Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська дипломна робота на тему:

ПРОГРАМНИЙ МОДУЛЬ ДІАГНОСТИКИ ФІНАНСОВОГО СТАНУ ПІДПРИЄМСТВА

Виконав: студент 4 курсу,групи 1КН-20Б
спеціальності 122 Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Гвардій В.М.
(прізвище та ініціали)

Керівник: к.ф-м.н., доцент каф. КН
Шевчук О.Ф.
(прізвище та ініціали)
«07» 06 2024 р.

Рецензент: к.т.н., доцент каф.САІТ
Варчук І. В.
(прізвище та ініціали)
«07» 06 2024 р.

Допущено до захисту
Зав. кафедри д. т. н. Яровий А.А.
«07» 06 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А. А.
«07» 06 2024 р

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Гвардію Валентину Михайловичу

1. Тема роботи: Програмний модуль діагностики фінансового стану підприємства.
Керівник роботи: доцент Шевчук Олександр Федорович.
Затверджені наказом вищого навчального закладу "11" 03 2024 року №__

2. Термін подання студентом роботи_____ 27.05.2024

3. Вихідні дані до роботи:
Мова програмування - об'єктно-орієнтована;
Кількість параметрів - не менше 5;
Мінімалістичний GUI;
Можливість відвантаження звітів.

4. Зміст текстової частини (перелік питань, які потрібно розробити):
вступ; аналіз сучасних засобів діагностики фінансового стану підприємства; проектування програмного модуля діагностики фінансового стану підприємства; розробка програмного модуля діагностики фінансового стану підприємства; висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):
Структура програмного модуля діагностики фінансового стану підприємства; алгоритм функціонування системи; скріншоти тестування програмного модуля.

6. загальний вигляд інтерфейсу користувачаКонсультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Шевчук., к.ф-м.н., доц. каф. КН		

7. Дата видачі завдання 07.03.2024

Календарний план

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз сучасних засобів діагностики фінансового стану підприємства	06.05.2024	07.05.2024	
2	Проектування програмного модуля діагностики фінансового стану підприємства	08.05.2024	09.05.2024	
3	Розробка програмного модуля діагностики фінансового стану підприємства	13.05.2024	14.05.2024	
4	Розробка інструкції користувача	15.05.2024	26.05.2024	
5	Аналіз отриманих результатів та формування висновків	26.05.2024	27.05.2024	
6	Оформлення бакалаврської дипломної роботи	04.06.2024	06.06.2024	

Студент _____ **Гвардій В.М.**
(підпис) *(прізвище та ініціали)*

Керівник роботи _____ **Шевчук О.Ф.**
(підпис) *(прізвище та ініціали)*

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 74 сторінок формату А4, на яких є 9 рисунків, список використаних джерел містить 15 найменувань.

Бакалаврська робота має на меті розробку комплексного програмного модуля для фінансового аналізу та підтримки прийняття рішень в управлінні ресурсами підприємства. Дослідження включає теоретичний огляд існуючих підходів до фінансового аналізу, алгоритми для розрахунку ключових фінансових коефіцієнтів та показників, автоматизацію збору та обробки даних, а також методики візуалізації результатів для їх інтуїтивної інтерпретації. Робота зосереджується на вирішенні завдань оцінки ліквідності, фінансової стійкості, ділової активності, рентабельності та ефективності використання ресурсів. Запропоноване рішення спрощує процес фінансового аналізу, сприяючи прийняттю рішень на основі даних та покращуючи загальне управління фінансами підприємства.

Ключові слова: фінансовий аналіз, управління ресурсами підприємства, фінансові коефіцієнти, візуалізація даних, система підтримки прийняття рішень, ліквідність, рентабельність, використання ресурсів.

ABSTRACT

The bachelor's thesis consists of 74 pages in A4 format, which include 9 figures. The list of references contains 15 sources.

The bachelor's thesis aims to develop a comprehensive software module for financial analysis and decision support in enterprise resource management. It involves a theoretical review of existing approaches to financial analysis, algorithms for calculating key financial ratios and indicators, automation of data collection and processing, and visualization techniques for intuitive interpretation of results. The research addresses the challenges of assessing liquidity, financial stability, business activity, profitability, and resource utilization efficiency. The proposed solution streamlines the financial analysis process, enabling data-driven decision-making and enhancing the overall financial management of enterprises.

Keywords: financial analysis, enterprise resource management, financial ratios, data visualization, decision support system, liquidity, profitability, resource utilization.

ЗМІСТ

ВСТУП

Актуальність теми фінансовий стан підприємства є одним з ключових показників його діяльності та стабільності. Від нього залежить здатність компанії залучати інвестиції, кредитні ресурси, розраховуватись за зобов'язаннями, забезпечувати безперебійне функціонування операційних процесів. Своєчасна та об'єктивна оцінка фінансового стану дозволяє керівництву приймати виважені управлінські рішення, попереджувати кризові ситуації та забезпечувати сталий розвиток бізнесу.

Діагностика фінансового стану традиційно проводиться шляхом розрахунку низки фінансових коефіцієнтів та показників за даними бухгалтерської звітності підприємства. Однак, така оцінка потребує значних витрат людських ресурсів, часу та є схильною до помилок через велику кількість обчислень.

Для вирішення цих проблем актуальним є створення програмного модуля, який би автоматизував процес діагностики фінансового стану, збільшив його точність та оперативність. Використання спеціалізованого програмного забезпечення дозволяє стандартизувати розрахунки, уніфікувати інтерпретацію результатів, а також надавати рекомендації щодо покращення фінансових показників.

Застосування автоматизованого програмного модуля для діагностики фінансового стану підприємства відкриває нові можливості для ефективного управління бізнесом. Основними перевагами такого рішення є швидкість та оперативність аналізу, підвищена точність розрахунків, виявлення прихованих закономірностей та тенденцій, а також стандартизація процесу оцінки фінансових показників.

Крім того, програмний модуль забезпечує комплексний підхід до аналізу фінансового стану, враховуючи різноманітні коефіцієнти та індикатори, такі як показники ліквідності, платоспроможності, ділової активності, рентабельності та інші. Це дозволяє отримати всебічну картину фінансового

здоров'я підприємства та виявити потенційні ризики чи можливості для покращення.

Важливою функцією автоматизованого модуля є можливість генерувати звіти та рекомендації на основі результатів аналізу. Використовуючи накопичену базу знань та кращі практики, програмне забезпечення може надавати керівництву компанії конкретні поради щодо оптимізації фінансових потоків, підвищення ефективності використання ресурсів, управління заборгованістю та інших заходів для зміцнення фінансової стійкості підприємства.

Метою дослідження є підвищення якості процесу автоматизованої оцінки фінансового стану підприємства. Виявлення потенційних ризиків та можливостей для вдосконалення фінансової діяльності, підвищення ефективності прийняття управлінських рішень та забезпечення стійкого розвитку підприємства.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1. Провести огляд теоретичних основ та існуючих підходів до аналізу фінансового стану підприємства, виявити їх переваги та недоліки.
2. Розробити алгоритми та методики для розрахунку ключових фінансових коефіцієнтів та показників, що характеризують ліквідність, фінансову стійкість, ділову активність, рентабельність та ефективність використання ресурсів підприємства.
3. Спроектувати та реалізувати програмний модуль, що забезпечить автоматизацію процесу збору, обробки та аналізу фінансових даних підприємства.
4. Розробити методику візуалізації результатів аналізу фінансового стану у вигляді графіків, діаграм та таблиць для полегшення інтерпретації та прийняття рішень.
5. Провести тестування розробленого програмного модуля.

Об'єктом дослідження є процеси діагностики, оцінки та аналізу фінансового стану підприємства.

Предметом дослідження є методи, алгоритми та інструменти комплексного аналізу фінансових показників.

Очікувані результати дослідження полягають у створенні програмного модуля, який забезпечить автоматизацію процесу збору та обробки фінансових даних, проведення комплексного аналізу фінансових показників, візуалізацію результатів та надання рекомендацій для підтримки фінансового здоров'я підприємства.

1 АНАЛІЗ СУЧАСНИХ ЗАСОБІВ ДІАГНОСТИКИ ФІНАНСОВОГО СТАНУ ПІДПРИЄМСТВА

1.1 Аналіз предметної області

Фінансовий стан підприємства відображає здатність суб'єкта господарювання фінансувати свою діяльність. Він характеризується забезпеченістю фінансовими ресурсами, раціональним їх розміщенням та ефективним використанням, фінансовими взаєминами з іншими юридичними та фізичними особами, платоспроможністю та фінансовою стійкістю.

Оцінка фінансового стану базується на даних бухгалтерської звітності підприємства, зокрема Балансу (Звіту про фінансовий стан) та Звіту про фінансові результати (Звіту про сукупний дохід). На основі показників цих форм розраховується низка фінансових коефіцієнтів та індикаторів [1-3].

Основні групи показників, що аналізуються:

– Показники ліквідності та платоспроможності (коефіцієнти абсолютної, проміжної та загальної ліквідності, чистий оборотний капітал тощо). Вони характеризують здатність підприємства своєчасно розраховуватися за поточними зобов'язаннями.

– Показники фінансової стійкості (коефіцієнти фінансової незалежності, фінансування, забезпеченості власними оборотними коштами тощо). Відображають ступінь залежності підприємства від зовнішніх джерел фінансування.

– Показники ділової активності (коефіцієнти оборотності активів, дебіторської та кредиторської заборгованості, операційного та фінансового циклів). Характеризують ефективність використання ресурсів підприємства.

– Показники рентабельності (коефіцієнти рентабельності активів, власного капіталу, продажів). Відображають прибутковість діяльності підприємства.

Інтерпретація значень наведених вище показників потребує залучення експертних знань, врахування галузевих особливостей та динаміки зміни коефіцієнтів у часі. Результати діагностики дозволяють виявити потенційні проблеми у фінансовій сфері підприємства та обґрунтувати управлінські рішення.

Впровадження програмного модуля для оцінки фінансового стану підприємства передбачає кілька ключових аспектів. По-перше, необхідно забезпечити відповідність функціоналу додатку вимогам та специфіці конкретного підприємства, галузі, в якій воно працює, масштабам діяльності тощо. Це дозволить отримувати максимально релевантні та корисні результати аналізу фінансових показників [4-5].

По-друге, важливим є інтеграція програмного модуля з існуючими на підприємстві інформаційними системами та базами даних, зокрема системами бухгалтерського обліку та звітності. Це забезпечить автоматичний обмін даними, необхідними для проведення діагностики фінансового стану, і значно спростить процес введення вхідної інформації.

Крім того, необхідно приділити увагу навчанню персоналу роботі з новим програмним забезпеченням, а також розробити чіткі інструкції та рекомендації щодо інтерпретації результатів аналізу та прийняття відповідних управлінських рішень на їх основі.

Успішне впровадження програмного модуля для діагностики фінансового стану відкриває ряд перспектив для підприємства. По-перше, це дозволить значно підвищити оперативність та точність оцінки фінансового стану, що є критично важливим для своєчасного виявлення потенційних ризиків та проблемних аспектів діяльності [6].

Регулярний моніторинг фінансових показників за допомогою програмного модуля дасть змогу відстежувати динаміку змін та тренди, що сприятиме прийняттю більш обґрунтованих та ефективних управлінських рішень.

1.2 Огляд та порівняльний аналіз програмних аналогів оцінки фінансового стану підприємства

На ринку представлено кілька програмних рішень для аналізу фінансового стану компаній, таких як FiscalNote, Фінансовий сканер, FineXtra та інші. Більшість із них пропонують розрахунок основних фінансових коефіцієнтів та генерацію звітів [7-9]. Однак, запропонований програмний модуль діагностики фінансового стану відрізняється більш широким функціоналом та використанням передових технологій машинного навчання та аналізу даних. Зокрема, він здатний виявляти приховані тенденції, автоматично надавати рекомендації щодо покращення показників, інтегруватися з корпоративними системами для безперервного моніторингу. Модуль також забезпечує високу швидкість та точність розрахунків, а також стандартизацію процесу оцінки.

