході ($C = 1$) та песимістичному підході $C = 0.2$).

Таблиця 2.5

Критерій Гурвіца (песимістичний підхід)

	α_1	α_2	α_3	α_4	α_5	f_{ir}	Z_{HW}
X_1	104	103	125	125	126	121	158
X_2	171	108	104	128	133	158	
X_3	125	114	113	111	141	135	

Друга альтернатива найкраща.

Програмне забезпечення для аналізу та оптимізації вибору в умовах невизначеності за допомогою класичних критеріїв включає мінімакс, Байєса-Лапласа, Севіджа та Гурвіца, що дозволяє проводити багатоаспектний аналіз ризиків і можливостей та дозволяє користувачам оцінювати альтернативи з різних точок зору (песимізм, оптимізм, ймовірнісний підхід, мінімізація жалю).

Цільова аудиторія ПЗ – кожен, хто прагне системного підходу до прийняття рішень в умовах, за яких майбутні події наперед невідомі, однак піддаються оцінці з певною ймовірністю або на підставі песимістичних/оптимістичних припущень.

Незважаючи на свою простоту, оформлення вікна програми добре продумане і орієнтоване на користувача, гарантуючи ясність, легкість вводу і обробки даних, а заодно і ефективну візуалізацію та експорт кінцевих результатів (див.рис.2.1).

Рисунок 2.1 – Діаграма варіантів використання ПЗ

2.2. Проєктування діаграм активностей, послідовностей та класів

Програма для розв'язання задач прийняття рішень в умовах невизначеності починається з вибору користувачем розмірності задачі, а конкретно – кількості альтернатив (варіантів рішень) та кількості станів природи (можливих сценаріїв майбутнього, які не залежать від дій особи, що приймає рішення). Для цього використовуються випадаючі списки (comboAlternatives та comboStates).

Після вибору розмірності інтерфейс динамічно адаптується до форми.

Матриця виграшів (таблиця рішень) є головною таблицею (*dataGrid*), в котру користувач вносить числові значення очікуваних прибутків (або збитків) для кожної альтернативи в кожному стані природи. Рядки таблиці позначені як альтернативи (X_1, X_2, X_3 ...), а стовпці – як стани (α_1, α_2, α_3 ...).

Таблиця ймовірностей реалізована окремою таблицею (*probabilityDataGrid*) з одним рядком для введення ймовірностей кожного стану природи. Після чого програма перевіряє, щоб ці значення перебували в діапазоні від 0 до 1, а їхня сума дорівнювала 1.

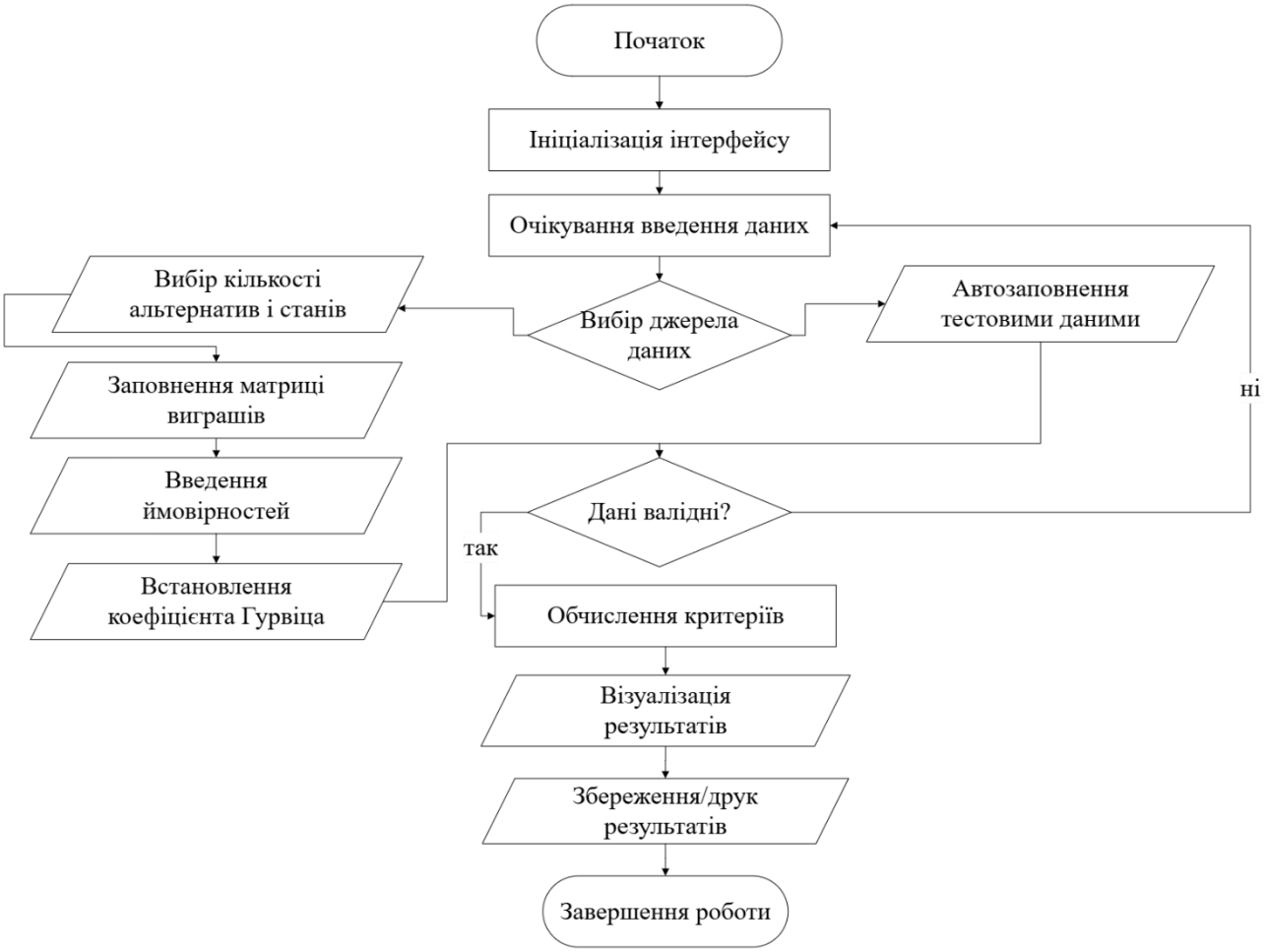


Рисунок 2.2 – Основний алгоритм роботи додатку

Основний клас форми *DecisionAnalysis* містить наступні методи:

– *InitComboBoxes()* для ініціалізації випадаючих списків з вибором кількості альтернатив і станів;

– *ComboBoxes_SelectedIndexChanged()* для оновлення таблиць при зміні вибору в комбобоксах;

– *ProbabilityDataGrid_EditingControlShowing()* для валідації введення ймовірностей (лише числа, крапка);

– *CheckProbabilitiesSum()* для перевірки, чи сума ймовірностей дорівнює 1;

– *GetProbabilities()* для отримання масиву ймовірностей з таблиці;

– *UpdateProbabilityGrid()* для оновлення таблиці ймовірностей при зміні кількості станів;

– *calculateButton_Click()* для обчислення результатів за критеріями;

– *GetMatrix()* для отримання матриці виграшів з *DataGridView*;

– *Minimax(), BayesLaplace(), Savage(), Hurwicz()* для реалізації класичних критеріїв;

– *ShowResults()* для відображення результатів у *RichTextBox*;

– *DrawChart()* для візуалізації результатів у вигляді гістограми;

– saveButton_Click() для збереження даних у Excel;

– *printToPDFMenuItem_Click()* для експорту результатів у PDF;

– *printDocument_PrintPage()* для друку результатів на принтері;

– *buttonExample_Click()* для завантаження тестових даних.

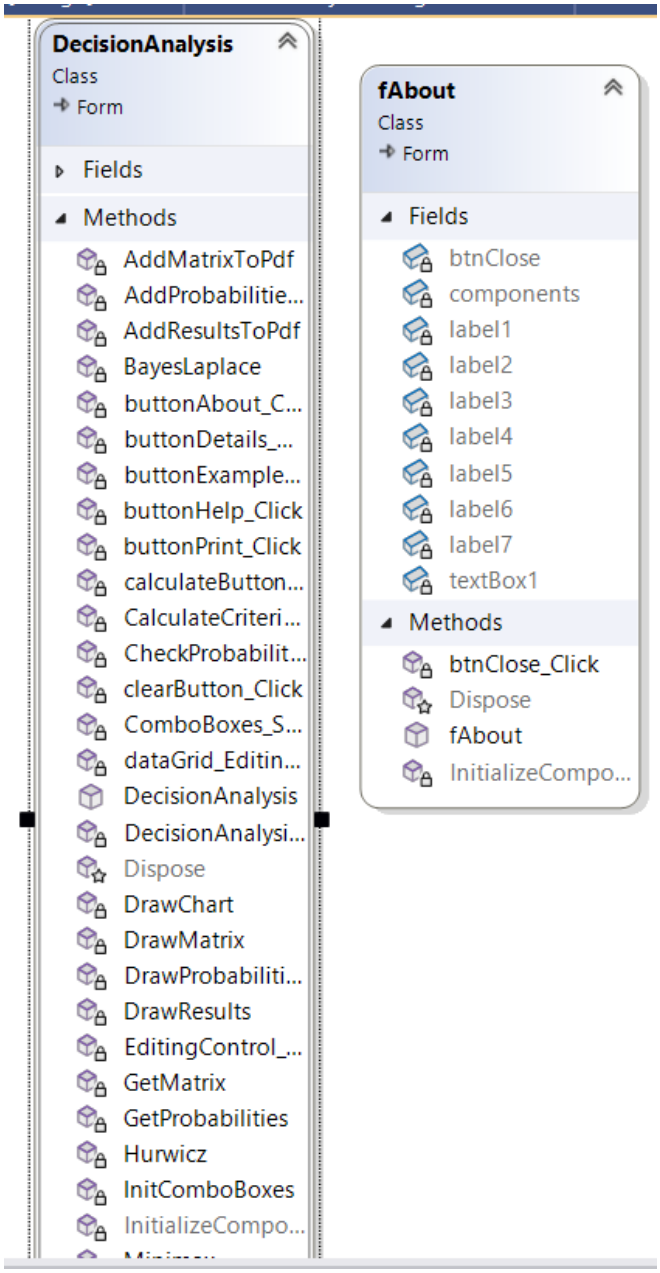


Рисунок 2.3 – Діаграма класів додатку

2.3. Етапи розробки системи. Розробка алгоритму конвертації програмного продукту

Розробку програмного забезпечення для аналізу та оптимізації вибору в умовах невизначеності за допомогою класичних критеріїв розпочинається з налаштування середовища Visual Studio Community [14, 15]. На рисунку 2.4 можна побачити конфігурування проєкту DecisionAnalysis, а саме налаштування іконки додатку, параметри збірки або публікації (здійснюється вибір іконки для програми та перевіряю конфігурацію для .NET Framework 4.8):

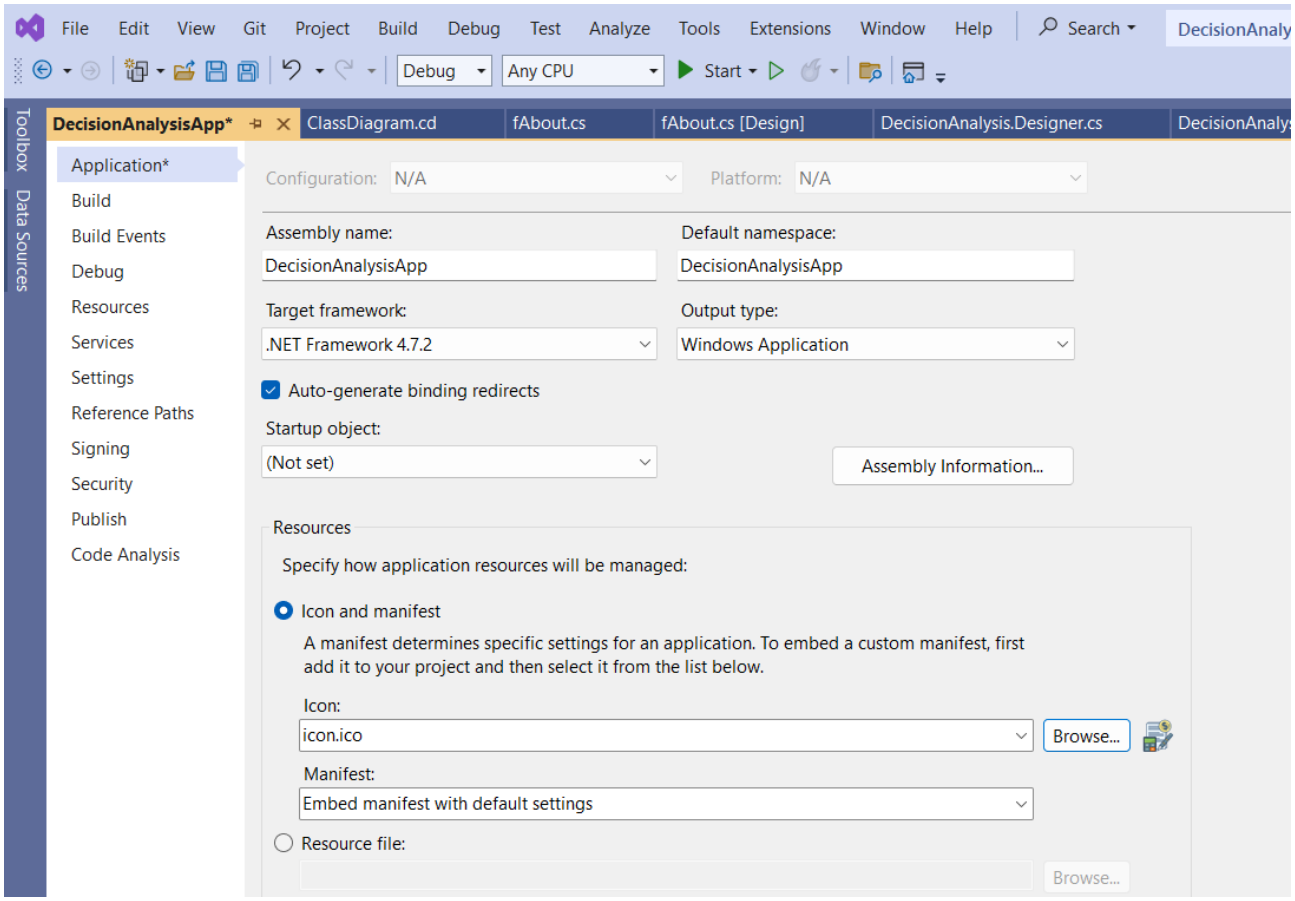
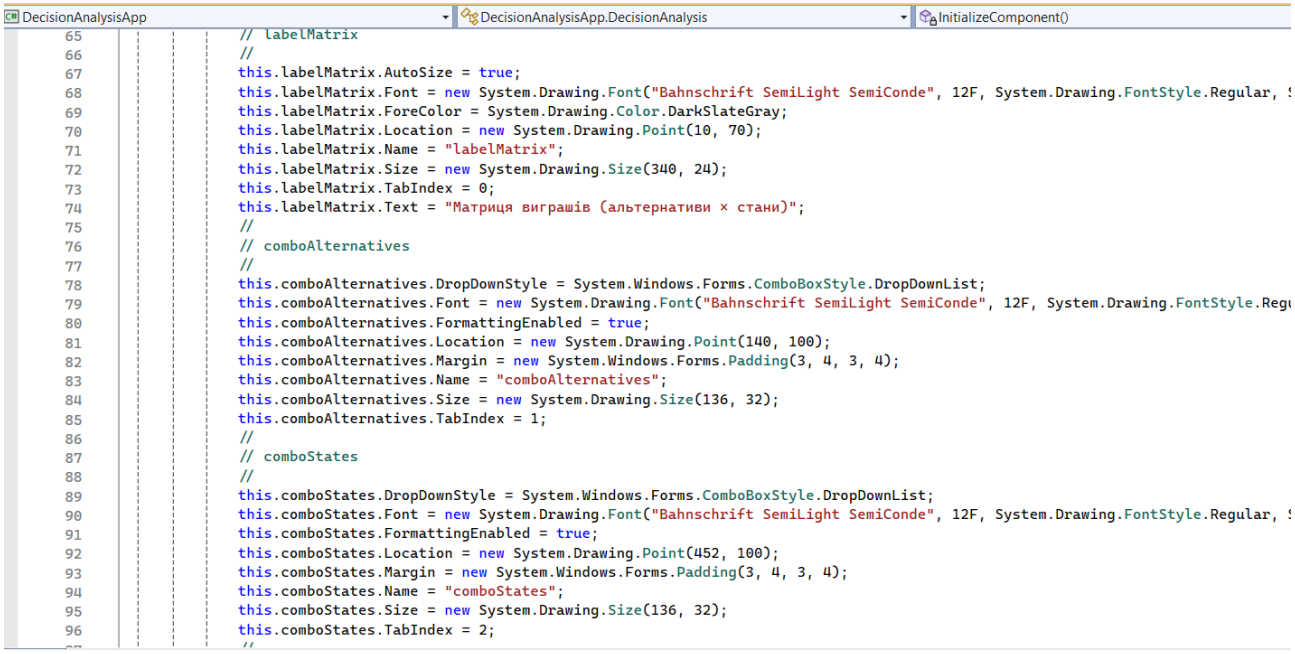


Рисунок 2.4 – Вкладка Application

Для розміщення основних елементів керування (контролів) у формі DecisionAnalysis використано Design Mode (режим контруктора). Завдяки якому швидше можна здійснити проєктування UI без ручного коду (перетягування кнопок, полів, налаштування властивостей тощо). Результат змін автоматично зберігатиметься у пов’язаному файлі DecisionAnalysis.Designer.cs (див.рис.2.5).



```
DecisionAnalysisApp | DecisionAnalysisApp.DecisionAnalysis | InitializeComponent()
65          // labelMatrix
66          //
67          this.labelMatrix.AutoSize = true;
68          this.labelMatrix.Font = new System.Drawing.Font("Bahnschrift SemiLight SemiConde", 12F, System.Drawing.FontStyle.Regular, 
69          this.labelMatrix.ForeColor = System.Drawing.Color.DarkSlateGray;
70          this.labelMatrix.Location = new System.Drawing.Point(10, 70);
71          this.labelMatrix.Name = "labelMatrix";
72          this.labelMatrix.Size = new System.Drawing.Size(340, 24);
73          this.labelMatrix.TabIndex = 0;
74          this.labelMatrix.Text = "Матриця виграшів (альтернативи × стани)";
75          //
76          // comboAlternatives
77          //
78          this.comboAlternatives.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;
79          this.comboAlternatives.Font = new System.Drawing.Font("Bahnschrift SemiLight SemiConde", 12F, System.Drawing.FontStyle.Regu
80          this.comboAlternatives.FormattingEnabled = true;
81          this.comboAlternatives.Location = new System.Drawing.Point(140, 100);
82          this.comboAlternatives.Margin = new System.Windows.Forms.Padding(3, 4, 3, 4);
83          this.comboAlternatives.Name = "comboAlternatives";
84          this.comboAlternatives.Size = new System.Drawing.Size(136, 32);
85          this.comboAlternatives.TabIndex = 1;
86          //
87          // comboStates
88          //
89          this.comboStates.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;
90          this.comboStates.Font = new System.Drawing.Font("Bahnschrift SemiLight SemiConde", 12F, System.Drawing.FontStyle.Regular, 
91          this.comboStates.FormattingEnabled = true;
92          this.comboStates.Location = new System.Drawing.Point(452, 100);
93          this.comboStates.Margin = new System.Windows.Forms.Padding(3, 4, 3, 4);
94          this.comboStates.Name = "comboStates";
95          this.comboStates.Size = new System.Drawing.Size(136, 32);
96          this.comboStates.TabIndex = 2;
```

Рисунок 2.5 – Вміст файлу DecisionAnalysis.Designer.cs

На формі DecisionAnalysis розміщено наступні елементи керування (контроли) (див. рис. 2.6):

– DataGridView (dataGrid) використовується для введення матриці виграшів (альтернативи × стани) та налаштований через AllowUserToAddRows = false;

– DataGridView (probabilityDataGrid) призначений для введення ймовірностей станів та має перевірку на коректність введених даних (сума ймовірностей має бути рівна 1);

– ComboBox (comboAlternatives та comboStates) призначені для вибору кількості альтернатив (2-10) і станів (2-10). В залежності від чого, оновлюють розмір таблиць при зміні;

– NumericUpDown (hurwiczAlpha) використовується для введення коефіцієнта оптимізму для критерію Гурвіца (від 0 до 1);

– Button (calculateButton) – кнопка «Обчислити» для аналізу даних за критеріями;

– Chart (chart) використовується для візуалізації значень критеріїв Гурвіца, Севіджа, мінімаксного та Байєра-Лапласа у вигляді стовпчастої діаграми;

– RichTextBox (resultBox) для відображення результатів аналізу (виведення оптимальних альтернатив для кожного критерію);

– ToolStrip (toolStrip1) – це панель інструментів з кнопками: Очистити дані (toolStripButton1), Зберегти у Excel (saveButton), Друк/PDF (buttonPrint), Довідка (buttonHelp), Інформація про критерії (buttonDetails), Приклад (buttonExample), Про програму (buttonAbout).

Додаткові елементи, які розміщені на формі:

– Label (labelMatrix, label1, label2, label3, label4) відображає текстові підписи для таблиць, комбобоксів та інших контролів;

– PrintDialog та PrintDocument використовуються для друку результатів;

– SaveFileDialog діалогове вікно для збереження даних у Excel/PDF (збереження їх в форматі pdf, якщо принтер не доступний).

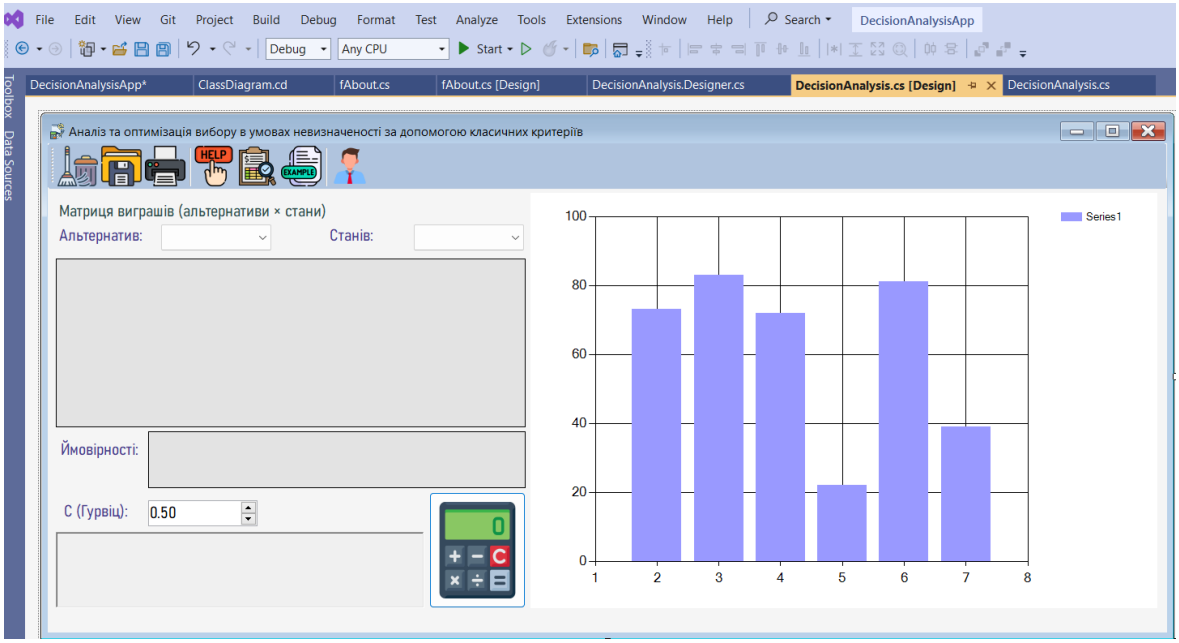


Рисунок 2.6 – Форма DecisionAnalysis в режимі Design Mode

Програмний код проєкту DecisionAnalysis мовою програмування C# наведено в додатках до дипломної роботи.

Важливим аспектом при написанні коду програми стає підключення бібліотек (в тому числі і зовнішніх) та використання вбудованих просторів імен .NET (див. рис. 2.7).

Під час розробки програмного забезпечення для аналізу та оптимізації вибору в умовах невизначеності за допомогою класичних критеріїв використовуються бібліотеки *ClosedXML* (для роботи з Microsoft Excel), *iTextSharp* (для генерації PDF), *System.Windows.Forms.DataVisualization.Charting*

(для виведення діаграм), *SixLabors.Fonts* (для роботи зі шрифтами (використовується разом із *iTextSharp*) [12, 13].

Вбудовані простори імен .NET, які використовуються в проєкті DecisionAnalysis:

– для роботи з даними:
  - using System.Data;
  - using System.Linq;
  - using System.Collections.Generic;
– для роботи з файлами:
  - using System.IO;
– для роботи з графікою:
  - using System.Drawing;
  - using System.Drawing.Drawing2D;
  - using System.Drawing.Printing;
– для інтернаціоналізації:
  - using System.Globalization;
– Для діагностики:
  - using System.Diagnostics;

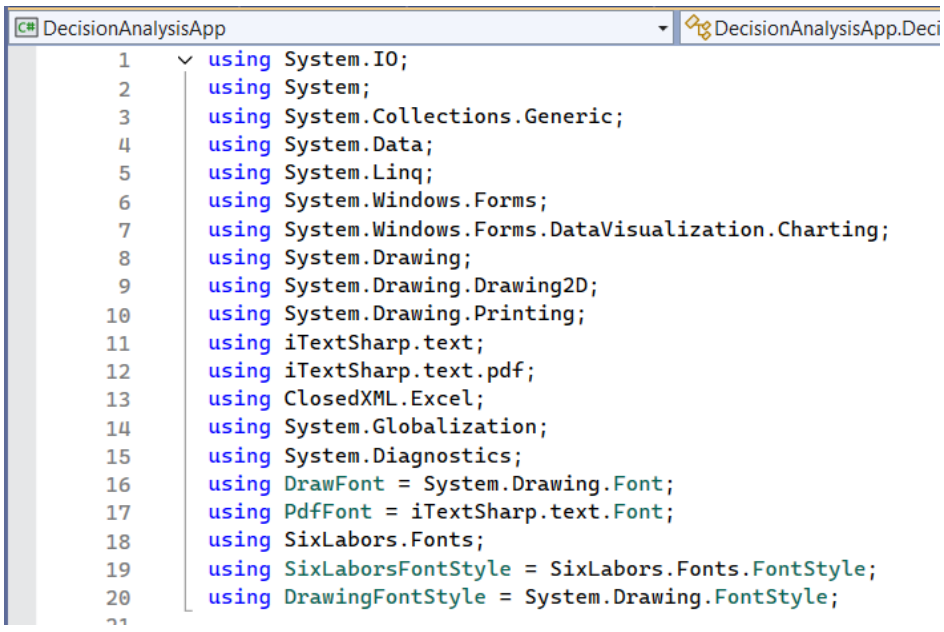


Рисунок 2.7 – Підключені бібліотеки та вбудовані простори імен .NET

Висновки до розділу 2

Програмне забезпечення для аналізу та оптимізації вибору в умовах невизначеності за допомогою класичних критеріїв: мінімакс (визначає максимальне значення серед мінімальних виграшів для кожної альтернативи), Байєса-Лапласа (обраховує математичне очікування виграшу для кожної альтернативи з урахуванням ймовірностей станів та підбирає альтернативу з найбільшим очікуваним значенням), Севіджа (формує матрицю втрат від невдалого рішення та визначає мінімальне значення серед максимальних значень втрат для кожної альтернативи), Гурвіца (розраховує мінімальні та максимальні виграші для кожної альтернативи, використовуючи коефіцієнт песимізму/оптимізму C) є додатком на основі Windows Forms, розробленим на мові $C\#$.

Отримані дані для кожного критерію виводяться у текстовому вікні, зокрема, значення критерію та рекомендована альтернатива (альтернативи). Стовпчаста діаграма наочно ілюструє значення, здобуті для кожного критерію, виділяючи стовпчики різними кольорами та приховавши легенду для кращої читабельності. Гнучкість забезпечується адаптивністю до змін та кількості альтернатив і станів природи, вбудовані підказки та приклади роблять програму доступною навіть для користувачів без спеціальної підготовки.

РОЗДІЛ 3. ОПИС РОБОТИ ПРОГРАМНОГО ДОДАТКУ, ЙОГО ВСТАНОВЛЕННЯ, КОРИСТУВАННЯ ТА ТЕСТУВАННЯ

3.1. Керівництво встановлення програмного продукту

Розроблене програмне забезпечення для аналізу та оптимізації вибору в умовах невизначеності за допомогою класичних критеріїв є готовим до використання, активація чи реєстрація не вимагається. Запускаємо виконуваний файл DecisionAnalysisApp.exe (див.рис.3.1) та розпочинаємо аналіз та оптимізацію вибору за допомогою класичних критеріїв Гурвіца, Севіджа, Байєса-Лапласа та Вадьда (мінімакс).