Розглянемо детальніше їх особливості, сильні та слабкі сторони.

FiscalNote: Ця програма фокусується на всебічному аналізі фінансових показників та генерації звітів. Вона пропонує розрахунок широкого спектру коефіцієнтів, включаючи показники ліквідності, рентабельності, ділової активності та фінансової стійкості. Одна з сильних сторін FiscalNote – зручний інтерфейс та наочна візуалізація даних у вигляді графіків та діаграм.

Однак, програма має кілька недоліків. По-перше, вона не здатна виявляти приховані тенденції та взаємозв'язки між показниками, покладаючись лише на стандартні розрахунки. По-друге, FiscalNote не надає автоматичних рекомендацій щодо покращення фінансового стану, залишаючи інтерпретацію результатів на користувача. Крім того, відсутня можливість інтеграції з корпоративними системами для безперервного моніторингу.

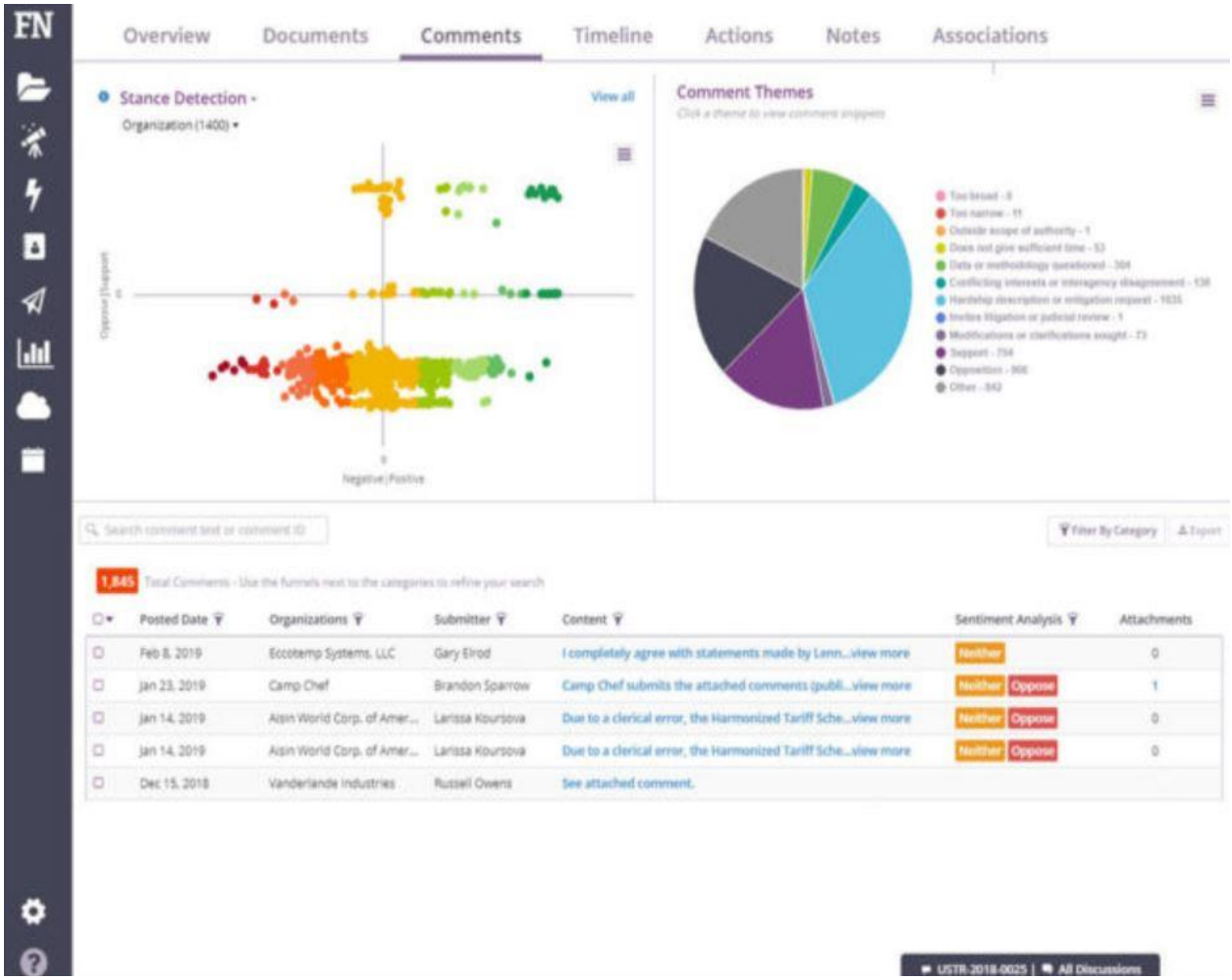

Рисунок 1.1 – Зовнішній вигляд FiscalNote

Фінансовий сканер: Це програмне забезпечення зосереджене на аналізі фінансової звітності підприємств. Воно дозволяє імпортувати дані з різних джерел та автоматизує розрахунок основних фінансових коефіцієнтів. Перевагою Фінансового сканера є простота використання та зрозумілий інтерфейс.

Однак, як і в попередньому випадку, програма обмежується лише стандартними розрахунками та не здатна виявляти приховані закономірності чи надавати рекомендації. Відсутня також можливість інтеграції з корпоративними системами та безперервного моніторингу показників.

FineXtra: Ця програма є більш комплексним рішенням для аналізу фінансового стану підприємств. Крім розрахунку коефіцієнтів та генерації

звітів, FineXtra пропонує модулі для прогнозування фінансових показників, оцінки ризиків та порівняння з галузевими стандартами.

Одна з сильних сторін FineXtra – можливість інтеграції з популярними ERP-системами, такими як SAP чи Oracle. Це дозволяє отримувати актуальні дані для аналізу безпосередньо з корпоративних систем. Крім того, програма пропонує гнучке налаштування звітів та можливість застосовувати індивідуальні формули для розрахунку показників.

Однак, FineXtra не використовує передові технології машинного навчання чи аналізу даних для виявлення прихованих тенденцій. Також відсутня функція автоматичного надання рекомендацій щодо покращення фінансових показників.

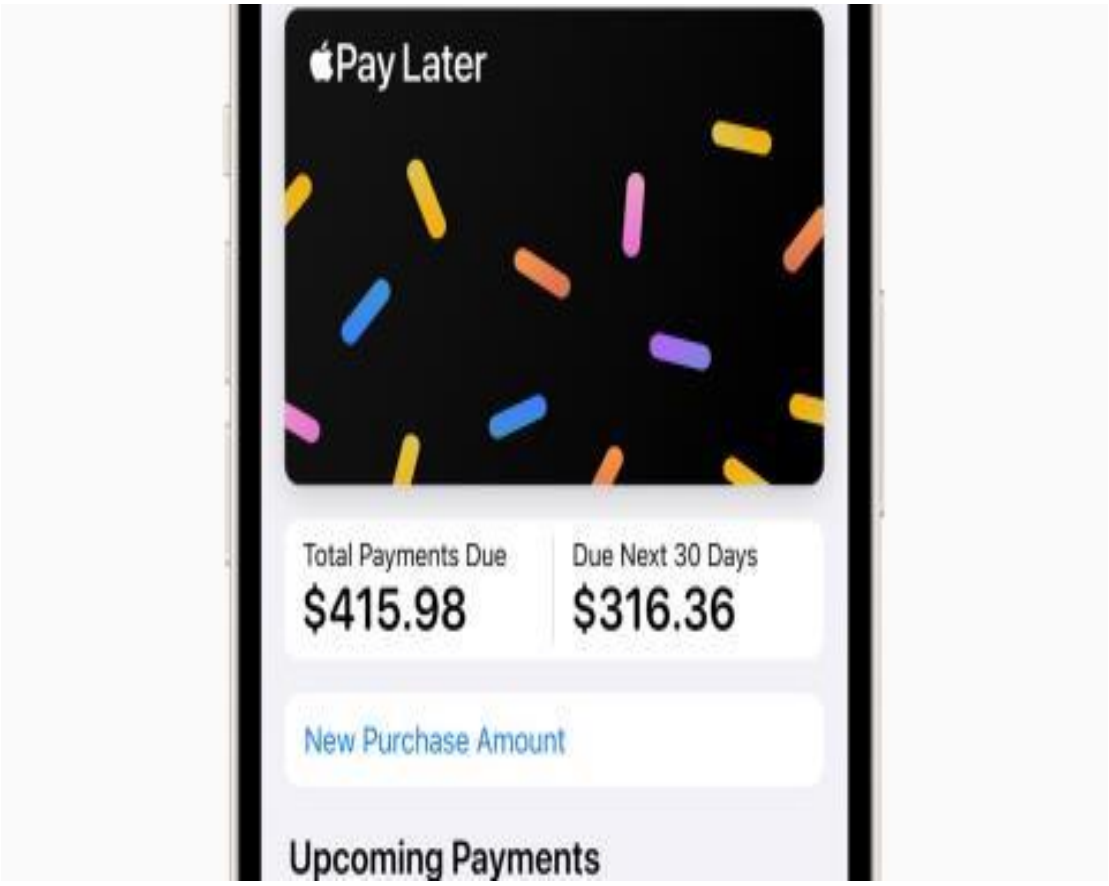

Рисунок 1.2 – Зовнішній вигляд FineXtra

Таблиця 1.1 – Порівняльний аналіз програм-аналогів

Програма	**Особливості**	**Переваги**	**Недоліки**
FiscalNote	Розрахунок широкого спектру коефіцієнтів Наочна візуалізація даних	Зручний інтерфейс Генерація звітів	Відсутність виявлення прихованих тенденцій Немає автоматичних рекомендацій Відсутня інтеграція з корпоративними системами
Фінансовий сканер	Імпорт даних з різних джерел Автоматизований розрахунок основних коефіцієнтів	Простота використання Зрозумілий інтерфейс	Обмежений стандартними розрахунками Немає виявлення закономірностей Відсутня інтеграція та безперервний моніторинг
FineXtra	Аналіз фінансової звітності Прогнозування показників Оцінка ризиків Порівняння з галузевими стандартами	Інтеграція з ERP-системами Гнучке налаштування звітів Можливість застосовувати індивідуальні формули	Відсутність використання машинного навчання Немає автоматичних рекомендацій

1.3 Постановка задачі

Сформуємо постановку задачі на розробку програмного модуля фінансової діагностики підприємств:

– Провести огляд існуючих методів та підходів до аналізу фінансового стану підприємства, виявити їх переваги та недоліки.

– Розробити алгоритми та методики для розрахунку ключових фінансових коефіцієнтів та показників, що характеризують ліквідність, фінансову стійкість, ділову активність, рентабельність та ефективність використання ресурсів підприємства.

– Спроектувати та реалізувати програмний модуль, який забезпечить автоматизацію процесу збору, обробки та аналізу фінансових даних підприємства.

– Розробити методику візуалізації результатів аналізу фінансового стану у вигляді графіків, діаграм та таблиць для полегшення інтерпретації та прийняття рішень.

– Сформувати систему рекомендацій щодо покращення фінансового стану підприємства на основі результатів проведеного аналізу.

– Провести апробацію розробленого програмного модуля на реальних даних підприємства та оцінити його ефективність і практичну цінність.

– Розробити методичні рекомендації щодо використання програмного модуля для діагностики фінансового стану підприємства.

Зведемо функціональні вимоги до системи в таблицю 1.2.