Технічні характеристики комп'ютера, при яких програма працює стабільно:

– Операційна система: Windows 11 Pro версія 23H2;
– Процесор: AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx 2.10 GHz;
– RAM: 24.0 GB;
– Наявність вільного місця на диску: 100 MB;

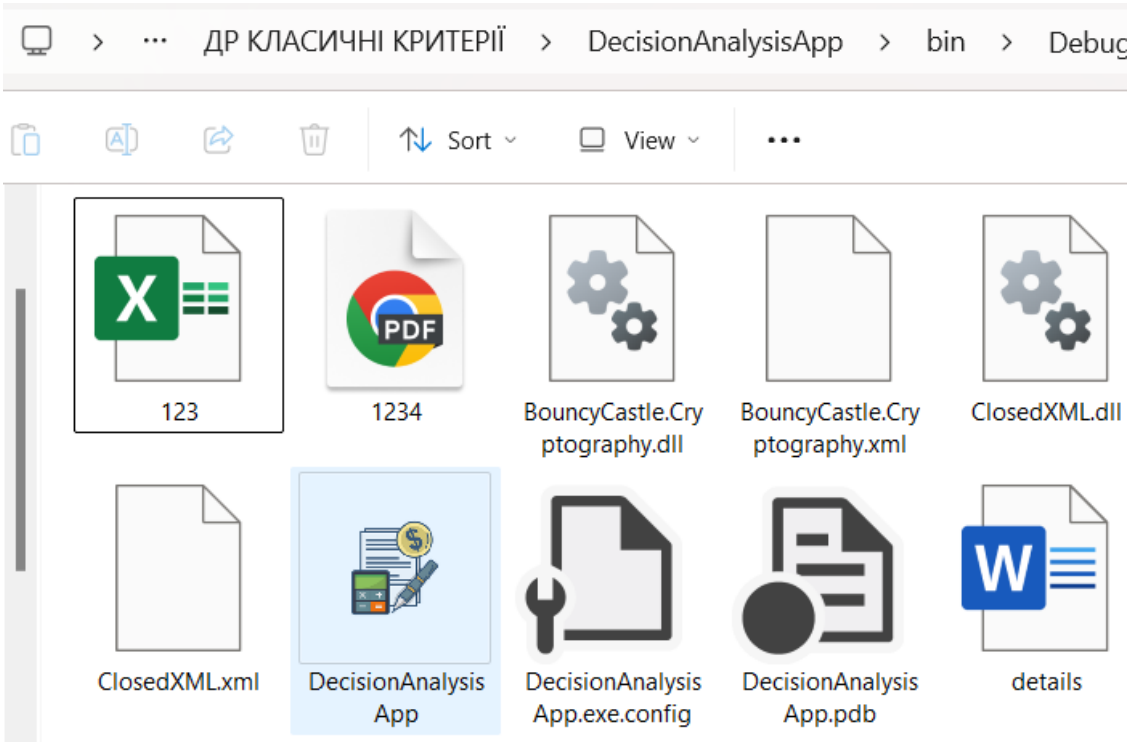


Рисунок 3.1. Запуск програми DecisionAnalysisApp

На рисунку 3.2 показано як виглядає програма, одразу після запуску.

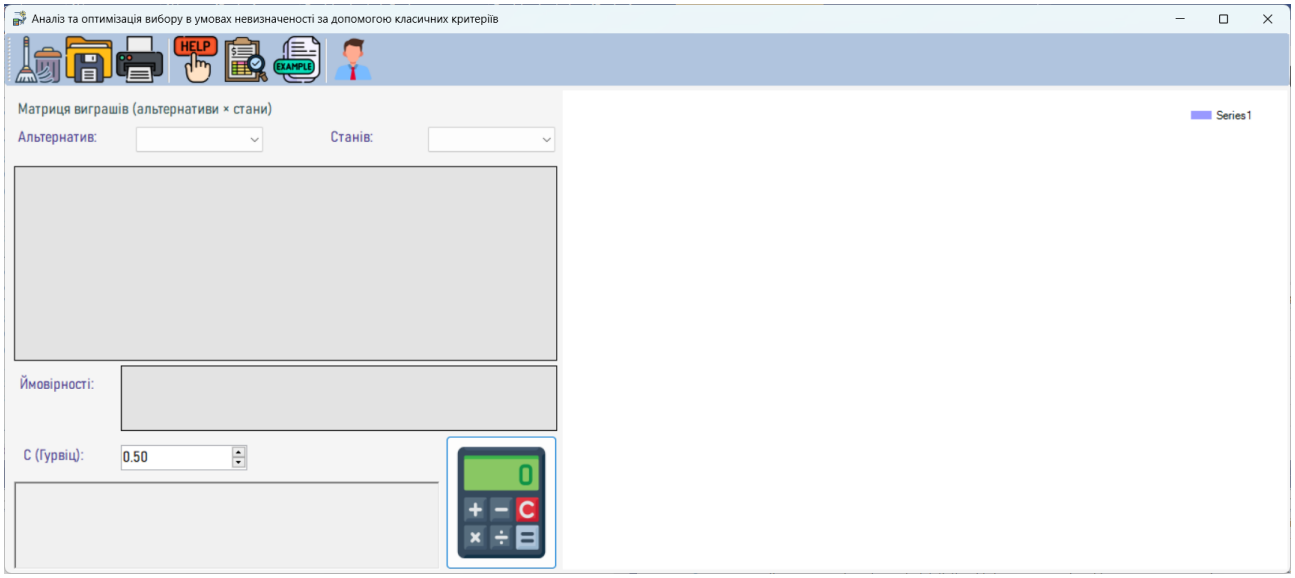


Рисунок 3.2 – Програмне забезпечення для аналізу та оптимізації вибору в умовах невизначеності за допомогою класичних критеріїв

3.2. Керівництво користування програмним продуктом

Вибираємо кількість альтернатив та кількість станів у випадаючих списках «Альтернативи», «Стани» відповідно (див.рис.3.3).

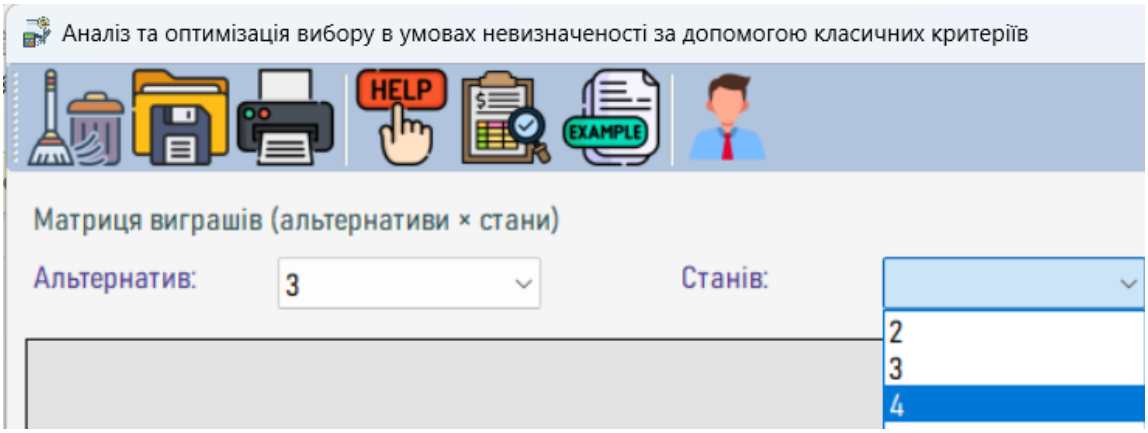


Рисунок 3.3 – Задання кількості станів та альтернатив

Після чого сформується автоматично розмірність таблиць «Матриця виграшів» та «Ймовірності». Кожен рядок представлятиме альтернативу (X_1, X_2, X_3), а кожен стовпець – стан $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$.

При введенні чисел у таблиці використовуємо крапку як десятковий роздільник (наприклад, 0.3, а не 0,3).

Порожні клітинки або некоректні числові значення будуть автоматично інтерпретовані як «0» при розрахунках та виділені червоним (див.рис.3.4).

	a1	a2	a3	a4
X1	102	114	111	125
X2	98	119	ст	
X3	114	115	56	79

Рисунок 3.4 – Невірне введення даних

Далі вводимо ймовірності для кожного стану в діапазоні від 0 до 1. Програма повідомить, якщо сума відрізнятиметься від 1 (більше або менше значення) (див.рис.3.5). Дробові числа вводимо через «крапку».

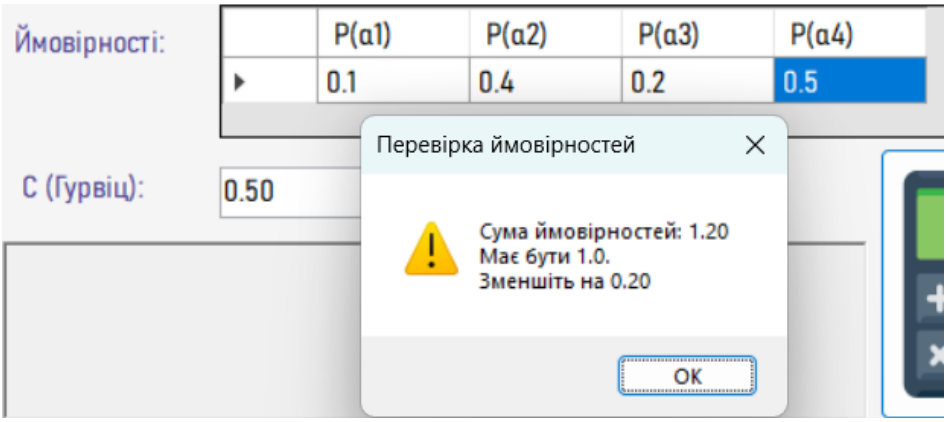


Рисунок 3.5 – Сума ймовірностей більша 1

Провести обчислення можна буде лише після виправлення. Про що успішно буде повідомлено (див.рис.3.6).

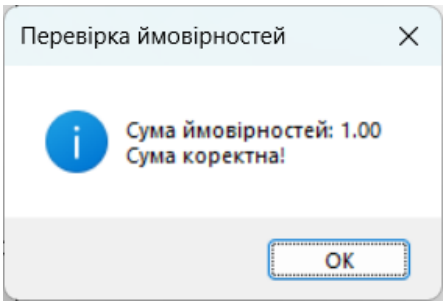


Рисунок 3.5 – Діалогове вікно

Використаємо повзунок або вручну задамо числове значення 0.7 для коефіцієнту Гурвіца від 0 до 1 (див.рис.3.6). По замовчуванню – 0.5.

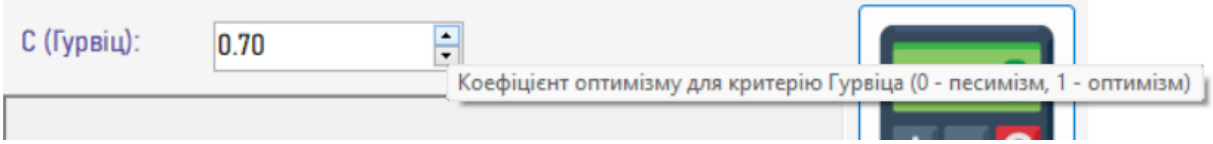


Рисунок 3.6 – Коефіцієнт оптимізму для критерію Гурвіца

Натискаємо кнопку «Обчислити». Кнопка задана графічно із зображенням калькулятора. Програма розрахує оптимальні альтернативи за такими критеріями: мінімакс, Байєса-Лапласа, Севіджа та Гурвіца. Результати будуть відображені у полі *resultBox* (зліва внизу) та візуалізовані на діаграмі (справа) (див.рис.3.7).

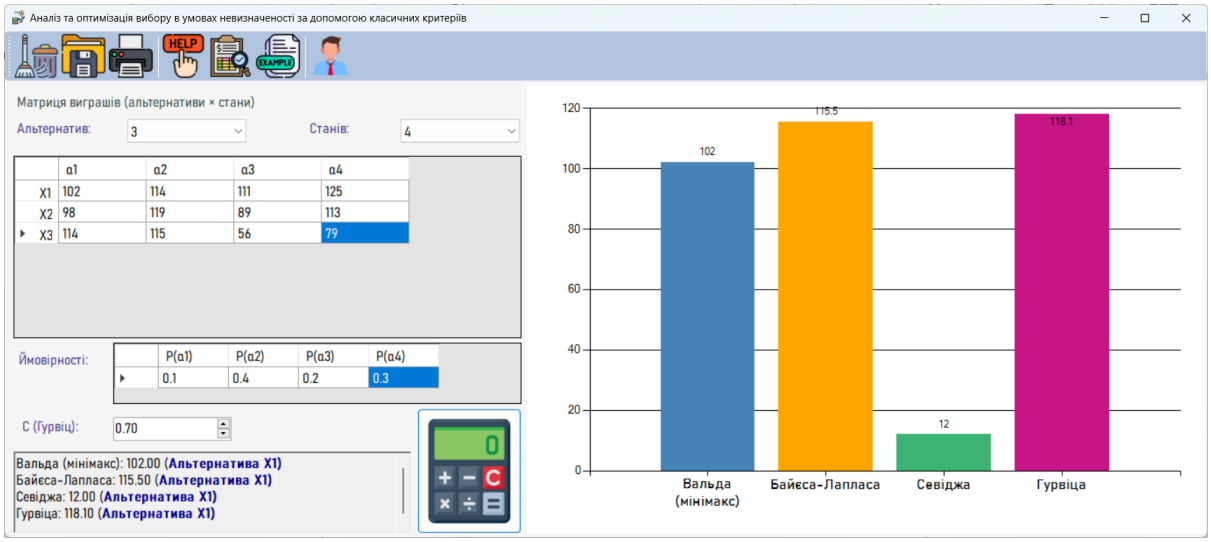


Рисунок 3.7 – Знаходження оптимальних альтернатив для кожного критерію

Якщо виникають питання щодо формату введення даних, натискаємо кнопку з надписом «Help» на панелі інструментів (див.рис.3.8).

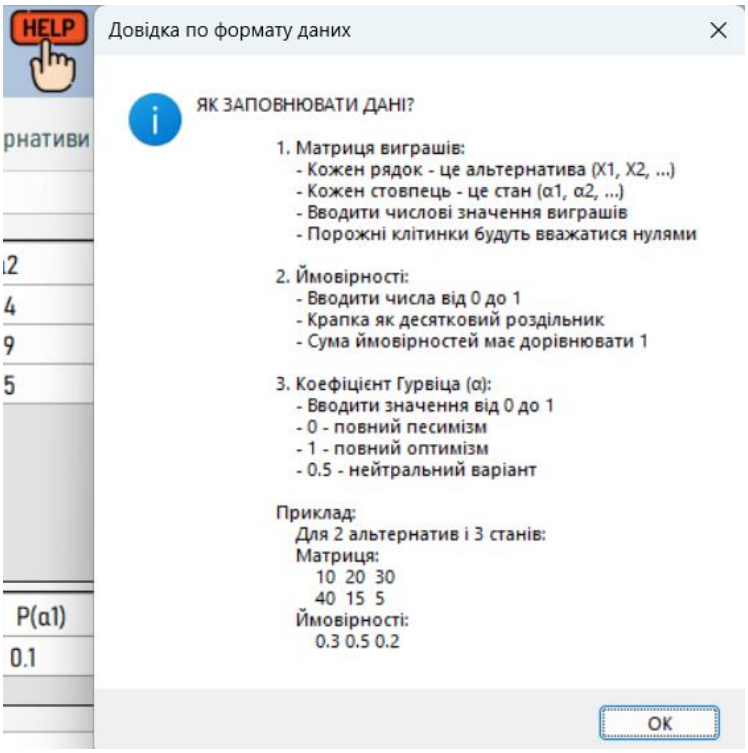


Рисунок 3.8 – Як заповнювати дані?

Щоб дізнатись про розробника програми, скористаємось кнопкою «Довідка про програму» (див.рис.3.9).

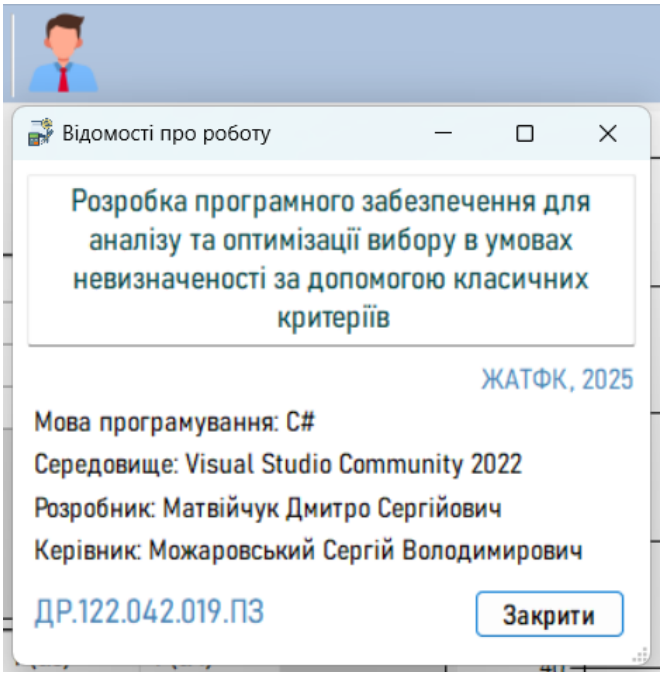


Рисунок 3.8 – Довідкова форма «Відомості про роботу»

Дуже часто, перед початком роботи, потрібно ознайомитись ознайомлення з функціоналом та структурою даних, тому натискаємо кнопку «Example». Оскільки в таблиці вже знаходяться дані, то програма попередить, що дані будуть перезаписані (див.рис.3.9).

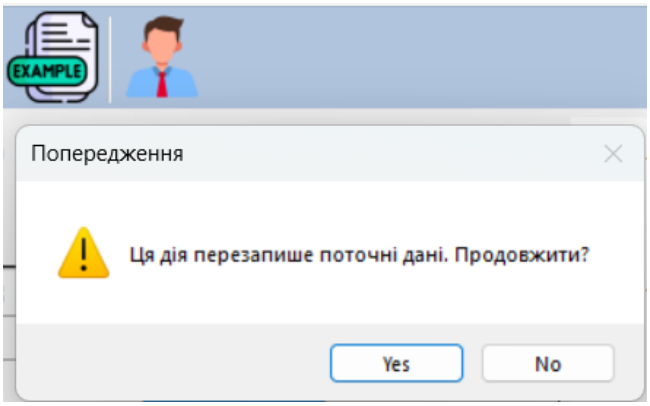


Рисунок 3.9 – Попередження

Натискаємо «Yes» і виводиться приклад з відповідним інформаційним вікном (див.рис.3.10).

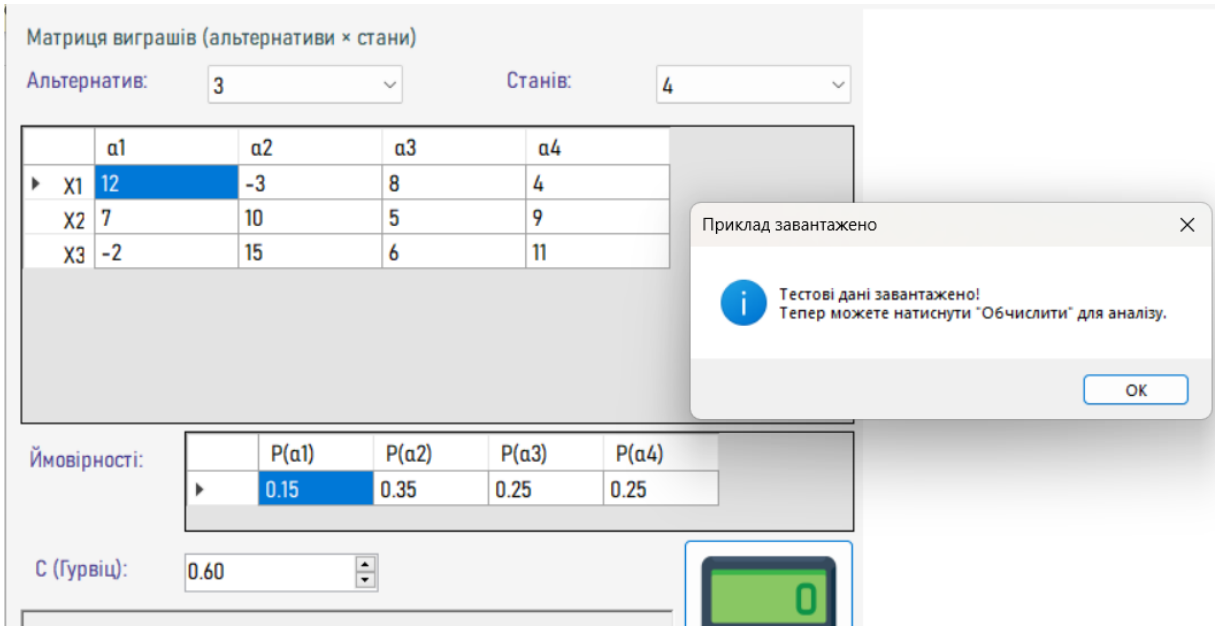


Рисунок 3.10 – Дані для прикладу

Натискаємо кнопку «Обчислити» і одержуємо результат (див.рис.3.11).

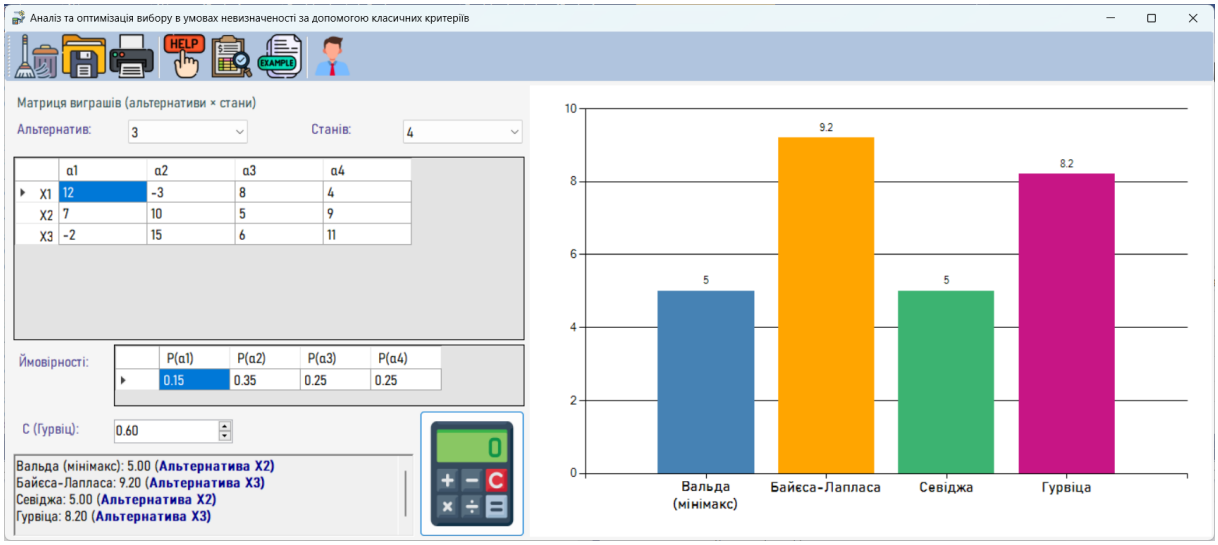


Рисунок 3.11 – Обчислення значень альтернатив для даних з прикладу

Якщо потрібно заново почати обрахунки, натискаємо кнопку «Очистити» (див.рис.3.12). Перед очисткою програма «запитає» чи дійсно ми хочемо очистити дані, оскільки ця дія незворотня і результат обрахунків буде втрачено.

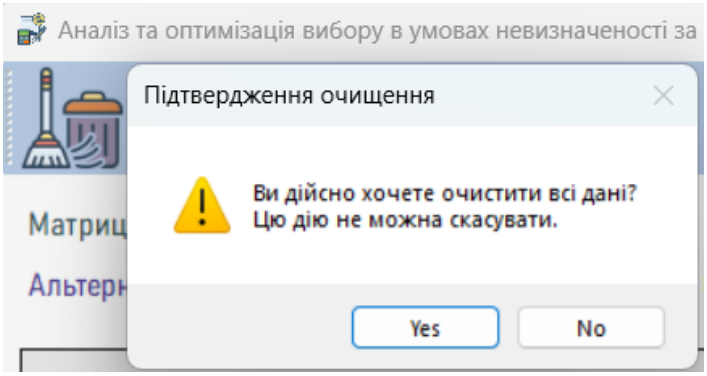


Рисунок 3.12 – Підтвердження очищення

Для збереження результатів у файл Excel натискаємо кнопку «Зберегти» у вигляді папки з дискетою (див.рис.3.13).

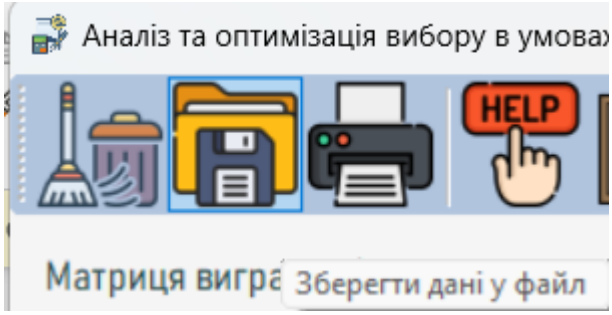


Рисунок 3.13 – Кнопка «Зберегти дані у файл»

Файл буде містити введені дані, результати та діаграму (див.рис.3.15).

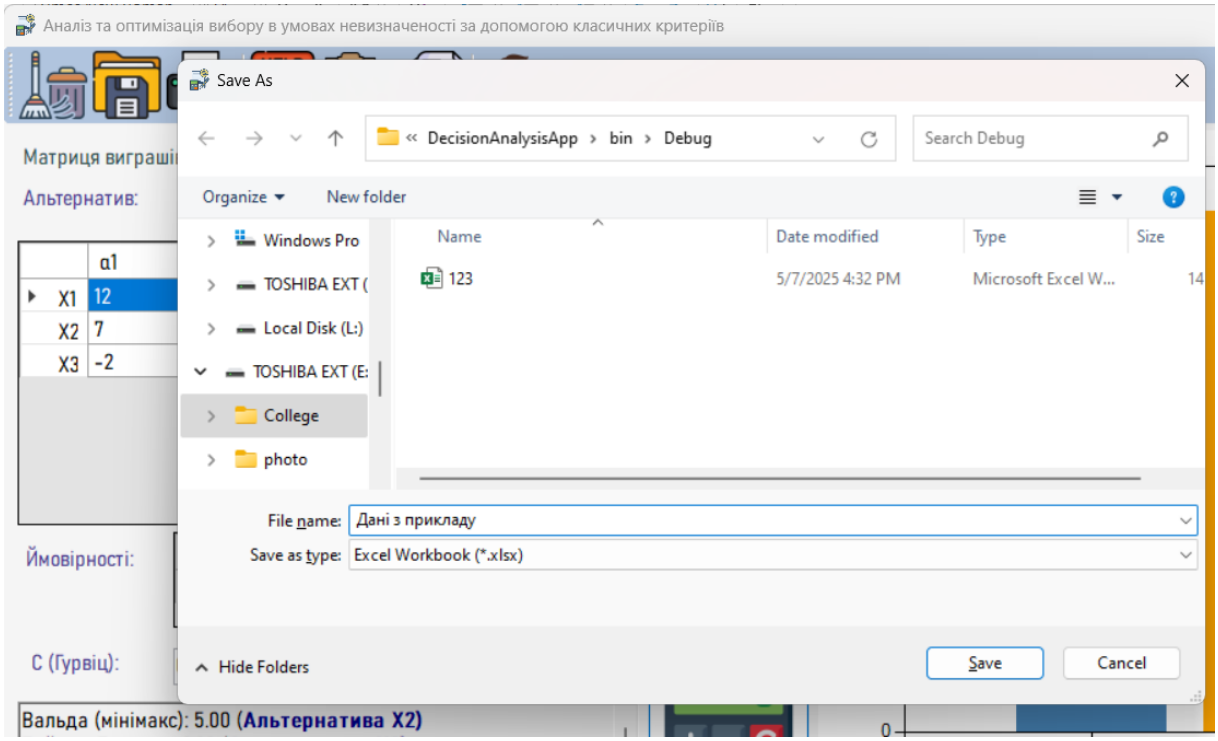


Рисунок 3.14 – Збереження даних у файл з навою «Дані з прикладу»

По замовчуванню файли зберігаються в папку з проєктом.