Таблиця 1.2 – Функціональні вимоги на розробку ПЗ

Вимога	**Опис**
1	2
Імпорт фінансових даних	Можливість імпортувати фінансові дані підприємства з різних джерел (Excel, CSV, бази даних тощо).
Розрахунок коефіцієнтів	Автоматичний розрахунок ключових фінансових коефіцієнтів та показників, таких як коефіцієнти ліквідності, фінансової стійкості, ділової активності, рентабельності.
Візуалізація результатів	Візуалізація результатів аналізу у вигляді графіків, діаграм та таблиць для полегшення інтерпретації.
Генерація звітів	Можливість генерувати звіти з детальним аналізом фінансового стану підприємства.
Рекомендації	Надання рекомендацій щодо покращення фінансового стану підприємства на основі результатів аналізу.
Порівняння з нормативами	Порівняння розрахованих коефіцієнтів із нормативними значеннями для відповідної галузі.
Аналіз тенденцій	Можливість аналізувати тенденції зміни фінансових показників у часі.
Користувацький інтерфейс	Зручний та інтуїтивно зрозумілий користувацький інтерфейс для взаємодії з програмним модулем.
Експорт даних	Можливість експортувати результати аналізу у різні формати (Excel, PDF тощо).

1.4 Висновки до розділу 1

Аналіз фінансового стану підприємства відіграє критично важливу роль для оцінки його фінансової спроможності, платоспроможності, ефективності використання ресурсів та прибутковості діяльності. Цей аналіз здійснюється на основі розрахунку низки фінансових показників та коефіцієнтів з використанням даних бухгалтерської звітності підприємства. Для забезпечення релевантності та корисності результатів аналізу важливо враховувати специфіку діяльності підприємства, його галузеву приналежність та масштаби діяльності.

На ринку представлено кілька програмних рішень для аналізу фінансового стану підприємств, таких як FiscalNote, Фінансовий сканер та FineXtra. Проте ці системи-аналоги мають певні обмеження та недоліки. Зокрема, вони здебільшого обмежуються лише стандартними розрахунками коефіцієнтів, не здатні виявляти приховані тенденції та взаємозв'язки між показниками, не надають автоматичних рекомендацій щодо покращення фінансового стану. Крім того, більшість з них не підтримують інтеграцію з корпоративними інформаційними системами для безперервного моніторингу фінансових показників.

Також сформовано постановку задачі і висвітлено функціональні вимоги на розробку.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ ДІАГНОСТИКИ ФІНАНСОВОГО СТАНУ ПІДПРИЄМСТВА

2.1 Розробка структури програмного модуля

Структуру програмного модуля можна умовно розділити на кілька основних компонентів:

1. Модуль введення даних. Цей компонент відповідає за забезпечення зручного та гнучкого інтерфейсу для введення вхідних даних, необхідних для проведення аналізу фінансового стану. Зазвичай це дані бухгалтерської звітності підприємства (Баланс, Звіт про фінансові результати та ін.). Модуль повинен дозволяти завантаження даних з різних джерел (файлів, баз даних, інтеграцію з існуючими системами обліку тощо) та забезпечувати перевірку та валідацію введених даних.

2. Модуль розрахунку фінансових коефіцієнтів. Ядро програмного модуля, яке реалізує алгоритми розрахунку різноманітних фінансових показників та коефіцієнтів на основі введених даних. Цей компонент повинен бути гнучким та масштабованим, дозволяючи додавання нових показників та методик аналізу у майбутньому.

3. Модуль інтерпретації результатів. Цей компонент відповідає за аналіз розрахованих фінансових коефіцієнтів, виявлення відхилень від нормативних значень або галузевих стандартів, а також формування рекомендацій щодо покращення фінансового стану підприємства. Він повинен містити базу знань з експертними правилами та алгоритмами прийняття рішень.

4. Модуль візуалізації та звітності. Забезпечує зручне відображення результатів діагностики у вигляді графіків, діаграм, таблиць та формування комплексних звітів для подальшого аналізу та прийняття управлінських рішень.

5. Модуль адміністрування та налаштувань. Дозволяє налаштовувати параметри роботи програмного модуля, управляти користувачами та їх правами доступу, оновлювати базу знань та налаштовувати методики аналізу відповідно до специфіки підприємства.

Даний програмний модуль планується як веб-додаток для візуалізації та аналізу взаємозв'язку між монетарною політикою та розподілом фінансів на прикладі датасету з США. Він буде розроблений з використанням фреймворку Dash для Python, який дозволяє створювати інтерактивні веб-додатки на основі даних.

Перш за все, модуль буде імпортувати необхідні бібліотеки та модулі, такі як Pandas для обробки даних, Plotly для візуалізації даних, NumPy для числових обчислень та Dash для створення веб-додатку. Ці бібліотеки забезпечать необхідну функціональність для роботи з даними, візуалізації та створення користувацького інтерфейсу.

Далі будуть реалізовані функції для візуалізації даних. Ключовою функцією буде plot_wealth, яка візуалізуватиме розподіл Капіталу за процентилями населення, використовуючи дані про частку загального чистого капіталу та загально чистого капіталу. Ця функція створюватиме складені графіки для відображення різних аспектів розподілу Капіталу, таких як частка Капіталу за процентилями, зміна загального чистого Капіталу, ефективна ставка фонду ФРС та балансовий рахунок. Інша функція, plot_economy, візуалізуватиме стан фінансової та реальної економіки США, відображаючи показники, такі як реальний ВВП, індекс споживчих настроїв, промислове виробництво, купівельна спроможність долара США, S&P 500 та медіанна ціна продажу будинків.

Модуль завантажуватиме дані з CSV-файлів для трьох наборів даних: частки Капіталу, чистого Капіталу та економічних показників. Ці дані будуть перетворені на об'єкти DataFrame бібліотеки Pandas для полегшення маніпуляцій і візуалізації.

Далі буде налаштовано веб-інтерфейс додатку за допомогою фреймворку Dash. Інтерфейс складатиметься з заголовка, повзунка для вибору діапазону дат, графіків візуалізації даних про розподіл Капіталу, фінансову та реальну економіку. Повзунок діапазону дат дозволить користувачам вибирати певний період часу для аналізу та візуалізації відповідних даних.

Для обробки користувацького вводу буде реалізована функція зворотного виклику, яка оновлюватиме візуалізації на основі діапазону дат, вибраного користувачем у повзунку. Ця функція фільтруватиме відповідні дані для вибраного діапазону, індексуватиме дані та передаватиме їх до функцій візуалізації для створення оновлених графіків.

Модуль матиме умовний блок, який запускатиме веб-додаток Dash у режимі розробки, якщо модуль запущений безпосередньо. Це дозволить розробникам тестувати та налагоджувати додаток під час розробки.

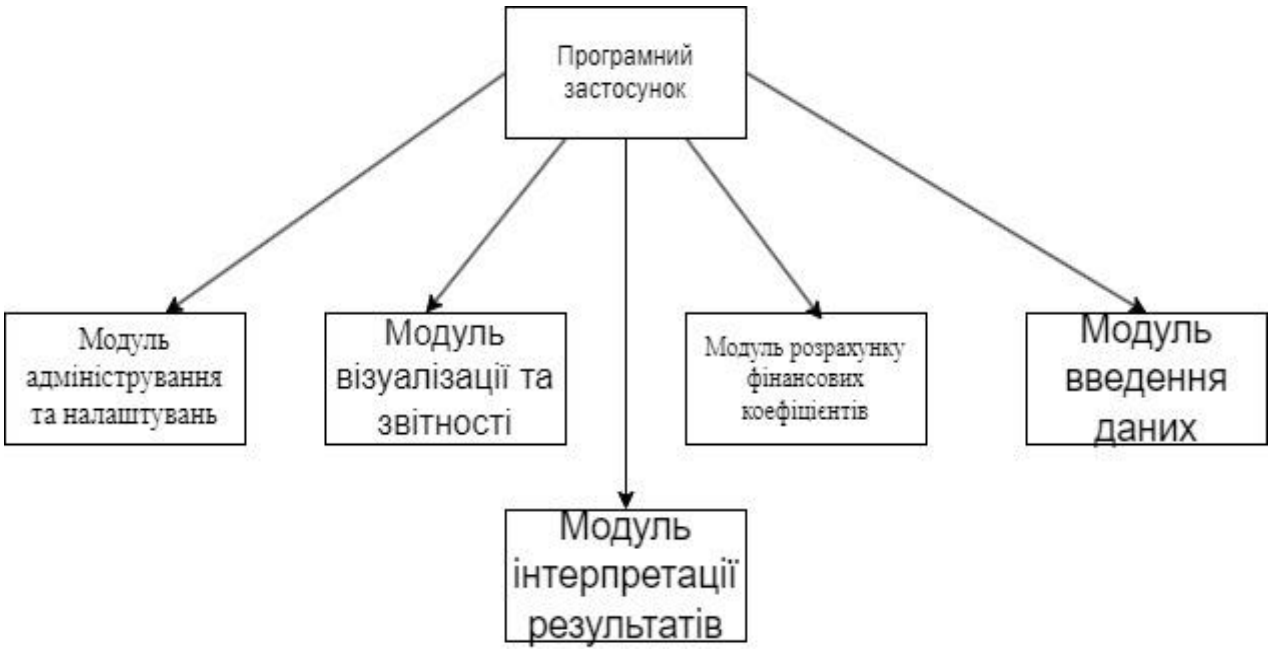

Рисунок 2.1 – Структура програмного модуля додатку

Також зобразимо діаграму прецедентів діагностики фінансового стану підприємства.

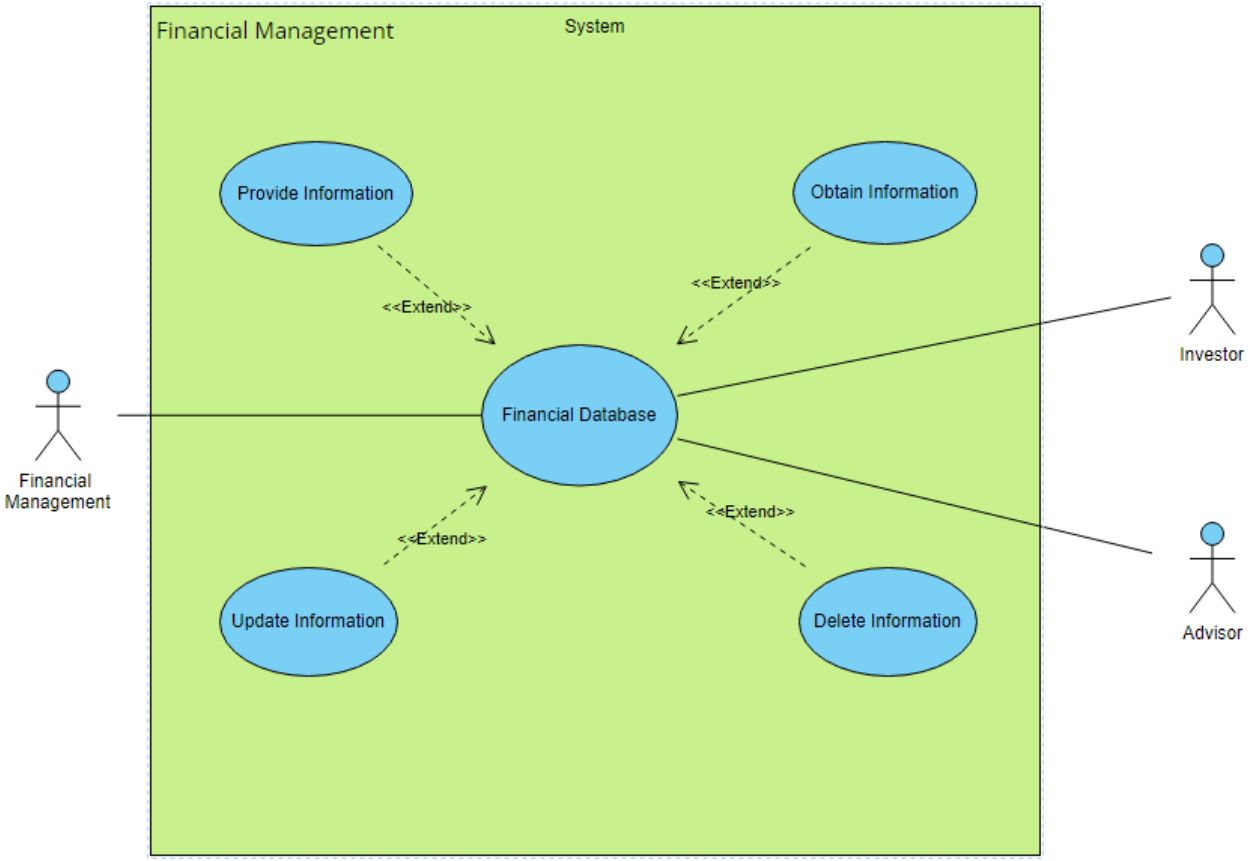

Рисунок 2.2 – Діаграма прецедентів діагностики фінансового стану підприємства

Дана діаграма представляє діаграму прецедентів для системи фінансового менеджменту. Головними компонентами є: фінансова база даних, яка є центральним елементом для зберігання та управління фінансовою інформацією. Учасниками є фінансовий менеджмент, відповідальний за управління фінансовими операціями, інвестор та консультант як зовнішні суб'єкти, які можуть отримувати інформацію з фінансової бази даних. Прецеденти використання включають: надання інформації, що дозволяє фінансовому менеджменту надавати інформацію до бази даних, отримання інформації, що дає змогу інвестору та консультанту отримувати інформацію з бази даних, оновлення інформації для фінансового менеджменту, видалення інформації для фінансового менеджменту. Зв'язки "Extend" вказують, що

прецедент "Отримати інформацію" розширюється для інвестора та консультанта.

2.2 Розробка алгоритму роботи системи

Розробка ефективного алгоритму роботи є критично важливою для забезпечення коректного функціонування програмного модуля оцінки фінансового стану підприємства. Алгоритм повинен враховувати всі необхідні етапи процесу діагностики та забезпечувати логічну послідовність дій. На рисунку 2.3 наведемо приблизний алгоритм функціонування системи та розглянемо детально основні кроки алгоритму.

1. Введення вхідних даних:
   - Забезпечення можливості завантаження даних бухгалтерської звітності з різних джерел (файлів, баз даних, інтеграція з обліковими системами);
   - Валідація та перевірка коректності введених даних.
2. Розрахунок фінансових коефіцієнтів:
   - Застосування формул та алгоритмів розрахунку показників ліквідності, фінансової стійкості, ділової активності, рентабельності;
   - Можливість додавання нових коефіцієнтів та методик розрахунку.
3. Порівняння з нормативними значеннями:
   - Визначення галузевих норм або внутрішніх стандартів підприємства для кожного коефіцієнта;
   - Виявлення відхилень розрахованих показників від нормативних діапазонів.
4. Інтерпретація результатів:
   - Застосування бази знань з експертними правилами для аналізу отриманих значень коефіцієнтів;

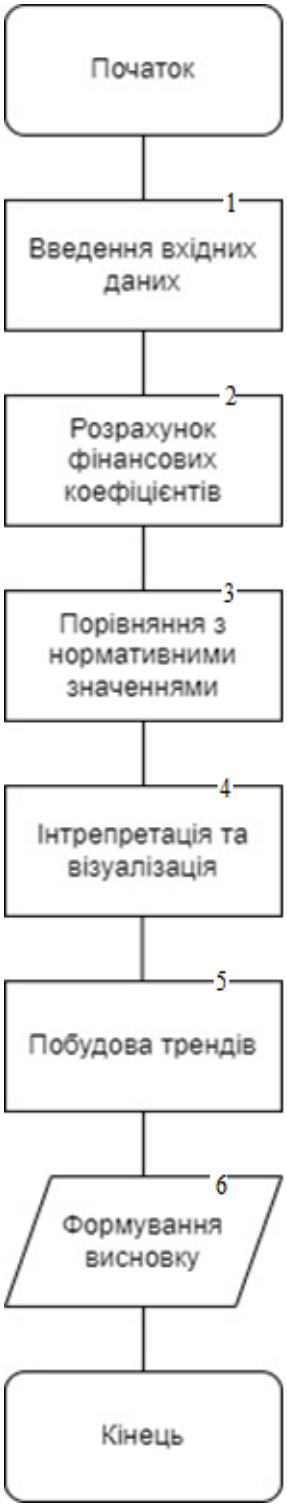

Рисунок 2.3 – Алгоритм функціонування системи

- Формування висновків щодо поточного фінансового стану підприємства (задовільний, незадовільний, критичний тощо);

- Генерація рекомендацій для покращення фінансового стану на основі виявлених проблемних аспектів.

5. Візуалізація та звітність:

- Відображення розрахованих коефіцієнтів у вигляді таблиць, діаграм, графіків;
- Формування комплексного звіту з висновками, рекомендаціями та аналітичними матеріалами;
- Можливість експорту звітів у різні формати (PDF, Excel тощо).

6. Історичний аналіз та тренди:

- Збереження результатів попередніх діагностик у базі даних;
- Відстеження динаміки зміни фінансових показників у часі;
- Виявлення трендів та тенденцій для прогнозування майбутнього стану.

7. Адміністрування та налаштування:

- Управління обліковими записами користувачів та правами доступу;
- Налаштування параметрів роботи модуля (галузеві норми, методики розрахунку тощо);
- Оновлення бази знань та правил інтерпретації результатів.

2.3 Обгрунтування вибору мови програмування та середовища розробки

Вибір відповідних інструментів для технічної реалізації програмного модуля діагностики фінансового стану підприємства є критично важливим етапом, який значною мірою визначає успіх проекту, його масштабованість, продуктивність та простоту подальшої підтримки і розвитку. Розглянемо детально обґрунтування вибору цих інструментів.

Мова програмування: Для розробки додатку рекомендується використовувати сучасну об'єктно-орієнтовану мову програмування, таку як

Java або C#. Ці мови забезпечують високий рівень абстракції, модульність та масштабованість коду, що полегшує підтримку та розширення функціоналу в майбутньому. Крім того, вони мають потужні бібліотеки та фреймворки для створення користувацьких інтерфейсів, роботи з базами даних та іншими необхідними компонентами.

Мову програмування було обрано виходячи з порівняльної таблиці.

Таблиця 2.1 – Порівняльна таблиця мов програмування

Критерій	**Java**	**C#**	**Python**	**C++**
Тип мови	Об'єктно-орієнтована	Об'єктно-орієнтована	Багато-парадигмальна	Багато-парадигмальна
Переносимість	Висока (Java Virtual Machine)	Висока (.NET Framework / .NET Core)	Висока (інтерпре-тована)	Низька (компілятор для кожної платформи)
Продуктивність	Висока	Висока	Помірна	Дуже висока
Бібліотеки та фреймворки	Багато (Spring, Hibernate, JavaFX та ін.)	Багато (.NET Framework, ASP.NET, WPF та ін.)	Багато (NumPy, Pandas, Matplotlib та ін.)	Стандартна бібліотека
Підтримка GUI	Добра (JavaFX, Swing)	Добра (WPF, Windows Forms)	Обмежена (PyQt, Tkinter)	Обмежена (Qt, GTK+)
Зручність для розробки	Висока	Висока	Дуже висока	Помірна
Навчальні ресурси	Багато	Багато	Багато	Помірно
Популярність	Дуже висока	Висока	Висока	Помірна
Підтримка паралельних обчислень	Добра (java.util.concurrent)	Добра (TPL, PLINQ)	Добра (multiprocessing threading)	Добра (OpenMP, Intel TBB)
Безпека	Висока	Висока	Помірна	Низька

На основі наведеної порівняльної таблиці мов програмування Python було обрано як мову розробки для даного програмного модуля з наступних причин:

Python є багатопарадигмальною мовою програмування, що дозволяє використовувати різні парадигми, такі як об'єктно-орієнтоване, функціональне та процедурне програмування. Це забезпечує гнучкість та можливість вибору найбільш підходящого підходу для вирішення конкретних завдань. Водночас Python характеризується дуже високою зручністю для розробки завдяки своєму лаконічному та зрозумілому синтаксису, що прискорює процес написання коду та полегшує його подальше супроводження.

Однією з головних переваг Python є його висока переносимість між різними операційними системами та платформами. Оскільки Python є інтерпретованою мовою, його код може виконуватись на будь-якій системі, де встановлено відповідне середовище виконання, без необхідності додаткової компіляції для кожної цільової платформи. Це робить Python ідеальним вибором для створення крос-платформних додатків, таких як веб-додаток, що розробляється в рамках цього проекту.

Незважаючи на те, що продуктивність Python вважається помірною порівняно з компільованими мовами, такими як C++ чи Java, для багатьох задач, включно з обробкою даних та візуалізацією, вона є цілком достатньою. Python має величезну кількість бібліотек та фреймворків, які значно розширюють його можливості та дозволяють ефективно вирішувати широкий спектр задач.

Ще однією перевагою Python є наявність великої кількості навчальних ресурсів та активної спільноти розробників, що полегшує вивчення та застосування цієї мови. Python також користується високою популярністю в багатьох галузях, включаючи аналіз даних, машинне навчання, веб-розробку та наукові обчислення, що забезпечує йому постійну підтримку та розвиток.

Середовище розробки: Для зручності розробки та налагодження коду

рекомендується використовувати інтегроване середовище розробки (IDE), таке як IntelliJ IDEA або Visual Studio. Ці IDE забезпечують багатий набір інструментів для написання, рефакторингу, відлагодження коду та інтеграції з системами контролю версій.

Система управління базами даних (СУБД): Для зберігання історичних даних діагностики, користувачів та налаштувань модуля доцільно використовувати реляційну СУБД, наприклад, MySQL або PostgreSQL. Вони є безкоштовними, надійними та масштабованими рішеннями, що підтримують SQL та мають великі спільноти користувачів.

Бібліотеки для візуалізації даних: Для відображення розрахованих фінансових коефіцієнтів та показників у вигляді графіків, діаграм та таблиць можна використовувати спеціалізовані бібліотеки, такі як JFreeChart для Java або ZedGraph для C#. Вони забезпечують багатий функціонал для створення різноманітних типів візуалізацій та легко інтегруються в додатки.

Генератори звітів: Для формування комплексних звітів з результатами діагностики фінансового стану рекомендується використовувати спеціалізовані інструменти, такі як JasperReports або Crystal Reports. Вони дозволяють створювати гнучкі та стилізовані звіти, які можна експортувати в різні формати, такі як PDF, Excel або HTML.

Система контролю версій: Для забезпечення належного управління змінами коду та спільної роботи в команді розробників необхідно використовувати систему контролю версій, наприклад, Git або Subversion. Ці інструменти дозволяють відстежувати зміни в коді, об'єднувати зміни від різних розробників та повертатися до попередніх версій у разі необхідності.

Інструменти безперервної інтеграції та розгортання: Для автоматизації процесів збірки, тестування та розгортання програмного забезпечення рекомендується використовувати інструменти безперервної інтеграції та розгортання, такі як Jenkins або Travis CI. Вони дозволяють скоротити час на ручні операції та забезпечити стабільність та надійність процесу розробки.

Перед початком реалізації додатку важливо обрати середовище розробки, адже це допоможе суттєво скоротити час на розробку, використовуючи найбільш оптимальні інструменти, а також якісно провести дослідницьку роботу.