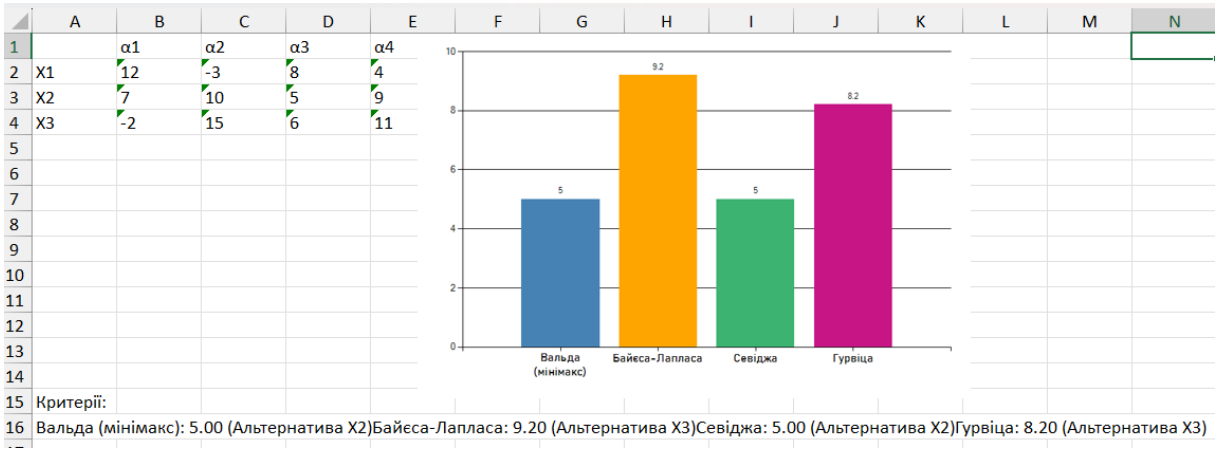


Рисунок 3.15 – Вміст збереженого файлу

Іноді виникає ситуація, коли потрібно більш детально ознайомитися з особливостями того чи іншого критерію. Натискаємо кнопку на панелі інструментів «Інформація про класичні критерії» (див.рис.3.16).

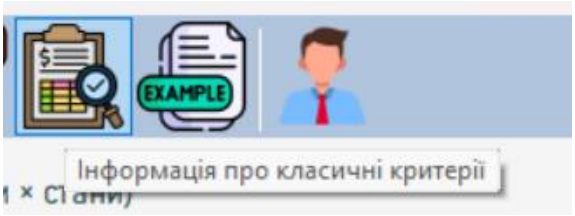


Рисунок 3.16 – Кнопка «Інформація про класичні критерії»

Відкриється текстовий документ (див.рис.3.17) з формулами та описом критерію Вальда (мінімаксного), Гурвіца, Байєса-Лапласа та Севіджа.

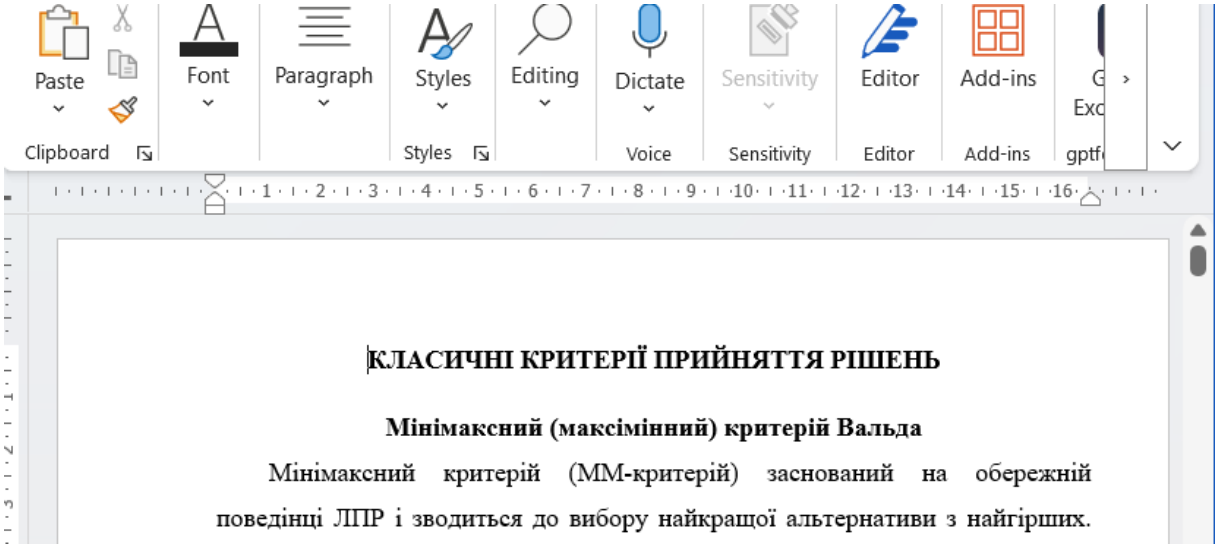


Рисунок 3.17 – Класичні критерії прийняття рішень

В програмі є можливість збереження даних у форматі PDF та друку на принтері (див.рис.3.18).

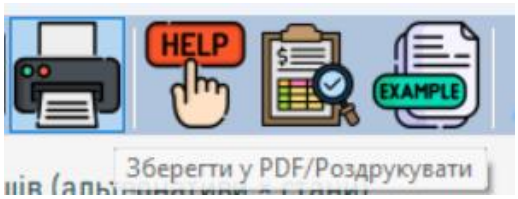


Рисунок 3.18 – Кнопка «Зберегти у PDF/Роздрукувати»

Вибираємо формат та властивості друку (див.рис.3.19).

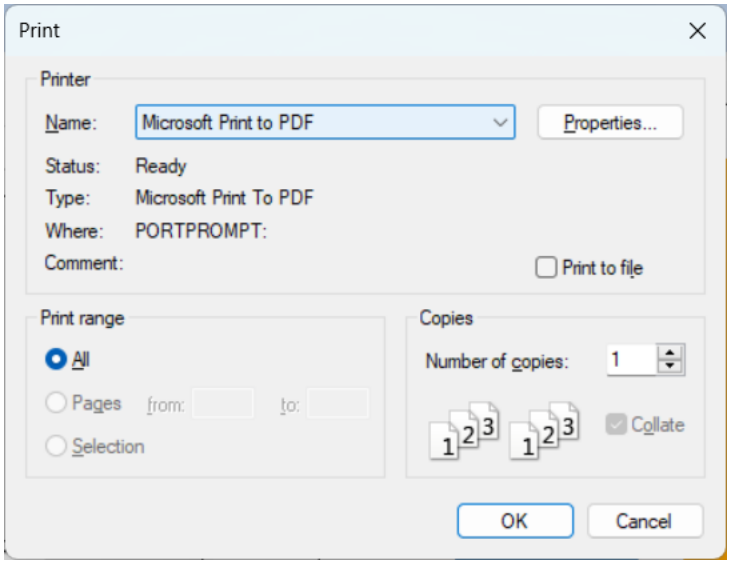


Рисунок 3.19 – Налаштування друку

Перш, ніж надрукувати результати, з’явиться інформаційне повідомлення з вимогою підтвердження (див.рис.3.20).

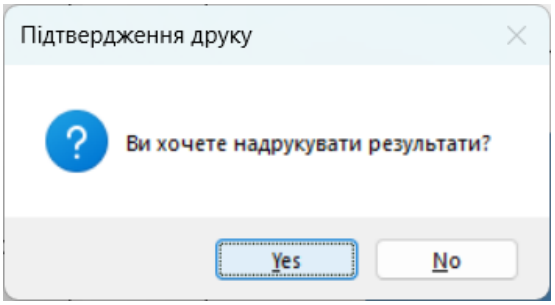


Рисунок 3.20 – Ви хочете надрукувати результати?

Після чого, зявиться вікно з можливістю збереження документу. Назвемо файл «Мій документ» (див.рис.3.21). По замовчуванню він буде разом з проєктом DecisionAnalysisApp.exe. Розміщення можна змінити.

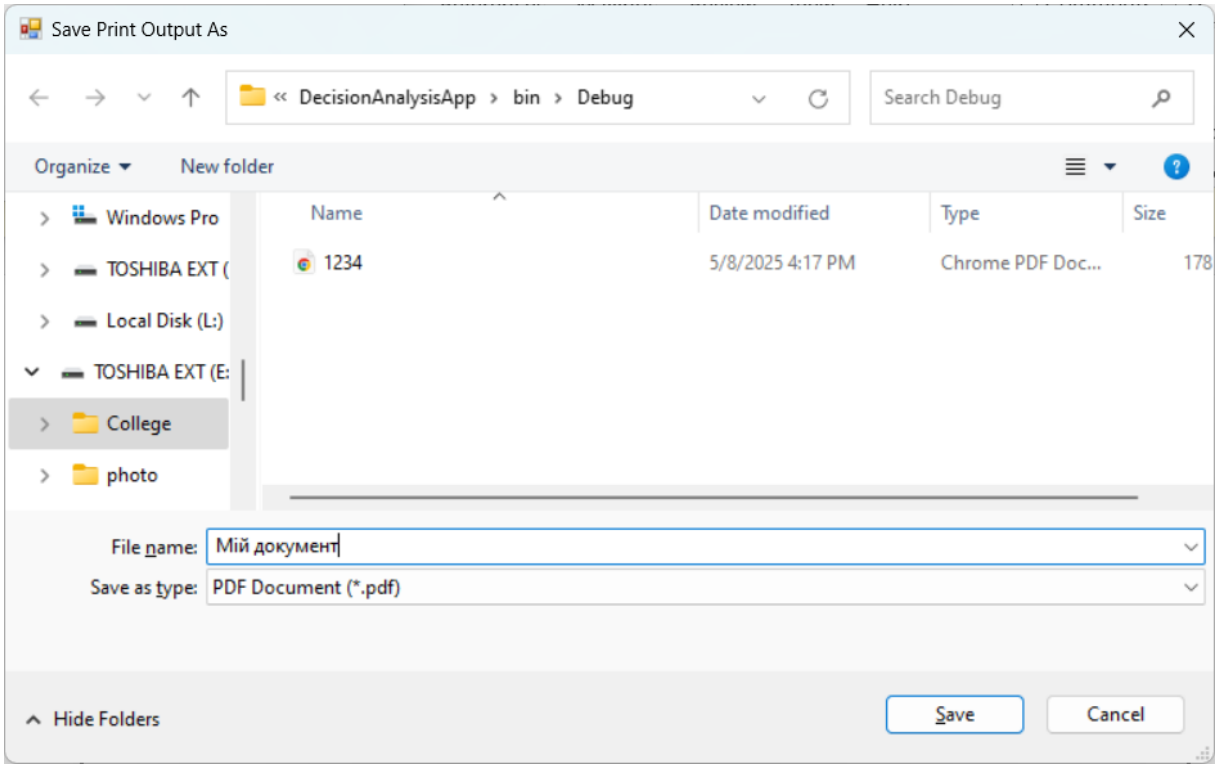


Рисунок 3.21 – Файл «Мій документ» з розширенням PDF

Переглянемо вміст файлу (див.рис.3.22).

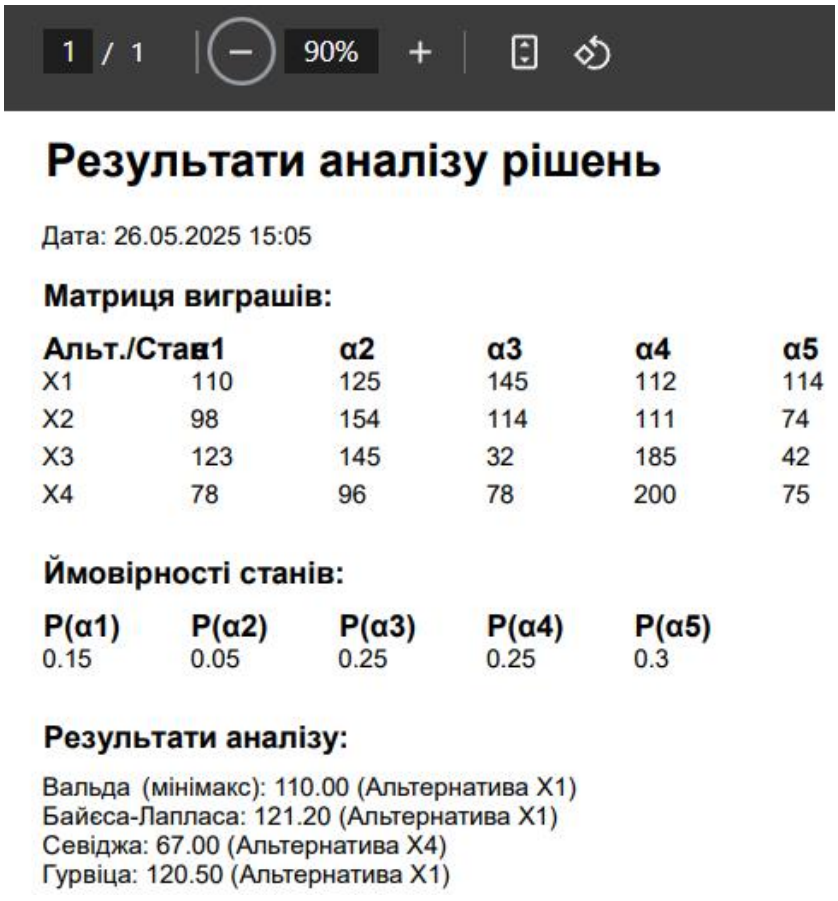


Рисунок 3.22 – Результати аналізу рішень

3.3. Тестування роботи програмного продукту

Переконаємося, що критерії (Вальда (мінімакс), Байєса-Лапласа, Севіджа, Гурвіца) обчислюються без помилок, оптимальні альтернативи визначаються вірно, оскільки невірні розрахунки можуть призводити до неправильних висновків, які, в свою чергу, матимуть серйозні наслідки в реальних умовах.

Таблиця 3.1

Тестування програмного продукту «DecisionAnalysisApp»

Категорія тесту	Тестовий сценарій	Очікуваний результат	Результат
Валідація даних	Введення тексту («ааа») у матрицю виграшів	Помилка: червоний фон, дані не приймаються	+
	Введення ймовірності 1.5 > 1	Помилка: «Значення має бути від 0 до 1»	+
	Сума ймовірностей ≠ 1	Попередження: «Сума має дорівнювати 1.0»	+
Функціональність	Критерій Мінімакс $\begin{pmatrix} 100 & 150 & 200 \\ -50 & 190 & 300 \\ 230 & 10 & 200 \end{pmatrix}$	Результат: 100 (Альтернатива X_1)	+
	Критерій Байєса-Лапласа $P_{\alpha_j}(0.1; 0.25; 0.65)$	Результат: 237.5 (Альтернатива X_2)	+
	Критерій Севіджа $\begin{pmatrix} 100 & 150 & 200 \\ -50 & 190 & 300 \\ 230 & 10 & 200 \end{pmatrix}$	Результат: 130 (Альтернатива X_1)	+
	Критерій Гурвіца $C = 0.42$ $\begin{pmatrix} 100 & 150 & 200 \\ -50 & 190 & 300 \\ 230 & 10 & 200 \end{pmatrix}$	Результат: 142 (Альтернатива X_1)	+

Продовження таблиці 3.1

Експорт даних	Експорт до Excel (.xlsx)	Файл містить матрицю, результати та графік	+
	Експорт до PDF	PDF з читабельним аналізом	+
Інтерфейс (UI)	Кнопка 	Коректний розрахунок і вивід результатів	+
	Кнопка 	Очищення всіх полів	+
	Кнопка 	Відкриття вікна з інформацією про критерії	+
	Кнопка 	Автозаповнення матриці виграшів тестовими даними	+
	Кнопка 	Відкриття вікна «Як працювати з програмою?»	+
	Кнопка 	Відкриття форми з інформацією про розробника дипломної роботи	+

Продовження таблиці 3.1

	Кнопка 	Відкриття діалогу друку	+
	Кнопка 	Створення файлу у вибраному форматі	+
	Діаграма результатів	Відображає стовпці для кожного критерію	+
Стабільність	Відкриття без встановлених бібліотек PDF/Excel	Попередження: «Не вдалося експортувати»	+
	Закриття під час обчислень	Програма не зависає, дані не втрачаються	+
Продуктивність	Матриця 10x10	Час обчислення < 0.5 сек	+
	Матриця 20x20	Час обчислення < 2 сек	±

Висновки до розділу 3

У третьому розділі детально описано процес використання й тестування розробленого програмного продукту DecisionAnalysisApp. Надано покрокову інструкцію для користувача, зокрема введення інформації, обчислення результатів за класичними критеріями ухвалення рішень (Вальда, Севіджа, Байєса-Лапласа, Гурвіца), а потім – здатність збереження результатів у форматах Excel та PDF, друку й побудови діаграм. Тестування засвідчило правильність роботи програми у різних ситуаціях: з різними обсягами даних, під час введення хибних значень, порушенні вимог до ймовірностей ($\sum \neq 1$). Програма демонструє високу стабільність, точність обрахунків й зручність в роботі.