Таблиця 2.2 – Порівняльна таблиця середовищ розробки

Критерій	**IntelliJ IDEA**	**Visual Studio**	**Jupyter Notebook**
Мови програмування	Java, Kotlin, Scala, Python та інші	C#, C++, F#, Python та інші	Python, R, Julia та інші
Платформа	Крос-платформна	Windows, macOS, Linux	Крос-платформна
Тип	Повноцінне IDE	Повноцінне IDE	Веб-середовище
Документація та підтримка	Відмінна	Відмінна	Добра
Інтеграція з Git	Вбудована	Вбудована	Вимагає додаткових розширень
Налагодження	Потужні інструменти	Потужні інструменти	Обмежені інструменти
Рефакторинг коду	Широкі можливості	Широкі можливості	Обмежені
Інтеграція з фреймворками	Підтримуються популярні фреймворки	Підтримуються популярні фреймворки	Підтримуються популярні фреймворки
Робота з ноутбуками	Обмежена підтримка	Обмежена підтримка	Основна функціональність
Візуалізація даних	Обмежена	Обмежена	Потужні можливості

З огляду на особливості даного проекту, а саме розробку веб-додатку з використанням бібліотек Python буде доцільно використати такі бібліотеки як: pandas, plotly.graph_objects та NumPy.

Pandas - високорівнева бібліотека Python для аналізу даних. Чому її називають високорівневою, тому що побудована вона поверх низькорівневої бібліотеки NumPy (написана на C), що є великим плюсом у продуктивності. В екосистемі Python, pandas є найбільш просунутою бібліотекою, що швидко розвивається, для обробки та аналізу даних.

Щоб ефективно працювати з pandas, необхідно освоїти найголовніші структури даних бібліотеки: DataFrame та Series. Без розуміння що вони собою представляють, неможливо надалі проводити якісний аналіз. **Series** – це маркована одновимірна структура даних, її можна як таблицю з одним рядком. З Series можна працювати як зі звичайним масивом (звертатися за номером індексу), і як з асоційованим масивом, коли можна використовувати ключ доступу до елементів даних.

DataFrame – це двовимірна маркована структура. Ідейно вона дуже схожа на звичайну таблицю, що виражається у способі її створення та роботі з її елементами. Для реалізації нам знадобиться працювати безпосередньо з DataFrame.

DataFrame можна сприймати як узагальнений масив NumPy. Якщо об'єкт Series – аналог одновимірного масиву з гнучкими індексами, об'єкт DataFrame – аналог двовимірного масиву з гнучкими індексами рядків та гнучкими іменами стовпців. Аналогічно тому, що двовимірний масив можна розглядати як упорядковану послідовність вирівняних стовпців, об'єкт DataFrame можна розглядати як впорядковану послідовність вирівняних об'єктів Series. Під «вирівняними» мається на увазі те, що вони використовують один і той самий індекс.

Якщо Series є одномірною структурою, яку собі можна представити як таблицю з одним рядком, то DataFrame – це вже двовимірна структура – повноцінна таблиця з безліччю рядків і стовпців.

Конструктор класу DataFrame виглядає так:

class pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)

data – масив ndarray,
словник (dict) чи інший DataFrame;
index – перелік міток для записів (імена рядків таблиці);
columns – список міток для полів (імена шпальт таблиці);
dtype – об'єкт numpy.dtype, що визначає тип даних;
copy – створює копію масиву даних, якщо параметр дорівнює True, інакше нічого не робить.

Структуру DataFrame можна створити на базі:
словника (dict) як елементи якого мають виступати: одномірні ndarray, списки, інші словники, структури Series;
двовимірні ndarray;
структури Series;
структуровані ndarray;
інші DataFrame.

Приклад використання операцій:

```
import numpy as np
import pandas as pd
#створимо DataFrame.
d = {"price": np.array([1, 2, 3]), "count": np.array([10, 20, 30])}
df = pd.DataFrame(d, index=['a', 'b', 'c'])
print(df)
 #Операція: вибору колонки.
print(df['count'])
# Операція: вибору рядку по мітці.
print(df.loc['a'])
```

```
# Операція: вибору рядку по індексу.
print(df.iloc[1])
#Операция: slice по рядкам.
print( df[0:2])
# Операція: вибору рядків що задовольняють умові
print(df[df['count'] >= 20])
```

Розглянемо методи роботи з елементами DataFrame, що найчастіше використовуються.

Таблиця 2.3 - елементи DataFrame, що найчастіше використовуються.

Операція	Синтаксис	Результат, що повертається
Вибір стовпця	df[col]	Series
Вибір рядка за міткою	df.loc[label]	Series
Вибір рядка за індексом	df.iloc[loc]	Series
Слайс за рядками	df[0:4]	DataFrame
Вибір рядків, які відповідають умові	df[bool_vec]	DataFrame

Plotly.graph_objects це бібліотека реалізована для створення, оброблення та відтворення графічних об’єктів які представлені деревоподібними структурами даних. Вони автоматично серіалізуються в JSON для відтворення бібліотекою JavaScript Plotly.js. Ці дерева складаються з іменованих вузлів, які називаються «атрибутами», їх структура визначається схемою фігур Plotly.js, яка доступна в машиночитаній формі. Модуль plotly.graph_objects (зазвичай імпортується як go) містить автоматично згенеровану ієрархію класів Python, які представляють нелистові вузли в цій схемі фігури. Термін "об'єкти графів" відноситься до екземплярів цих класів. Основними класами, визначеними в модулі plotly.graph_objects, є Figure та ipywidgets-сумісний варіант під назвою FigureWidget, які обидва представляють цілі фігури. Екземпляри цих класів мають багато зручних методів для Pythonical маніпулювання їхніми атрибутами (наприклад,

.update_layout() або .add_trace(), які всі приймають «магічне підкреслення» позначення), а також їх рендерингу (наприклад, .show()) та експорту їх до різні формати (наприклад, .to_json() або .write_image() або .write_html()).Кожен нелистовий атрибут фігури представлений екземпляром класу в ієрархії plotly.graph_objects. Наприклад, фігура fig може мати атрибут layout.margin, який містить атрибути t, l, b і r, які є листками дерева: вони не мають дітей. Поле fig.layout є об'єктом класу plotly.graph_objects.Layout, а fig.layout.margin є об'єктом класу plotly.graph_objects.layout.Margin, який представляє вузол поля, і має поля t, l, b та r, що містить значення відповідних листових вузлів. Зауважте, що вказати всі ці значення можна без створення проміжних об'єктів за допомогою нотації «магічного підкреслення»: go.Figure(layout_margin=dict(t=10, b=10, r=10, l=10)).

Об'єкти, що містяться у списку, який є значенням атрибутивних даних, називаються «слідами» та можуть бути одного з понад 40 можливих типів, кожен із яких має відповідний клас у plotly.graph_objects. Наприклад, сліди типу scatter представлені екземплярами класу plotly.graph_objects.Scatter. Це означає, що фігура, створена як go.Figure(data=[go.Scatter(x=[1,2], y=[3,4)]), матиме представлення JSON {"data": [{"type" : "scatter", "x": [1,2], "y": [3,4]}]}.

Графічні об'єкти порівняно зі словниками

Об'єкти Graph мають кілька переваг порівняно зі звичайними словниками Python:

Графічні об'єкти забезпечують точну перевірку даних. Якщо ви вкажете недійсне ім'я властивості або недійсне значення властивості як ключ до об'єкта графіка, буде створено виняток із корисним повідомленням про помилку з описом проблеми. Це не той випадок, якщо ви використовуєте звичайні словники та списки Python для створення своїх фігур.

Об'єкти Graph містять описи кожної дійсної властивості як рядки документів Python із повним посиланням на API. Ви можете використовувати

ці рядки документів у обраному вами середовищі розробки, щоб дізнатися про доступні властивості як альтернативу перегляду повного довідника онлайн.
Доступ до властивостей об’єктів графа можна отримати за допомогою пошуку ключа у стилі словника (наприклад, fig["layout"]) або доступу до властивостей у стилі класу (наприклад, fig.layout).

Графічні об’єкти підтримують зручні функції вищого рівня для оновлення вже побудованих фігур (.update_layout(), .add_trace() тощо).
Конструктори графічних об’єктів і методи оновлення приймають «магічні підкреслення» (наприклад, go.Figure(layout_title_text="The Title"), а не dict(layout=dict(title=dict(text="The Title")))) для більш компактного коду .
Об’єкти Graph підтримують прикріплену візуалізацію (.show()) і функції експорту (.write_image()), які автоматично викликають відповідні функції з модуля plotly.io.

Рекомендований спосіб створення фігур — це використання функцій у модулі plotly.express, спільно відомому як Plotly Express, усі вони повертають екземпляри plotly.graph_objects.Figure, тому кожна фігура, створена за допомогою бібліотеки plotly, фактично використовує графічні об’єкти під капотом, якщо не створено вручну зі словників.

Тим не менш, певні типи фігур поки що неможливо створити за допомогою Plotly Express, наприклад фігури, які використовують певні типи 3D трасування, як-от сітка або ізоповерхня. Крім того, деякі фігури важко створювати, починаючи з фігури, створеної за допомогою Plotly Express, наприклад, фігури з підсхемами різних типів, двовісними графіками або фасетними графіками з декількома різними типами трас. Щоб побудувати такі фігури, може бути легше розпочати з порожнього об’єкта plotly.graph_objects.Figure (або об’єкта, налаштованого з підсхемами за допомогою функції make_subplots()) і поступово додавати трасування та оновлювати атрибути, як описано вище.

NumPy не просто працює з багатомірними масивами, але робить це швидко. Взагалі, інтерпретовані мови роблять вичислення повільніше тих які компілюються, а Python як раз мова яка підлягає інтерпретації. NumPy створений так, щоб ефективно працювати з наборами чисел будь-якого розміру в Python.

NumPy — основний пакет для наукових обчислень на Python. Це бібліотека Python, яка надає об’єкт багатовимірного масиву, різноманітні похідні об’єкти (такі як замасковані масиви та матриці), а також ряд процедур для швидких операцій над масивами, включаючи математичні, логічні, маніпуляції формою, сортування, вибір, введення/виведення. , дискретне перетворення Фур'є, базова лінійна алгебра, основні статистичні операції, випадкове моделювання та багато іншого.

В основі пакета NumPy лежить об’єкт ndarray. Це інкапсулює n-вимірні масиви однорідних типів даних, причому багато операцій виконуються в скомпільованому коді для підвищення продуктивності. Існує кілька важливих відмінностей між масивами NumPy і стандартними послідовностями Python:

Масиви NumPy мають фіксований розмір під час створення, на відміну від списків Python (які можуть динамічно зростати). Зміна розміру масиву ndarray створить новий масив і видалить оригінальний.

Усі елементи в масиві NumPy мають мати однаковий тип даних і, таким чином, матиме однаковий розмір у пам’яті. Виняток: можна мати масиви (Python, включаючи NumPy) об’єктів, що дозволяє створювати масиви елементів різного розміру.

Масиви NumPy полегшують розширені математичні та інші види операцій над великою кількістю даних. Як правило, такі операції виконуються ефективніше та з меншою кількістю коду, ніж це можливо за допомогою вбудованих послідовностей Python.

Зростаюча кількість наукових і математичних пакетів на основі Python використовує масиви NumPy; хоча вони зазвичай підтримують введення послідовності Python, вони перетворюють такі введення в масиви NumPy перед обробкою, і вони часто виводять масиви NumPy. Іншими словами, щоб ефективно використовувати більшу частину (можливо, навіть більшість) сьогоднішнього наукового/математичного програмного забезпечення на основі Python, просто знати, як використовувати вбудовані типи послідовностей Python, недостатньо - потрібно також знати, як використовувати масиви NumPy.

Пункти про розмір і швидкість послідовності особливо важливі в наукових обчисленнях. Як простий приклад розглянемо випадок множення кожного елемента в одновимірній послідовності з відповідним елементом в іншій послідовності такої ж довжини. Якщо дані зберігаються в двох списках Python, a і b, ми можемо повторити кожен елемент:

```
c = []
for i in range(len(a)):
    c.append(a[i]*b[i])
```

Це дає правильну відповідь, але якщо кожне з a і b містить мільйони чисел, ми заплатимо ціну за неефективність циклу в Python. Ми могли б виконати те саме завдання набагато швидше на C, написавши (для ясності ми нехтуємо оголошеннями та ініціалізаціями змінних, розподілом пам’яті тощо)

```
for (i = 0; i < rows; i++) {
 c[i] = a[i]*b[i];
}
```

Це економить усі накладні витрати, пов’язані з інтерпретацією коду Python і маніпулюванням об’єктами Python, але за рахунок переваг, отриманих від кодування на Python. Крім того, необхідна робота з кодування збільшується разом із розмірністю наших даних. Наприклад, у випадку 2-D масиву код C (скорочений, як і раніше) розширюється до

```
for (i = 0; i < rows; i++) {
 for (j = 0; j < стовпці; j++) {
 c[i][j] = a[i][j]*b[i][j];
 }
```

NumPy дає нам найкраще з обох світів: поелементні операції є «режимом за замовчуванням», коли використовується ndarray, але поелементна операція швидко виконується попередньо скомпільованим кодом C. У NumPy

```
c = a * b
```

робить те ж саме, що й попередні приклади, зі швидкістю, близькою до C, але з простотою коду, яку ми очікуємо від чогось на основі Python. Дійсно, ідіома NumPy ще простіша! Цей останній приклад ілюструє дві функції NumPy, які є основою більшої частини його потужності: векторизація та трансляція.

Векторизація описує відсутність будь-яких явних циклів, індексування тощо в коді - ці речі відбуваються, звичайно, просто "за лаштунками" в оптимізованому, попередньо скомпільованому коді C. Векторизований код має багато переваг, серед яких:

Векторизований код більш стислий і легший для читання;

Менше рядків коду зазвичай означає менше помилок;

код більше нагадує стандартну математичну нотацію (що полегшує, як правило, правильне кодування математичних конструкцій) векторизація призводить до більш «Pythonic» коду. Без векторизації наш код був би усіяний неефективними і складними для читання циклами for.

Трансляція — це термін, який використовується для опису неявної поелементної поведінки операцій; загалом кажучи, у NumPy усі операції, не лише арифметичні, а й логічні, порозрядні, функціональні тощо, поводяться таким неявним поелементним способом, тобто транслюють. Крім того, у наведеному вище прикладі a і b можуть бути багатовимірними масивами однакової форми, або скаляром і масивом, або навіть двома масивами з

різними формами, за умови, що менший масив «розширюється» до форми більшого в таким чином, щоб кінцева трансляція була однозначною. Детальні «правила» трансляції дивіться в розділі «Трансляція».

NumPy повністю підтримує об’єктно-орієнтований підхід, починаючи знову ж таки з ndarray. Наприклад, ndarray — це клас, який має численні методи та атрибути. Багато з його методів віддзеркалюються функціями в самому зовнішньому просторі імен NumPy, що дозволяє програмісту кодувати в будь-якій парадигмі, яка йому подобається. Ця гнучкість дозволила діалекту масиву NumPy і класу NumPy ndarray стати де-факто мовою обміну багатовимірними даними, що використовується в Python.

Також NumPy це бібліотека Python з відкритим кодом, яка широко використовується в науці та техніці. Бібліотека NumPy містить багатовимірні структури даних масиву, такі як однорідний N-вимірний ndarray, і велику бібліотеку функцій, які ефективно працюють із цими структурами даних. Дізнайтеся більше про NumPy на сайті What is NumPy, а якщо у вас є коментарі чи пропозиції, зв’яжіться з ними! Бібліотека частково написана на Python, а частково на C і C++ — у тих місцях, які вимагають швидкості. Крім того, код NumPy оптимізований під більшість сучасних процесорів. Як і в Matlab, для NumPy існують пакети, що розширюють її функціональність, — наприклад, бібліотека SciPy або Matplotlib.

Списки Python є чудовими контейнерами загального призначення. Вони можуть бути «гетерогенними», що означає, що вони можуть містити елементи різних типів, і вони досить швидкі, коли використовуються для виконання окремих операцій над кількома елементами.

Залежно від характеристик даних і типів операцій, які необхідно виконати, інші контейнери можуть бути більш доцільними; Використовуючи ці характеристики, ми можемо покращити швидкість, зменшити споживання пам’яті та запропонувати синтаксис високого рівня для виконання

різноманітних типових завдань обробки. NumPy сяє, коли є велика кількість «однорідних» (однотипних) даних, які потрібно обробити на ЦП.

Основним об'єктом NumPy є однородний багатомірний масив (в numpy називається numpy.ndarray). Це багатомірний масив елементів (звичайно чисел), одного типу.

Найбільш важливі атрибути об'єктів ndarray:

ndarray.ndim - число вимірювань (чаще їх називають "осі") масиву.

ndarray.shape - розміри масиву, його форма. Це кортеж натуральних чисел, що показує довжину масиву по кожній осі. Для матриці з n рядків і m стовпців форма буде (n,m). Число елементів кортежа shape рівно ndim. ndarray.size - кількість елементів масиву. Очевидно, рівні множенню всіх елементів атрибута shape.ndarray.dtype - об'єкт, описуючий тип елементів масиву. Можна визначити dtype, використовуючи стандартні типи даних Python. NumPy тут надає цілий набір можливостей, як вбудованих, наприклад: bool_, character, int8, int16, int32, int64, float8, float16, float32, float64, complex64, object_, так і можливість визначити власні типи даних, у тому числі та складові. ndarray.itemsize - розмір кожного елемента масиву в байтах.

ndarray.data - буфер, що містить фактичні елементи масиву. Зазвичай не потрібно використовувати цей атрибут, так як звертатися до елементів масиву просто за допомогою індексів

Тому посилаючись на вище перечислені бібліотеки, для обробки даних та візуалізації, найбільш доцільним вибором середовища розробки буде Jupyter Notebook.

Jupyter Notebook є веб-середовищем, яке особливо зручне для роботи з Python, особливо в сферах обробки даних, машинного навчання та наукових обчислень. Його основною перевагою є можливість комбінувати код, візуалізацію даних та пояснювальний текст в одному ноутбуку, що значно полегшує розробку, документацію та презентацію проектів.

Крім того, Jupyter Notebook має потужні можливості для візуалізації даних, що є критично важливим для даного проекту. Бібліотеки, такі як Matplotlib, Plotly та Bokeh, легко інтегруються з Jupyter Notebook, дозволяючи створювати якісні інтерактивні візуалізації безпосередньо в середовищі розробки.

Хоча Jupyter Notebook не є повноцінним інтегрованим середовищем розробки (IDE), він забезпечує достатні можливості для написання, редагування та відлагодження коду Python. Крім того, існує безліч доповнень та розширень, які можуть розширити функціональність Jupyter Notebook, наприклад, для інтеграції з системами контролю версій, такими як Git.

2.4 Висновки до розділу 2

Визначено основні компоненти програмного модуля: модуль введення даних, модуль розрахунку фінансових коефіцієнтів, модуль інтерпретації результатів, модуль візуалізації та звітності, а також модуль адміністрування та налаштувань.

Розроблено детальний алгоритм функціонування системи, який охоплює всі етапи процесу діагностики: від введення вхідних даних до генерації звітів, історичного аналізу трендів та адміністрування системи. Алгоритм забезпечує логічну послідовність дій та враховує необхідність валідації даних, порівняння з нормативними значеннями, інтерпретації результатів та формування рекомендацій.

Обґрунтовано вибір засобів реалізації системи, таких як мова програмування (Java або C#), інтегроване середовище розробки (IntelliJ IDEA або Visual Studio), система управління базами даних (MySQL або PostgreSQL), бібліотеки для візуалізації даних (JFreeChart або ZedGraph), генератори звітів (JasperReports або Crystal Reports), система контролю версій (Git або Subversion) та інструменти безперервної інтеграції та розгортання (Jenkins або Travis CI).

Проведено порівняльний аналіз альтернативних варіантів для вибору оптимальних інструментів реалізації, враховуючи такі критерії, як продуктивність, наявність бібліотек і фреймворків, підтримка графічного інтерфейсу, зручність розробки, навчальні ресурси, популярність, підтримка паралельних обчислень та безпека.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО МОДУЛЯ ДІАГНОСТИКИ ФІНАНСОВОГО СТАНУ ПІДПРИЄМСТВА

3.1 Робота з вибіркою даних

Модуль завантажує дані з CSV-файлів для трьох наборів даних: "Частка загального чистого вартості" (df_s), "Загально чистий капітал" (df_h) та "Економічні показники" (df_ec). Дані перетворюються на об'єкти DataFrame бібліотеки Pandas для подальшої обробки.

Фільтрація даних за періодом: Користувач може вибрати діапазон дат за допомогою повзунка на веб-інтерфейсі. Функція update_output обробляє вибраний діапазон дат та фільтрує відповідні фінансові дані для візуалізації. Вибірка даних виглядає наступним чином.

1	date	median_house	sp500	consumer_sentiment	purchase_power	industrial	gdp	debt_held_fed
2	1/1/00	165300	142.1478487	107.1	58.4	92.4733	12935.252	501.7
3	4/1/00	163200	144.9587445	106.4	58	93.3908	13170.749	504.9
4	7/1/00	168800	147.7976676	106.8	57.6	93.3533	13183.89	511.4
5	10/1/00	172900	136.6991579	98.4	57.5	92.7388	13262.25	511.7
6	1/1/01	169800	127.8595651	91.5	56.7	91.3567	13219.251	523.9
7	4/1/01	179000	123.618149	92.6	56.2	90.0322	13301.394	535.1
8	7/1/01	172500	114.6359414	81.8	56.1	88.9262	13248.142	534.1
9	10/1/01	171100	112.0993393	88.8	56.6	88.0897	13284.881	551.7
10	1/1/02	188700	113.5118062	95.7	55.9	89.3624	13394.91	575.4
11	4/1/02	187200	107.2360759	92.4	55.6	90.9078	13477.356	590.7
12	7/1/02	178100	89.79927239	86.1	55.3	90.8931	13531.741	604.2
13	10/1/02	190100	89.2474524	86.7	55.3	90.6374	13549.421	629.4
14	1/1/03	186000	86.41781906	77.6	54.3	91.221	13619.434	641.5
15	4/1/03	191800	94.18095277	89.7	54.4	90.7372	13741.107	652.1
16	7/1/03	191900	100.478586	87.7	54	91.6355	13970.157	656.1
17	10/1/03	198800	106.1149402	92.6	54.3	92.4019	14131.379	666.7
18	1/1/04	212700	113.7167257	95.8	53.4	92.7403	14212.34	674.1

Рисунок 3.1 – Формування вибірки даних

Методи візуалізації розподілу Капіталу:

1. Функція plot_wealth(df_share, df_networth) створює візуалізації для розподілу Капіталу за процентилями. Вона використовує бібліотеку Plotly для створення складених графіків, які відображають:

– Частку загального чистого вартості за процентилями (перекривні області)

– Зміну загального чистого Капіталу, утримуваного кожним процентилем (багатолінійний графік)

– Ефективну ставку фонду (лінійний графік)

– Баланс (лінійний графік)

2. Для візуалізації зміни загального чистого Капіталу, утримуваного кожним процентилем, використовується індексація даних. Функція index_data(data) перетворює вихідні дані, встановлюючи початкове значення як 0 на задану дату. Це дозволяє порівнювати темпи зростання між процентилями протягом вибраного періоду.

Методи візуалізації фінансової та реальної економіки: а. Функція plot_economy(df_economy) створює візуалізації для фінансової та реальної економіки. Вона використовує бібліотеку Plotly для створення складених графіків з груповаными стовпчиками та областю. б. Для візуалізації фінансової економіки використовуються показники: купівельна спроможність долара, S&P 500 та медіанна ціна продажу будинків. Дані також індексуються за допомогою функції index_data(data) для порівняння темпів зростання. в. Для візуалізації реальної економіки використовуються показники: реальний ВВП, індекс споживчих настроїв та промислове виробництво. Ці дані також індексуються.