ВИСНОВКИ

У першому розділі було визначено мету розробки – створення програмного забезпечення для аналізу та оптимізації вибору в умовах невизначеності на основі критеріїв Вальда, Севіджа, Гурвіца та Байєса-Лапласа. Описано особливості кожного критерію, обґрунтовано їх вибір та передбачено обробку матриці виграшів, перевірку введених даних (зокрема, суму ймовірностей), підтримку динамічної зміни кількості альтернатив і станів, копіювання в Excel, виведення результатів у вигляді тексту й гістограм, а також генерацію PDF-звітів. У розділі також проаналізовано аналогічні програмні продукти: SAS, SPSS і Excel. Зроблено висновок про доцільність створення власного десктопного рішення. Обрано C# і платформу .NET Framework для реалізації, Visual Studio як середовище розробки, а також бібліотеки ClosedXML, iTextSharp і Charting для роботи з Excel, PDF та побудови графіків. Такий стек дозволив створити інтуїтивний інтерфейс, реалізувати всі математичні обчислення та забезпечити повноцінну взаємодію з користувачем.

У другому розділі дипломної роботи докладно описані основні критерії, покладені в основу розробленого мною програмного забезпечення.

Для критерію Вальда (Мінімакс) алгоритм визначає мінімальний виграш для кожної альтернативи, після чого обирається та альтернатива, для якої цей виграш є максимальним з усіх мінімальних значень.

Критерій Байєса-Лапласа ґрунтується на підрахунку математичного сподівання виграшу по кожній альтернативі шляхом зважування виграшів на відповідні ймовірності станів, з обов'язковою перевіркою суми ймовірностей на рівність одиниці; оптимальною буде визнана та альтернатива, для якої середньозважений виграш є найбільшим.

Алгоритм критерію Севіджа (мінімуму жалю) зводиться до побудови матриці втрат (жалю) через віднімання фактичного виграшу від максимально можливого виграшу для кожного стану; при цьому подальший вибір відбуватиметься шляхом знаходження максимального значення жалю для

кожної альтернативи і вибору альтернативи з мінімальним з усіх максимальних значень.

Для критерію Гурвіца розрахунок зваженої суми максимального та мінімального виграшів для кожної альтернативи здійснюється з урахуванням заданого коефіцієнта оптимізму/песимізму C, причому оптимальною вважається альтернатива з найбільшим значенням цієї суми.

У третьому розділі показано загальний вигляд програми після запуску, включаючи скріншот головного вікна та детальний опис його елементів керування. Інтерфейс містить випадаючі списки для вибору кількості станів та альтернатив (розмірів матриці), динамічні DataGridView для введення матриці виграшів/втрат та ймовірностей станів, а також поле для введення коефіцієнта Гурвіца. Для керування програмою на панелі інструментів є графічні кнопки: «Обчислити» (запускає розрахунки), «Очистити» (скидає дані), «Зберегти» (зберігає поточний стан), «Приклад» (завантажує демо-дані), «Інформація про класичні критерії» (інформація про те, як знаходиться оптимальна альтернатива для кожного критерію), «Довідка про програму» (інформація про розробника). Результати відображаються у текстовій області та візуалізуються за допомогою компонента Chart. Додаткові можливості доступні через кнопки: «Зберегти у PDF/Роздрукувати», та «Зберегти дані у файл».

Крім того, в даному розділі було проведено серію тестів з коректним та некоректним вводом (нечислові значення в матриці виграшів, невірна сума ймовірностей в таблиці ймовірностей) з оцінкою відповідних повідомлень про помилки та блокування роботи програми.

Всупереч надскладним комерційним рішенням, розроблене програмне забезпечення для аналізу та оптимізації вибору в умовах невизначеності за допомогою класичних критеріїв Вальда, Севіджа, Гурвіца та Байєса-Лапласа дозволяє зайняти нішу доступного, інтуїтивно зрозумілого та багатофункціонального інструменту, що чітко реалізовує основоположні принципи класичного підходу до аналізу. Його перевагою є орієнтованість на легкість використання, дієва наочність та надійність в обробці даних, завдяки

чому він ідеально підходить для освітніх цілей, оперативного аналізу та ухвалення рішень у невеликих за обсягом, але критично важливих ситуаціях. Завдяки можливості експорту в Excel і PDF, а також друкування, підвищується його практична цінність, адже він дає змогу легко вбудовувати результати в інші робочі процеси та звітність.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Платформа SAS Risk Management: веб-сайт. https://www.sas.com/en_us/solutions/risk-management.html (дата звернення: 22.03.2025).
2. Платформа Chartis RiskTech Quadrant: веб-сайт. URL: https://www.sas.com/content/dam/SAS/documents/analyst-reports-papers/en/chartis-risktech-alm-solutions-2024.pdf (дата звернення: 22.03.2025)
3. Платформа IBM: веб-сайт. URL: https://www.ibm.com/products/spss-statistics/decision-trees: 22.03.2025)
4. Платформа IBM SPSS Statistics: веб-сайт. URL: https://www.ibm.com/products/spss-statistics#trial (дата звернення: 22.03.2025)
5. Платформа SciPy: веб-сайт. URL: https://scipy.org/ (дата звернення: 23.03.2025)
6. Платформа Sonatafy Technology: веб-сайт. URL: https://sonatafy.com/what-is-net-framework/ (дата звернення: 01.04.2025)
7. Платформа Microsoft: веб-сайт. URL: https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework (дата звернення: 01.04.2025)
8. Платформа MIT Press Direct: веб-сайт. URL: https://direct.mit.edu/books/monograph/4074/Decision-Making-Under-UncertaintyTheory-and (дата звернення: 10.04.2025).
9. Платформа Wharton: веб-сайт. URL: https://executiveeducation.wharton.upenn.edu/thought-leadership/wharton-at-work/2021/08/decision-making-under-uncertainty/ (дата звернення: 10.04.2025).
10. Платформа Imperial College London: веб-сайт. URL: https://www.imperial.ac.uk/stories/decision-making/ (дата звернення: 10.04.2025)
11. Платформа ResearchGate: веб-сайт. URL: https://www.researchgate.net/publication/369452739_METHODS_FOR_DECISION-MAKING_UNDER_CONDITIONS_OF_UNCERTAINTY_AND_RISK (дата звернення: 10.04.2025)

12. Платформа GeeksforGeeks: веб-сайт URL: https://www.geeksforgeeks.org/c-sharp-namespaces/ (дата звернення: 15.04.2025)
13. Платформа IBM: веб-сайт. URL: https://www.ibm.com/docs/en/db2/11.1.0?topic=net-data-server-provider-namespaces (дата звернення: 15.04.2025)
14. Платформа Microsoft: веб-сайт. URL: https://learn.microsoft.com/en-us/shows/csharp-fundamentals-for-absolute-beginners/understanding-namespaces-and-working-with-the-dotnet-class-library (дата звернення: 13.04.2025)
15. Платформа DEV Community: веб-сайт. URL: https://dev.to/teshanecrawford/demystifying-namespace-naming-in-c-and-net-2d2n (дата звернення: 13.04.2025)
16. Платформа Softonic: веб-сайт. URL: https://spss.en.softonic.com/ (дата звернення: 01.05.2025)
17. Платформа University of Sussex: веб-сайт URL: https://www.sussex.ac.uk/its/pdfs/SPSS_Decision_Trees_21.pdf (дата звернення: 01.05.2025)
18. Платформа Smart Vision Europe: веб-сайт URL: https://www.sv-europe.com/product/ibm-spss-decision-trees-authorised-user-fixed-term-license/ (дата звернення: 09.05.2025)
19. Платформа SPSS Analytics Partner: веб-сайт. URL: https://www.spssanalyticspartner.com/learning-hub/tutorials/running-decision-trees-part-1/ (дата звернення: 10.05.2025)
20. Платформа NuGet: веб-сайт. URL: https://www.nuget.org/packages/closedxml/ (дата звернення: 18.05.2025)