Модуль завантажує дані з CSV-файлів для трьох наборів даних: "Частка загального чистого вартості" (df_s), "Загально чистий капітал" (df_h) та "Економічні показники" (df_ec), і перетворює їх на об'єкти DataFrame бібліотеки Pandas для подальшої обробки. Коли користувач вибирає діапазон

дат за допомогою повзунка на веб-інтерфейсі, функція update_output обробляє цей діапазон і фільтрує відповідні фінансові дані для візуалізації.

Для набору даних "Частка загального чистого вартості" (df_s) підпрограма створює новий DataFrame df_share, який містить стовпчики "date", "Top 1%", "90th to 99th", "50th to 90th", "Bottom 50%", "EFFR" (ефективна ставка федеральних фондів). Дані в цих стовпчиках фільтруються за вибраним діапазоном дат, а значення стовпчиків з відсотками ("Top 1%", "90th to 99th", "50th to 90th", "Bottom 50%", "EFFR") діляться на 100 для відображення їх у форматі відсотків.

Для набору "Загально чистий капітал" (df_h) підпрограма створює новий DataFrame df_networth, що містить стовпчики "date", "Top 1%", "90th to 99th", "50th to 90th", "Bottom 50%", "balance_sheet". Дані фільтруються за вибраним діапазоном дат, а значення стовпчиків "Top 1%", "90th to 99th", "50th to 90th", "Bottom 50%" та "balance_sheet" перетворюються на індексовані дані. Це означає, що значення на початкову дату (мінімальна дата в діапазоні) встановлюється як 100, а решта значень розраховуються як відносне відхилення від цього базового значення. Таким чином, ці показники візуалізуються як зміна чистого Капіталу відносно початкової дати.

Для набору "Економічні показники" (df_ec) підпрограма створює новий DataFrame df_economy, який містить стовпчики "date", "gdp" (реальний ВВП), "consumer_sentiment" (індекс споживчих настроїв), "industrial" (промислове виробництво), "purchase_power" (купівельна спроможність долара), "sp500" (індекс S&P 500), "median_house" (медіанна ціна продажу житла), "debt_held_fed" (державний борг, що утримується Федеральним резервом). Дані в цих стовпчиках також фільтруються за вибраним діапазоном дат, а їхні значення перетворюються на індексовані показники, де значення на початкову дату встановлюється як 100, а решта значень розраховуються відносно цього базису. Це дозволяє візуалізувати відносну зміну економічних показників порівняно з початковою датою в діапазоні.

Після фільтрації даних за вибраним діапазоном дат функція update_output передає відфільтровані набори даних df_share, df_networth та df_economy до функцій візуалізації plot_wealth та plot_economy для створення інтерактивних графіків та діаграм, що відображають фінансовий стан та економічні тенденції.

Для візуалізації результатів аналізу фінансового стану компанії у вигляді графіків, діаграм та таблиць з метою полегшення інтерпретації та прийняття рішень, необхідно розробити комплексну методику. Дана методика повинна включати такі основні елементи:

1. Гістограми:

– Горизонтальні або вертикальні стовпчикові діаграми, що відображають значення фінансових показників для окремих періодів часу (квартали, роки).

– Використовуються для наочного відображення змін у часі таких показників, як дохід, витрати, прибуток, грошові потоки тощо.

– Різні кольори можуть бути використані для порівняння різних категорій фінансових даних.

2. Лінійні графіки:

– Графіки, що відображають зміни фінансових показників у часі за допомогою ліній.

– Корисні для відстеження тенденцій та візуалізації безперервних даних, наприклад, вартості акцій, курсів валют чи процентних ставок.

– Можна комбінувати кілька ліній на одному графіку для порівняння різних показників або сегментів бізнесу.

3. Кругові діаграми (пироги):

– Використовуються для відображення відносного розподілу часток або пропорцій різних категорій фінансових даних.

– Корисні для наочної демонстрації структури витрат, доходів, активів або зобов'язань компанії.

– Різні сектори пирога відображають різні категорії, а їх розмір - відносний внесок у загальну суму.

4. Стовпчикові діаграми:

– Схожі на гістограми, але використовуються для порівняння значень фінансових показників між різними категоріями або сегментами бізнесу для певного періоду.

– Корисні для порівняння продажів, доходів або витрат між різними продуктами, регіонами або відділами.

5. Таблиці:

– Структуровані представлення фінансових даних у вигляді рядків і стовпців.

– Зручні для відображення точних числових значень та деталізованих фінансових показників.

– Можуть містити розрахунки, підсумки та форматування для полегшення читання та аналізу.

6. Панелі інструментів (dashboards):

– Об'єднують кілька візуалізацій, таких як діаграми, графіки та таблиці, на одному екрані.

– Забезпечують комплексний огляд фінансового стану компанії та ключових показників ефективності.

– Можуть бути інтерактивними, дозволяючи користувачам фільтрувати та змінювати відображення даних.

Для реалізації цієї методики візуалізації можуть знадобитися такі інструменти та засоби, як спеціалізоване програмне забезпечення для бізнес-аналітики, табличні процесори (наприклад, Microsoft Excel або Google Sheets), бібліотеки візуалізації даних у мовах програмування (наприклад, Matplotlib для Python або ggplot2 для R), а також інтерфейси для створення панелей інструментів та інтерактивних візуалізацій.

Matplotlib є потужною бібліотекою для візуалізації даних у Python, і вона особливо корисна для створення комбінованих діаграм, які допомагають оцінити фінансовий стан підприємств у часовому розрізі (таймлайнах). Виводи комбінованих діаграм Matplotlib можуть бути надзвичайно корисними для аналізу фінансових показників і тенденцій, оскільки вони дозволяють поєднувати різні типи візуалізацій на одному графіку.

Наприклад, можна створити комбіновану діаграму, що складається з лінійного графіка та гістограми. Лінійний графік ідеально підходить для відображення безперервних даних, таких як ціна акцій або курс валют у часі, тоді як гістограма дозволяє наочно представити дискретні дані, такі як квартальний або річний дохід, витрати чи прибуток. Поєднання цих двох візуалізацій на одному графіку може надати цінну інформацію про взаємозв'язок між фінансовими показниками та зовнішніми факторами, наприклад, коливаннями ринку.

Matplotlib дозволяє додавати кілька осей у на одну діаграму, що дає можливість відображати різні масштаби та одиниці вимірювання на одному графіку. Це може бути особливо корисним для порівняння різних фінансових показників, таких як дохід, прибуток і грошові потоки, які можуть мати різні порядки величин.

Комбіновані діаграми в Matplotlib також можуть включати вертикальні лінії або зони, що позначають важливі події або періоди, такі як запуск нового продукту, зміни в керівництві або економічні спади. Ці візуальні індикатори допомагають легко зв'язати зміни у фінансових показниках з відповідними подіями.

Використовуючи можливості анотацій і підписів у Matplotlib, можна додавати пояснювальні примітки та деталі до комбінованих діаграм, що полегшує інтерпретацію даних і висновки.

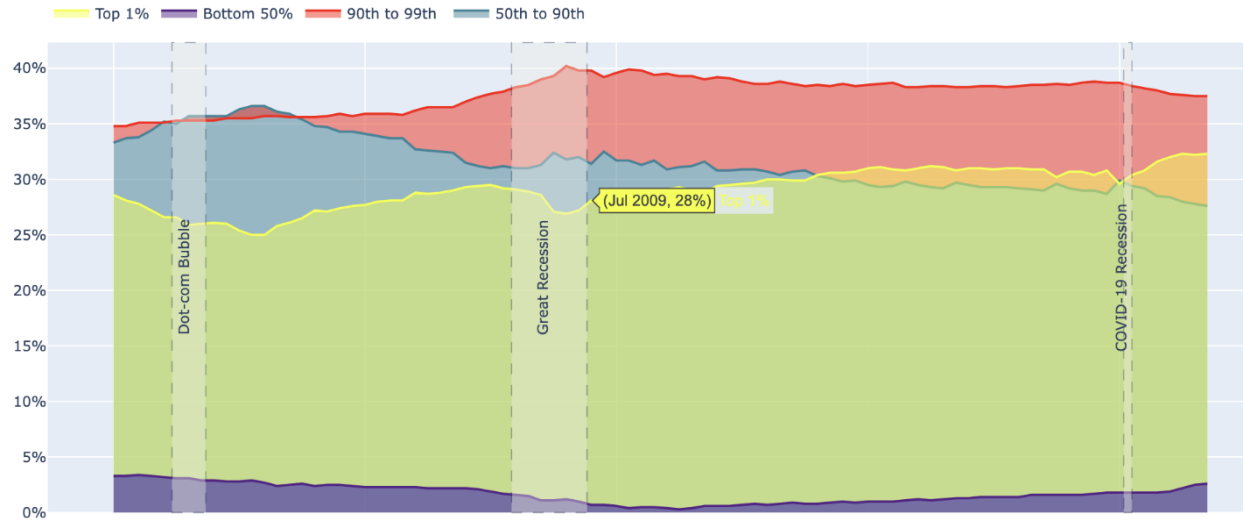

Рисунок 3.2 – Приклад застосування інтегрованої методики візуалізації даних при розробці

3.2 Тестування роботи програмного модуля діагностики фінансового стану підприємств

Розподіл Капіталу візуалізовано за допомогою двох наборів даних – "Частка сукупної чистої вартості" та "Сукупна чиста вартість, що утримується". На рисунку 3.2 нижче зображена візуалізація для обох наборів даних.

"Частка сукупної чистої вартості" показує, яку частку загального Капіталу країни володіє кожна квінтиль населення, ранжованого від найбіднішого до найбагатшого. Дані демонструють значну нерівність: верхня 20% населення володіє близько 70% сукупного чистого Капіталу, тоді як нижня 40% майже не мають чистих активів.

"Сукупна чиста вартість, що утримується" вимірює абсолютну вартість чистих активів, що належать кожній квінтилі. Очевидно, що верхня 20% населення володіє величезною часткою загального Капіталу країни у грошовому еквіваленті.

Візуалізація охоплює період з 2000 по 2021 рік і демонструє циклічні коливання, пов'язані з економічними спадами. Під час рецесій нерівність у розподілі Капіталу, як правило, зростає.

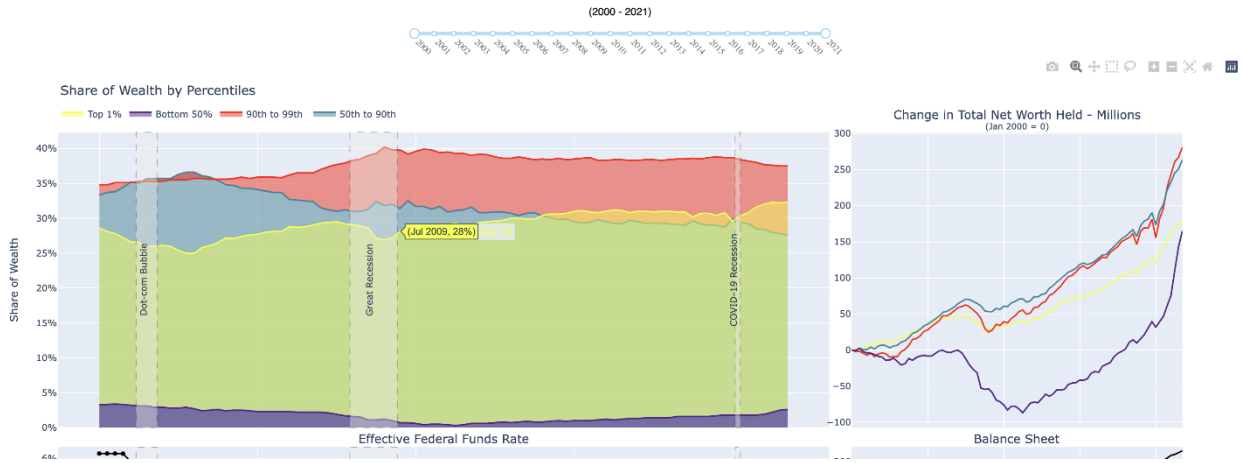

Рисунок 3.3 – Графік розподілу Капіталу

У контексті моделювання програмного модуля діагностики фінансового стану підприємств, ці візуалізації та висновки мають важливе значення. Програмний модуль, що діагностує фінансовий стан підприємств, повинен враховувати різноманітні економічні показники, такі як індикатори реальної та фінансової економіки, які демонструють стан макроекономічного середовища. Особливу увагу слід приділити змінам, що відбуваються під час різних економічних циклів, як це було під час Світової економічної кризи та рецесії COVID-19.