ДОДАТКИ

Додаток А

Програмний код

Лістинг DecisionAnalysis.cs

```
using System.IO;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Printing;
using iTextSharp.text;
using iTextSharp.text.pdf;
using ClosedXML.Excel;
using System.Globalization;
using System.Diagnostics;
using DrawFont = System.Drawing.Font;
using PdfFont = iTextSharp.text.Font;
using SixLabors.Fonts;
using SixLaborsFontStyle = SixLabors.Fonts.FontStyle;
using DrawingFontStyle = System.Drawing.FontStyle;
namespace DecisionAnalysisApp
{
    public partial class DecisionAnalysis : Form
    {       private PrintDialog printDialog;
        private PrintDocument printDocument;
        public DecisionAnalysis()
        {  InitializeComponent();
```

```
            InitComboBoxes();
            // Налаштування DataGridView
            dataGrid.AllowUserToAddRows = false;
            probabilityDataGrid.AllowUserToAddRows = false;
            // Підключення обробників подій
            probabilityDataGrid.EditingControlShowing +=
ProbabilityDataGrid_EditingControlShowing;
            probabilityDataGrid.CellEndEdit += ProbabilityDataGrid_CellEndEdit;
            dataGrid.EditingControlShowing += dataGrid_EditingControlShowing;
            // dataGrid.CellValidating += dataGrid_CellValidating;
            // Initialize print components
            printDialog = new PrintDialog();
            printDocument = new PrintDocument();
            printDocument.PrintPage += printDocument_PrintPage;        }
        private void InitComboBoxes()        {
            for (int i = 2; i <= 10; i++)          {
                comboAlternatives.Items.Add(i);
                comboStates.Items.Add(i);          }
            comboAlternatives.SelectedIndexChanged +=
ComboBoxes_SelectedIndexChanged;
            comboStates.SelectedIndexChanged +=
ComboBoxes_SelectedIndexChanged;        }
        private void ComboBoxes_SelectedIndexChanged(object sender, EventArgs e)
{   if (comboAlternatives.SelectedItem == null || comboStates.SelectedItem == null)
                return;
            int rows = int.Parse(comboAlternatives.SelectedItem.ToString());
            int cols = int.Parse(comboStates.SelectedItem.ToString());
            dataGrid.ColumnCount = cols;
            dataGrid.RowCount = rows;
            for (int i = 0; i < cols; i++)
```

```
            dataGrid.Columns[i].HeaderText = $"α{i + 1}";
        for (int i = 0; i < rows; i++)
            dataGrid.Rows[i].HeaderCell.Value = $"X{i + 1}";
        // Оновлюємо таблицю ймовірностей
        UpdateProbabilityGrid(cols); // Передаємо кількість станів (cols)
    }
    private void ProbabilityDataGrid_EditingControlShowing(object sender,
DataGridViewEditingControlShowingEventArgs e)
    {   if (e.Control is TextBox textBox)
        {
            textBox.KeyPress += (s, args) =>
            {
                // Дозволяємо: цифри (0-9), крапку, Backspace
                if (!char.IsDigit(args.KeyChar) &&
                    args.KeyChar != '.' &&
                    args.KeyChar != '\b')                {
                    args.Handled = true;                }
                // Забороняємо кілька крапок
                if (args.KeyChar == '.' && (s as TextBox).Text.Contains('.'))
                {                    args.Handled = true;
                }            };          }       }
    private void ProbabilityDataGrid_CellEndEdit(object sender,
DataGridViewCellEventArgs e)       {
        if (probabilityDataGrid.Rows[e.RowIndex].Cells[e.ColumnIndex].Value !=
null)
        {              string value =
probabilityDataGrid.Rows[e.RowIndex].Cells[e.ColumnIndex].Value.ToString();
            if (!double.TryParse(value, out double result) || result < 0 || result > 1)
            {                MessageBox.Show("Введіть число від 0 до 1");
```

```
                probabilityDataGrid.Rows[e.RowIndex].Cells[e.ColumnIndex].Value =
0;              }           }
            // Автоматична перевірка суми
            CheckProbabilitiesSum();        }
        private void CheckProbabilitiesSum()
        {           try
            {
                if (probabilityDataGrid.ColumnCount == 0 ||
probabilityDataGrid.Rows.Count == 0)
                {                   MessageBox.Show("Таблиця ймовірностей порожня");
                    return;
                }
                double sum = 0;
                int validCells = 0;
                for (int i = 0; i < probabilityDataGrid.ColumnCount; i++)               {
                    var cell = probabilityDataGrid.Rows[0].Cells[i];
                    if (cell.Value == null ||
string.IsNullOrWhiteSpace(cell.Value.ToString()))
                    {               cell.ErrorText = "Введіть значення ймовірності";
                        continue;                   }
                    if (!double.TryParse(cell.Value.ToString(), out double val))
                    {                   cell.ErrorText = "Невірний формат числа";
                        continue;                           }
                    if (val < 0 || val > 1)
                    {                       cell.ErrorText = "Значення має бути від 0 до 1";
                        continue;                   }
                    cell.ErrorText = string.Empty;
                    sum += val;
                    validCells++;
                }
```

```
            // Show sum in MessageBox instead of label
            string message = $"Сума ймовірностей: {sum:F2}";
            if (validCells < probabilityDataGrid.ColumnCount)
            {                   message += "\nНе всі значення валідні!";
            }               else if (Math.Abs(sum - 1.0) > 0.0001)
            {
               message += $"\nМає бути 1.0.\n";
               message += sum < 1.0
                  ? $"Додайте ще {(1.0 - sum):F2}"
                  : $"Зменшіть на {(sum - 1.0):F2}";               }
            else               {
               message += "\nСума коректна!";               }
            MessageBox.Show(message, "Перевірка ймовірностей",
MessageBoxButtons.OK,
               Math.Abs(sum - 1.0) > 0.0001 ? MessageBoxIcon.Warning :
MessageBoxIcon.Information);           }
         catch (Exception ex)           {
            MessageBox.Show($"Помилка обчислення: {ex.Message}", "Помилка",
                  MessageBoxButtons.OK, MessageBoxIcon.Error);
         }       }
      private double[] GetProbabilities(int count)
      {
         double[] probs = new double[count];
         double sum = 0;
         for (int i = 0; i < count; i++)
         {
            var cellValue = probabilityDataGrid.Rows[0].Cells[i].Value;
            probs[i] = cellValue != null ? Convert.ToDouble(cellValue) : 0;
            sum += probs[i];
         }
```

```
        if (Math.Abs(sum - 1.0) > 0.0001)           {
            throw new Exception($"Сума ймовірностей {sum:F2} ≠ 1.0");
        }           return probs;
    }                   private void UpdateProbabilityGrid(int statesCount)
    {
        probabilityDataGrid.ColumnCount = statesCount;
        probabilityDataGrid.RowCount = 1; // Один рядок для ймовірностей
        for (int i = 0; i < statesCount; i++)
        {
            probabilityDataGrid.Columns[i].HeaderText = $"P(α{i + 1})";
            probabilityDataGrid.Columns[i].Width = 80;
            probabilityDataGrid.Columns[i].ValueType = typeof(double);
           /* // Встановлюємо значення за замовчуванням
            probabilityDataGrid.Rows[0].Cells[i].Value = (1.0 /
statesCount).ToString("F2");*/          }       }
    private void calculateButton_Click(object sender, EventArgs e)
    {           try
        {
            var matrix = GetMatrix();
            var alpha = (double)hurwiczAlpha.Value;
            var probs = GetProbabilities(matrix.GetLength(1)); // Використовуємо
таблицю ймовірностей
            var results = new Dictionary<string, double>();
            results["Мінімакс"] = Minimax(matrix);
            results["Байєса-Лапласа"] = BayesLaplace(matrix, probs);
            results["Севіджа"] = Savage(matrix);
            results["Гурвіца"] = Hurwicz(matrix, alpha);
            ShowResults(results);
            DrawChart(results);
        }
```

```
            catch (Exception ex)
            {               MessageBox.Show("Помилка: " + ex.Message);
            }       }
        private double[,] GetMatrix()       {
            int rows = dataGrid.RowCount;
            int cols = dataGrid.ColumnCount;
            double[,] matrix = new double[rows, cols];
            for (int i = 0; i < rows; i++)
                for (int j = 0; j < cols; j++)
matrix[i, j] = double.Parse(dataGrid.Rows[i].Cells[j].Value?.ToString() ?? "0");
            return matrix;
        }       private double Minimax(double[,] matrix)
        {           int rows = matrix.GetLength(0);
            int cols = matrix.GetLength(1);
            double maxOfMins = double.MinValue;
            for (int i = 0; i < rows; i++)
            {               double min = double.MaxValue;
                for (int j = 0; j < cols; j++)
                    min = Math.Min(min, matrix[i, j]);
                maxOfMins = Math.Max(maxOfMins, min);
            }           return maxOfMins;
        }
        private double BayesLaplace(double[,] matrix, double[] probs)
        {           int rows = matrix.GetLength(0);
            int cols = matrix.GetLength(1);
            double bestValue = double.MinValue;
            for (int i = 0; i < rows; i++)
            {               double expected = 0;
                for (int j = 0; j < cols; j++)
                {
```

```
            expected += matrix[i, j] * probs[j];
        }
        if (expected > bestValue)
            bestValue = expected;
    }
    return bestValue;       }
private double Savage(double[,] matrix)       {
    int rows = matrix.GetLength(0);
    int cols = matrix.GetLength(1);
    double[,] regret = new double[rows, cols];
    for (int j = 0; j < cols; j++)
    {               double max = double.MinValue;
        for (int i = 0; i < rows; i++)
            max = Math.Max(max, matrix[i, j]);
        for (int i = 0; i < rows; i++)
            regret[i, j] = max - matrix[i, j];
    }           double minMaxRegret = double.MaxValue;
    for (int i = 0; i < rows; i++)
    {               double maxRegret = double.MinValue;
        for (int j = 0; j < cols; j++)
            maxRegret = Math.Max(maxRegret, regret[i, j]);
        minMaxRegret = Math.Min(minMaxRegret, maxRegret);
    }           return minMaxRegret;
}       private double Hurwicz(double[,] matrix, double alpha)
{           double bestScore = double.MinValue;
    for (int i = 0; i < matrix.GetLength(0); i++)
    {               double min = Enumerable.Range(0, matrix.GetLength(1))
                        .Min(j => matrix[i, j]);
        double max = Enumerable.Range(0, matrix.GetLength(1))
                        .Max(j => matrix[i, j]);
```

```
            // alpha — рівень оптимізму: ближче до 1 = більша вага max
            double score = alpha * max + (1 - alpha) * min;
            if (score > bestScore)
                bestScore = score;          }
        return bestScore;       }
    private void ShowResults(Dictionary<string, double> results)
    {           resultBox.Clear();
        double[,] matrix = GetMatrix();
        double[] probs = GetProbabilities(matrix.GetLength(1));
        double alpha = (double)hurwiczAlpha.Value;
        foreach (var kvp in results)
        {
            string criterion = kvp.Key;
            double criterionValue = kvp.Value;
            List<int> bestAlternatives = new List<int>();
            double bestValue = criterionValue;
            bool isMinimizing = criterion == "Севіджа";
            for (int i = 0; i < matrix.GetLength(0); i++)           {
                double currentValue = CalculateCriterionValue(criterion, matrix, probs,
alpha, i);
                if (Math.Abs(currentValue - bestValue) < 0.001)              {
                    bestAlternatives.Add(i + 1);                }
                else if ((!isMinimizing && currentValue > bestValue) ||
                        (isMinimizing && currentValue < bestValue))
                {                   bestAlternatives.Clear();
                    bestAlternatives.Add(i + 1);
                    bestValue = currentValue;               }
            }
            // Формуємо текст
            string baseText = $"{criterion}: {criterionValue:F2} (";
```

```
            string altText = $"Альтернатива X{bestAlternatives.First()})";
            // Додаємо базовий текст
            resultBox.SelectionStart = resultBox.TextLength;
            resultBox.SelectionColor = Color.Black;
            resultBox.AppendText(baseText);
            // Додаємо альтернативу з кольором
            resultBox.SelectionStart = resultBox.TextLength;
            resultBox.SelectionColor = Color.DarkBlue;
            resultBox.SelectionFont = new DrawFont(resultBox.Font,
DrawingFontStyle.Bold);
            resultBox.AppendText(altText);
            resultBox.AppendText(Environment.NewLine);          }       }
        private double CalculateCriterionValue(string criterion, double[,] matrix,
double[] probs, double alpha, int alternative)       {
            switch (criterion)
            {
                case "Мінімакс":
                    return matrix[alternative, Enumerable.Range(0, matrix.GetLength(1))
                                                .OrderBy(j => matrix[alternative, j])
                                                .First()];
                case "Байєса-Лапласа":
                    return Enumerable.Range(0, matrix.GetLength(1))
                                .Sum(j => matrix[alternative, j] * probs[j]);
                case "Севіджа":
                    double maxRegret = 0;
                    for (int j = 0; j < matrix.GetLength(1); j++)
                    {
                        double bestInState = Enumerable.Range(0, matrix.GetLength(0))
                                                .Max(k => matrix[k, j]);
                        double regret = bestInState - matrix[alternative, j];
```

```
                if (regret > maxRegret) maxRegret = regret;
            }
            return maxRegret;
        case "Гурвіца":
            double min = matrix[alternative, 0];
            double max = matrix[alternative, 0];
            for (int j = 1; j < matrix.GetLength(1); j++)
            {
                if (matrix[alternative, j] < min) min = matrix[alternative, j];
                if (matrix[alternative, j] > max) max = matrix[alternative, j];
            }
            return alpha * max + (1 - alpha) * min;
                default:
            return 0;           }
}
private void DrawChart(Dictionary<string, double> results)
{           chart.Series.Clear();
   chart.ChartAreas[0].RecalculateAxesScale();
   // Прибрати легенду
   chart.Legends.Clear();
   var series = new Series("Критерії")
   {
      ChartType = SeriesChartType.Column,
      IsValueShownAsLabel = true                    };
   chart.Series.Add(series);
   // Налаштування осі X
   chart.ChartAreas[0].AxisX.LabelStyle.Angle = 0;
   chart.ChartAreas[0].AxisX.Interval = 1;
  //  chart.ChartAreas[0].AxisX.Title = "Класичні критерії";
```

```
        chart.ChartAreas[0].AxisX.TitleFont = new DrawFont("Bahnschrift", 12,
DrawingFontStyle.Bold); // Жирний шрифт
        chart.ChartAreas[0].AxisX.TitleForeColor = Color.DarkBlue; // Темно-синій
колір
        chart.ChartAreas[0].AxisX.MajorGrid.Enabled = false;
        chart.ChartAreas[0].AxisX.LabelStyle.Font = new DrawFont("Bahnschrift",
12, DrawingFontStyle.Regular);
        // Кольори для стовпців
        Color[] colors = new Color[]          {
     Color.SteelBlue, Color.Orange, Color.MediumSeaGreen,
     Color.MediumVioletRed, Color.SaddleBrown          };
        // Додавання даних
        int i = 0;
        foreach (var kvp in results)
        {              int index = series.Points.AddXY(kvp.Key, kvp.Value);
          series.Points[index].Color = colors[i % colors.Length];
          i++;
        }      }
    private void saveButton_Click(object sender, EventArgs e)
    {
        SaveFileDialog sfd = new SaveFileDialog();
        sfd.Filter = "Excel Workbook (*.xlsx)|*.xlsx";
        if (sfd.ShowDialog() == DialogResult.OK)
        {
          var wb = new XLWorkbook();
          var ws = wb.Worksheets.Add("Результати");
          // 1. Заголовки таблиці
          for (int i = 0; i < dataGrid.ColumnCount; i++)
          {
            ws.Cell(1, i + 2).Value = dataGrid.Columns[i].HeaderText.ToString();
```

```
            }               // 2. Дані таблиці
            for (int i = 0; i < dataGrid.RowCount; i++)
            {
                ws.Cell(i + 2, 1).Value =
dataGrid.Rows[i].HeaderCell.Value?.ToString();
                for (int j = 0; j < dataGrid.ColumnCount; j++)
                {
                    var cellValue = dataGrid.Rows[i].Cells[j].Value;
                    ws.Cell(i + 2, j + 2).Value = cellValue != null ? cellValue.ToString() :
string.Empty;                 }             }
            // 3. Вивід resultBox
            ws.Cell("A15").Value = "Критерії:";
            var resultLines = resultBox.Text.Split(new[] { Environment.NewLine },
StringSplitOptions.None);
            for (int i = 0; i < resultLines.Length; i++)              {
                ws.Cell(16 + i, 1).Value = resultLines[i];
            }               // 4. Збереження графіка як зображення
            string tempPath = Path.Combine(Path.GetTempPath(), "chart.png");
            chart.SaveImage(tempPath, ChartImageFormat.Png);
            // 5. Додавання зображення до Excel
            var img = ws.AddPicture(tempPath)
                    .MoveTo(ws.Cell("H2"))
                    .Scale(0.75);
            wb.SaveAs(sfd.FileName);
            File.Delete(tempPath);
            MessageBox.Show("Файл успішно збережено!", "Готово",
MessageBoxButtons.OK, MessageBoxIcon.Information);          }       }
        private void clearButton_Click(object sender, EventArgs e)
        {
            // Очистити таблицю
```

```
            dataGrid.Rows.Clear();
            dataGrid.Columns.Clear();
            // Скинути вибрані значення у комбо-боксах
            comboAlternatives.SelectedIndex = -1;
            comboStates.SelectedIndex = -1;
            // Очистити таблицю ймовірностей
            probabilityDataGrid.Rows.Clear();
            probabilityDataGrid.Columns.Clear();
            // Скинути альфа (Гурвіц)
            hurwiczAlpha.Value = 0.5m;
            // Очистити поле результатів
            resultBox.Clear();
            // Очистити діаграму
            chart.Series.Clear();
            chart.Invalidate();        }
        private void dataGrid_EditingControlShowing(object sender,
DataGridViewEditingControlShowingEventArgs e)        {
            TextBox tb = e.Control as TextBox;
            if (tb != null)           {
                tb.TextChanged -= EditingControl_TextChanged;
                tb.TextChanged += EditingControl_TextChanged;           }        }
        private void EditingControl_TextChanged(object sender, EventArgs e)
        {           TextBox tb = sender as TextBox;
            if (tb == null) return;
            string value = tb.Text.Trim();
            if (string.IsNullOrWhiteSpace(value) ||
                !double.TryParse(value, System.Globalization.NumberStyles.Any,
System.Globalization.CultureInfo.InvariantCulture, out _))
            {               tb.BackColor = Color.MistyRose;
            }
```

```
            else            {
                tb.BackColor = Color.White;            }
        }         private void DecisionAnalysis_Load(object sender, EventArgs e)
        {
            ToolTip tip = new ToolTip();
            tip.SetToolTip(comboAlternatives, "Кількість альтернатив (рішень, які можна обрати)");
            tip.SetToolTip(comboStates, "Кількість станів (можливі сценарії)");
            tip.SetToolTip(probabilityDataGrid, "Введіть ймовірності. Сума має бути 1. Наприклад: 0.3,0.4,0.3");
            tip.SetToolTip(hurwiczAlpha, "Коефіцієнт оптимізму для критерію Гурвіца (0 - песимізм, 1 - оптимізм)");
            tip.SetToolTip(dataGrid, "Заповніть таблицю значеннями виграшів для кожної альтернативи і стану.");
            tip.SetToolTip(calculateButton, "Обчислити оптимальні альтернативи за класичними критеріями");        }
        private void buttonDetails_Click(object sender, EventArgs e)
        {            try            {
                string filePath = Path.Combine(Application.StartupPath, "details.docx");
                if (!File.Exists(filePath))
                {                    MessageBox.Show("Файл не знайдено: " + filePath, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }                System.Diagnostics.Process.Start(new System.Diagnostics.ProcessStartInfo()
                {                    FileName = filePath,
                    UseShellExecute = true // Відкриває за замовчуванням через Word або Word Viewer                });            }
            catch (Exception ex)
```

```
        {           MessageBox.Show("Не вдалося відкрити документ: " +
ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);        }
}
    private void buttonAbout_Click(object sender, EventArgs e)
    {        fAbout aboutForm = new fAbout();
      aboutForm.ShowDialog();      }
    private void buttonHelp_Click(object sender, EventArgs e)
    {
      string helpText = @"ЯК ЗАПОВНЮВАТИ ДАНІ?
        1. Матриця виграшів:
          - Кожен рядок - це альтернатива (X1, X2, ...)
          - Кожен стовпець - це стан (α1, α2, ...)
          - Вводити числові значення виграшів
          - Порожні клітинки будуть вважатися нулями
        2. Ймовірності:
          - Вводити числа від 0 до 1
          - Крапка як десятковий роздільник
          - Сума ймовірностей має дорівнювати 1
        3. Коефіцієнт Гурвіца (α):
          - Вводити значення від 0 до 1
          - 0 - повний песимізм
          - 1 - повний оптимізм
          - 0.5 - нейтральний варіант
        Приклад:
          Для 2 альтернатив і 3 станів:
          Матриця:
            10  20  30
            40  15  5
          Ймовірності:
            0.3 0.5 0.2";
```

```
            MessageBox.Show(helpText, "Довідка по формату даних",
MessageBoxButtons.OK, MessageBoxIcon.Information);        }
        private void buttonExample_Click(object sender, EventArgs e)        {
            if (MessageBox.Show("Ця дія перезапише поточні дані. Продовжити?",
"Попередження",
                MessageBoxButtons.YesNo,
                MessageBoxIcon.Warning) == DialogResult.No)            {
                return;            }
            try            {
                // Встановлюємо кількість альтернатив і станів
                comboAlternatives.SelectedItem = 3;
                comboStates.SelectedItem = 4;
                // Даємо час для оновлення інтерфейсу
                Application.DoEvents();
                // Заповнюємо матрицю виграшів (3x4)
                double[,] exampleMatrix = {
                    { 12, -3, 8, 4 },
                    { 7, 10, 5, 9 },
                    { -2, 15, 6, 11 }
                    };
                for (int i = 0; i < 3; i++)                    {
                    for (int j = 0; j < 4; j++)                    {
                        dataGrid.Rows[i].Cells[j].Value = exampleMatrix[i, j].ToString();
                    }                }                // Заповнюємо ймовірності
                double[] exampleProbs = { 0.15, 0.35, 0.25, 0.25 };
                for (int j = 0; j < 4; j++)                {
                    probabilityDataGrid.Rows[0].Cells[j].Value =
exampleProbs[j].ToString("F2");
                }                // Встановлюємо коефіцієнт Гурвіца
                hurwiczAlpha.Value = 0.6m;
```

```
            MessageBox.Show("Тестові дані завантажено!\nТепер можете
натиснути \"Обчислити\" для аналізу.",
                    "Приклад завантажено",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information);            }
            catch (Exception ex)            {
                MessageBox.Show("Помилка при завантаженні прикладу: " +
ex.Message,
                    "Помилка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
            }       }
        private void printToPrinterMenuItem_Click(object sender, EventArgs e)
        {           try         {
            // Перевірка, чи є дані для друку
            if (dataGrid.Rows.Count == 0 || dataGrid.Columns.Count == 0)
            {       MessageBox.Show("Немає даних для друку!", "Попередження",
                        MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;             }
            // Ініціалізація printDialog, якщо ще не ініціалізований
            if (printDialog == null)
            {               printDialog = new PrintDialog();
                printDialog.Document = printDocument ?? new PrintDocument();
            }
            if (printDialog.ShowDialog() == DialogResult.OK)
            {               // Додаємо підтвердження перед друком
                if (MessageBox.Show("Ви хочете надрукувати результати?",
"Підтвердження друку",
                        MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
DialogResult.Yes)               {
```

```
                printDocument.Print();                }            }          }
        catch (Exception ex)          {
            MessageBox.Show($"Помилка при друку:\n{ex.Message}", "Помилка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);          }       }
    private void printToPDFMenuItem_Click(object sender, EventArgs e)
    {          // Перевірка наявності даних          if (dataGrid.Rows.Count == 0 ||
dataGrid.Columns.Count == 0)
        {          MessageBox.Show("Немає даних для експорту!", "Попередження",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;          }
        using (SaveFileDialog saveFileDialog = new SaveFileDialog())
        {
            saveFileDialog.Filter = "PDF файли (*.pdf)|*.pdf";
            saveFileDialog.Title = "Зберегти результати як PDF";
            saveFileDialog.FileName =
$"Результати_аналізу_{DateTime.Now:yyyyMMdd_HHmmss}";
            if (saveFileDialog.ShowDialog() == DialogResult.OK)              {
                try                {
                    // Створюємо PDF документ
             using (Document pdfDoc = new Document(PageSize.A4, 40f, 40f, 60f, 60f))
        using (FileStream stream = new FileStream(saveFileDialog.FileName,
FileMode.Create))                {
                    PdfWriter writer = PdfWriter.GetInstance(pdfDoc, stream);
                    pdfDoc.Open();
                    // Додаємо заголовок
                    iTextSharp.text.Font titleFont = FontFactory.GetFont("Arial", 18,
iTextSharp.text.BaseColor.BLUE);
                    Paragraph title = new Paragraph("Результати аналізу рішень\n\n",
titleFont)
                    {    Alignment = Element.ALIGN_CENTER                };
```

```
                pdfDoc.Add(title);
                // Додаємо дату
                iTextSharp.text.Font dateFont = FontFactory.GetFont("Arial", 10,
iTextSharp.text.BaseColor.BLACK);
                pdfDoc.Add(new Paragraph($"Дата: {DateTime.Now:dd.MM.yyyy
HH:mm}", dateFont));
                pdfDoc.Add(Chunk.NEWLINE);
                // Додаємо матрицю виграшів
                AddMatrixToPdf(pdfDoc);
                // Додаємо ймовірності (якщо є)
                if (probabilityDataGrid.ColumnCount > 0)
                {
                    AddProbabilitiesToPdf(pdfDoc);                    }
                // Додаємо результати (якщо є)
                if (!string.IsNullOrWhiteSpace(resultBox.Text))                    {
                    AddResultsToPdf(pdfDoc);                    }                }
                MessageBox.Show($"Файл PDF успішно
збережено:\n{saveFileDialog.FileName}",
                    "Готово",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information);                }
            catch (Exception ex)                {
                MessageBox.Show($"Помилка при створенні PDF:\n{ex.Message}",
                    "Помилка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);            }            }        }        }
    private void AddMatrixToPdf(Document pdfDoc)
    {
        iTextSharp.text.Font tableHeaderFont = FontFactory.GetFont("Arial", 12,
iTextSharp.text.BaseColor.BLACK);
```

```
            pdfDoc.Add(new Paragraph("Матриця виграшів:", tableHeaderFont));
            PdfPTable matrixTable = new PdfPTable(dataGrid.ColumnCount + 1)
            {
               WidthPercentage = 100          };
            // Заголовки стовпців
            matrixTable.AddCell(new Phrase("Альт./Стан", tableHeaderFont));
            for (int j = 0; j < dataGrid.ColumnCount; j++)          {
               matrixTable.AddCell(new Phrase(dataGrid.Columns[j].HeaderText,
tableHeaderFont));          }
            // Дані матриці
            iTextSharp.text.Font cellFont = FontFactory.GetFont("Arial", 10,
iTextSharp.text.BaseColor.BLACK);
            for (int i = 0; i < dataGrid.RowCount; i++)          {
               matrixTable.AddCell(new
Phrase(dataGrid.Rows[i].HeaderCell.Value?.ToString() ?? $"X{i + 1}", cellFont));
               for (int j = 0; j < dataGrid.ColumnCount; j++)              {
                  matrixTable.AddCell(new
Phrase(dataGrid.Rows[i].Cells[j].Value?.ToString() ?? "0", cellFont));
               }          }
            pdfDoc.Add(matrixTable);
            pdfDoc.Add(Chunk.NEWLINE);       }
         private void AddProbabilitiesToPdf(Document pdfDoc)
         {   iTextSharp.text.Font tableHeaderFont = FontFactory.GetFont("Arial", 12,
iTextSharp.text.BaseColor.BLACK);
            iTextSharp.text.Font cellFont = FontFactory.GetFont("Arial", 10,
iTextSharp.text.BaseColor.BLACK);
            pdfDoc.Add(new Paragraph("Ймовірності станів:", tableHeaderFont));
            PdfPTable probTable = new PdfPTable(probabilityDataGrid.ColumnCount)
            {              WidthPercentage = 100          };
            for (int j = 0; j < probabilityDataGrid.ColumnCount; j++)
```

```
        {               probTable.AddCell(new
Phrase(probabilityDataGrid.Columns[j].HeaderText, tableHeaderFont));          }
        for (int j = 0; j < probabilityDataGrid.ColumnCount; j++)          {
            probTable.AddCell(new
Phrase(probabilityDataGrid.Rows[0].Cells[j].Value?.ToString() ?? "0", cellFont));
        }                   pdfDoc.Add(probTable);
        pdfDoc.Add(Chunk.NEWLINE);       }
    private void AddResultsToPdf(Document pdfDoc)       {
        iTextSharp.text.Font tableHeaderFont = FontFactory.GetFont("Arial", 12,
iTextSharp.text.BaseColor.BLACK);
        iTextSharp.text.Font resultFont = FontFactory.GetFont("Arial", 11,
iTextSharp.text.BaseColor.BLACK);
        pdfDoc.Add(new Paragraph("Результати аналізу:", tableHeaderFont));
        string[] resultLines = resultBox.Text.Split(new[] { Environment.NewLine },
StringSplitOptions.RemoveEmptyEntries);
        foreach (string line in resultLines)
        {               pdfDoc.Add(new Paragraph(line, resultFont));
        }       }

    private void printDocument_PrintPage(object sender, PrintPageEventArgs e)
    {           try
        {               Graphics g = e.Graphics;
            using (DrawFont titleFont = new DrawFont("Arial", 18,
DrawingFontStyle.Bold))
            using (DrawFont headerFont = new DrawFont("Arial", 12,
DrawingFontStyle.Bold))
            using (DrawFont normalFont = new DrawFont("Arial", 10))
            {                   Brush brush = Brushes.Black;
                float yPos = 50;
                float leftMargin = e.MarginBounds.Left;
```

```
                // Заголовок
                g.DrawString("Результати аналізу рішень", titleFont, brush,
leftMargin, yPos);
                yPos += titleFont.GetHeight() + 20;
                // Дата
                g.DrawString($"Дата: {DateTime.Now:dd.MM.yyyy HH:mm}",
normalFont, brush, leftMargin, yPos);
                yPos += normalFont.GetHeight() + 15;              // Матриця виграшів
                DrawMatrix(g, ref yPos, leftMargin, headerFont, normalFont, brush);
                // Ймовірності (якщо є)
                if (probabilityDataGrid.ColumnCount > 0)                  {
                   DrawProbabilities(g, ref yPos, leftMargin, headerFont, normalFont,
brush);                  }
                // Результати (якщо є)
                if (!string.IsNullOrWhiteSpace(resultBox.Text))                  {
                   DrawResults(g, ref yPos, leftMargin, headerFont, normalFont, brush);
                }              }
             e.HasMorePages = false;           }
          catch (Exception ex)          {
             MessageBox.Show($"Помилка при підготовці до
друку:\n{ex.Message}",
                             "Помилка",
                             MessageBoxButtons.OK,
                             MessageBoxIcon.Error);          }       }
       private void DrawMatrix(Graphics g, ref float yPos, float leftMargin, DrawFont
headerFont, DrawFont normalFont, Brush brush)
       {  g.DrawString("Матриця виграшів:", headerFont, brush, leftMargin, yPos);
          yPos += headerFont.GetHeight() + 10;
          float xPos = leftMargin;
          float cellWidth = 80;
```

```
            float cellHeight = normalFont.GetHeight() + 5;
            g.DrawString("Альт./Стан", headerFont, brush, xPos, yPos);
            xPos += cellWidth;
            for (int j = 0; j < dataGrid.ColumnCount; j++)
            { g.DrawString(dataGrid.Columns[j].HeaderText, headerFont, brush, xPos,
yPos);
                xPos += cellWidth;          }
            yPos += cellHeight;
            for (int i = 0; i < dataGrid.RowCount; i++)          {
                xPos = leftMargin;
                g.DrawString(dataGrid.Rows[i].HeaderCell.Value?.ToString() ?? $"X{i +
1}", normalFont, brush, xPos, yPos);
                xPos += cellWidth;
                for (int j = 0; j < dataGrid.ColumnCount; j++)              {
                    g.DrawString(dataGrid.Rows[i].Cells[j].Value?.ToString() ?? "0",
normalFont, brush, xPos, yPos);
                    xPos += cellWidth;              }
                yPos += cellHeight;          }
            yPos += 20;
        }
        private void DrawProbabilities(Graphics g, ref float yPos, float leftMargin,
DrawFont headerFont, DrawFont normalFont, Brush brush)
        {
            g.DrawString("Ймовірності станів:", headerFont, brush, leftMargin, yPos);
            yPos += headerFont.GetHeight() + 10;
            float xPos = leftMargin;
            float cellWidth = 80;
            float cellHeight = normalFont.GetHeight() + 5;
            for (int j = 0; j < probabilityDataGrid.ColumnCount; j++)          {
```

```
            g.DrawString(probabilityDataGrid.Columns[j].HeaderText, headerFont,
brush, xPos, yPos);
            xPos += cellWidth;          }
        yPos += cellHeight;
        xPos = leftMargin;
        for (int j = 0; j < probabilityDataGrid.ColumnCount; j++)          {
            g.DrawString(probabilityDataGrid.Rows[0].Cells[j].Value?.ToString() ??
"0", normalFont, brush, xPos, yPos);
            xPos += cellWidth;          }
        yPos += cellHeight + 20;       }
    private void DrawResults(Graphics g, ref float yPos, float leftMargin, DrawFont
headerFont, DrawFont normalFont, Brush brush)
    {
        g.DrawString("Результати аналізу:", headerFont, brush, leftMargin, yPos);
        yPos += headerFont.GetHeight() + 10;
        string[] resultLines = resultBox.Text.Split(new[] { Environment.NewLine },
StringSplitOptions.RemoveEmptyEntries);
        foreach (string line in resultLines)
        {
            g.DrawString(line, normalFont, brush, leftMargin, yPos);
            yPos += normalFont.GetHeight() + 5;
        }
    }
    private void buttonPrint_Click(object sender, EventArgs e)
    {
        // Дія за замовчуванням - друк на принтер
        printToPrinterMenuItem_Click(sender, e);       }    }}
```

Лістинг fAbout.cs

```
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace DecisionAnalysisApp
{
   public partial class fAbout : Form
   {
      public fAbout()
      {
         InitializeComponent();
      }
      private void btnClose_Click(object sender, EventArgs e)
      {
         this.DialogResult = DialogResult.OK;
         this.Close();       }    }  }
```