Основними елементами модулю є:

1. Інтеграція макроекономічних індикаторів: Модуль повинен включати індикатори, такі як споживчі настрої, фондові індекси (наприклад, S&P 500), та показники боргу. Це дозволить краще зрозуміти вплив макроекономічних умов на фінансовий стан підприємств.

2. Аналіз часових рядів: Важливо забезпечити аналіз часових рядів для відстеження змін у фінансових показниках підприємства в контексті

загальної економічної ситуації. Це дозволить виявити тенденції та аномалії, що можуть сигналізувати про потенційні проблеми.

3. Моделювання сценаріїв: Модуль має включати можливість моделювання різних економічних сценаріїв, таких як рецесія або періоди економічного зростання. Це допоможе передбачити потенційні ризики та підготувати стратегії для їхнього мінімізації.

4. Оцінка ліквідності: Враховуючи висновки про зростання частки високоліквідних активів у домогосподарств під час політики кількісного пом'якшення, модуль має забезпечувати детальний аналіз ліквідності підприємства. Це дозволить оцінити здатність підприємства витримати фінансові стреси.

5. Моніторинг ринкових умов: Постійний моніторинг ринкових умов та їхнього впливу на фінансовий стан підприємства є критичним для своєчасного прийняття управлінських рішень.

Діаграма комплексного оцінювання ряду підприємств на макрорівні в розрізі таймлайну 20 років зображена на рисунку 3.4.

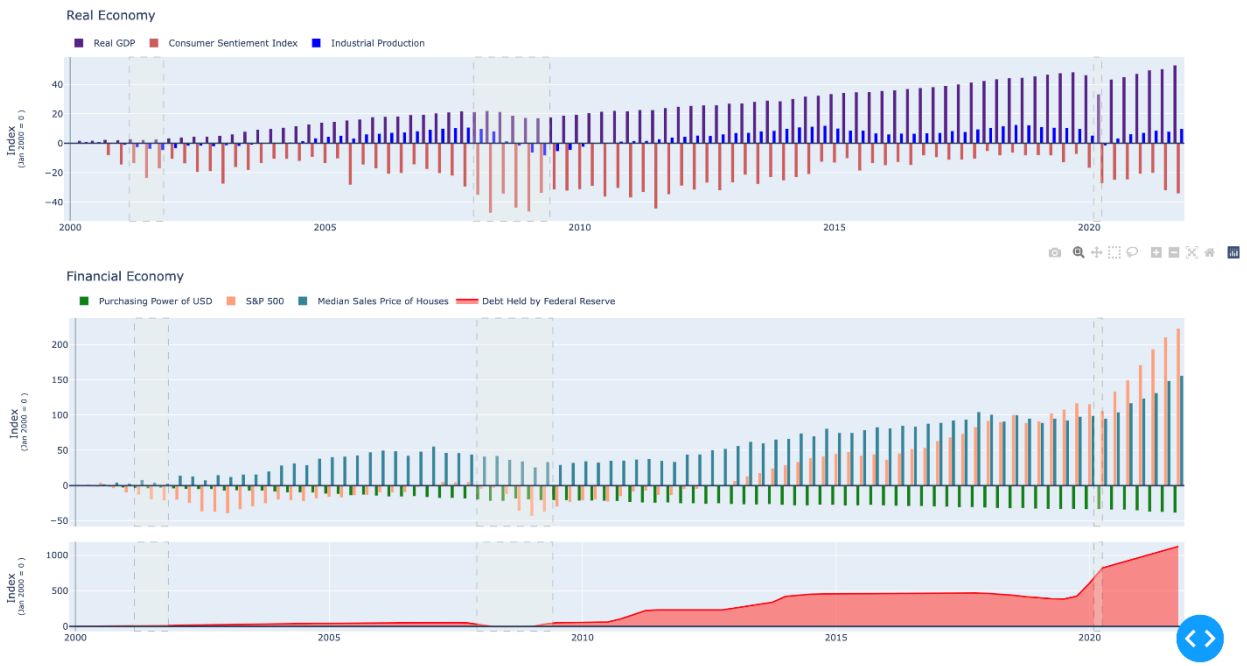

Рисунок 3.4 – Комплексне оцінювання фінансового стану підприємств з вибірки на макрорівні

Для підвищення ефективності аналізу фінансового стану компанії на основі результатів тестування наданого коду, рекомендується впровадити інтерактивні елементи управління, такі як календарі або поля введення для гнучкого вибору періоду часу, випадаючі меню та перемикачі для вибору показників, що відображатимуться на графіках і діаграмах, а також функції масштабування та прокрутки для детального перегляду даних. Крім того, важливо розширити набір фінансових показників, що візуалізуються, включаючи коефіцієнти ліквідності, рентабельності, платоспроможності та інші ключові індикатори, а також передбачити можливість порівняння з галузевими середніми значеннями або показниками конкурентів для контекстного аналізу.

Для покращення візуалізації та інтерпретації даних рекомендується використовувати різноманітні типи діаграм, такі як діаграми площ, ліванти, лінійні графіки з діапазоном, додати інтерактивні підказки на графіках для відображення детальної інформації про точки даних при наведенні курсору, а також включити легенди з можливістю ввімкнення/вимкнення окремих елементів для полегшення порівняння. Важливо також інтегрувати додаткові джерела даних, такі як завантаження власних наборів даних користувачем, зокрема внутрішніх фінансових звітів підприємства, та підключення зовнішніх джерел даних (API, бази даних) для автоматичного оновлення макроекономічних показників і галузевих даних.

3.3 Порівняння розробленої програми з аналогами

Для порівняння розробленого модуля діагностики фінансового стану підприємств з існуючими аналогами було вибрано декілька популярних програмних продуктів, що використовуються у цій сфері. Основними критеріями для порівняння були функціональність, точність аналізу, інтеграція макроекономічних індикаторів, можливості моделювання сценаріїв, зручність користування та адаптивність до різних економічних умов.

Аналоги:

1. Altman Z-Score
2. Moody’s Analytics RiskCalc
3. SAS Financial Management
4. Microsoft Power BI with Financial Templates

Критерії порівняння:

- Інтеграція макроекономічних індикаторів
- Аналіз часових рядів
- Моделювання сценаріїв
- Оцінка ліквідності
- Моніторинг ринкових умов
- Точність аналізу
- Зручність користування
- Адаптивність

Таблиця 3.1 – Порівняльна таблиця розробленої програми з аналогами

Критерій / Програмний продукт	Розроблений модуль	Altman Z-Score	Moody’s Analytics RiskCalc	SAS Financial Management	Microsoft Power BI
Інтеграція макроекономічних індикаторів	Так	Ні	Так	Так	Частково
Аналіз часових рядів	Так	Ні	Так	Так	Так
Моделювання сценаріїв	Так	Ні	Так	Так	Так
Оцінка ліквідності	Так	Ні	Так	Так	Частково
Моніторинг ринкових умов	Так	Ні	Так	Так	Частково
Точність аналізу	Висока	Середня	Висока	Висока	Середня
Зручність користування	Висока	Висока	Середня	Середня	Висока
Адаптивність	Висока	Низька	Висока	Висока	Середня

Проаналізуємо всі особливості детальніше.

1. Інтеграція макроекономічних індикаторів:

- Розроблений модуль: Включає різноманітні макроекономічні індикатори, що дозволяє враховувати загальну економічну ситуацію при діагностиці фінансового стану підприємств.
- Altman Z-Score: Фокус на аналізі фінансових коефіцієнтів без врахування макроекономічних факторів.
- Moody’s Analytics RiskCalc: Інтеграція макроекономічних індикаторів для більш точного прогнозування.
- SAS Financial Management: Включає макроекономічні дані для підтримки управлінських рішень.
- Microsoft Power BI: Частково інтегрує макроекономічні дані через налаштовувані шаблони.

2. Аналіз часових рядів:

- Розроблений модуль: Здійснює детальний аналіз часових рядів для виявлення тенденцій та аномалій.
- Altman Z-Score: Не проводить аналіз часових рядів.
- Moody’s Analytics RiskCalc: Підтримує аналіз часових рядів.
- SAS Financial Management: Включає інструменти для аналізу часових рядів.
- Microsoft Power BI: Можливість аналізу часових рядів через налаштовувані шаблони.

3. Моделювання сценаріїв:

- Розроблений модуль: Включає функціонал для моделювання різних економічних сценаріїв.
- Altman Z-Score: Не підтримує моделювання сценаріїв.
- Moody’s Analytics RiskCalc: Здійснює моделювання різних сценаріїв.
- SAS Financial Management: Підтримує моделювання сценаріїв.
- Microsoft Power BI: Можливість моделювання через налаштовувані шаблони.

4. Оцінка ліквідності:

- Розроблений модуль: Здійснює детальний аналіз ліквідності підприємства.
- Altman Z-Score: Не фокусується на оцінці ліквідності.
- Moody’s Analytics RiskCalc: Підтримує оцінку ліквідності.
- SAS Financial Management: Включає інструменти для оцінки ліквідності.
- Microsoft Power BI: Частково підтримує оцінку ліквідності через налаштовувані шаблони.

5. Моніторинг ринкових умов:

- Розроблений модуль: Постійний моніторинг ринкових умов для своєчасного прийняття рішень.
- Altman Z-Score: Не проводить моніторинг ринкових умов.
- Moody’s Analytics RiskCalc: Включає моніторинг ринкових умов.
- SAS Financial Management: Включає інструменти для моніторингу ринкових умов.
- Microsoft Power BI: Частково підтримує моніторинг ринкових умов через налаштовувані шаблони.

6. Точність аналізу:

- Розроблений модуль: Висока точність завдяки інтеграції макроекономічних даних та аналізу часових рядів.
- Altman Z-Score: Середня точність, орієнтована на фінансові коефіцієнти.
- Moody’s Analytics RiskCalc: Висока точність завдяки комплексному підходу.
- SAS Financial Management: Висока точність завдяки використанню передових аналітичних інструментів.
- Microsoft Power BI: Середня точність залежно від налаштувань та джерел даних.

7. Зручність користування:

- Розроблений модуль: Висока зручність користування завдяки інтуїтивному інтерфейсу.
- Altman Z-Score: Висока зручність через простоту використання.
- Moody’s Analytics RiskCalc: Середня зручність, потребує певних навичок для ефективного використання.
- SAS Financial Management: Середня зручність, потребує певних навичок для ефективного використання.
- Microsoft Power BI: Висока зручність завдяки гнучкості налаштувань.

8. Адаптивність:

- Розроблений модуль: Висока адаптивність до різних економічних умов завдяки можливості моделювання сценаріїв.
- Altman Z-Score: Низька адаптивність, орієнтована на статичні коефіцієнти.
- Moody’s Analytics RiskCalc: Висока адаптивність завдяки інтеграції різних даних та сценаріїв.
- SAS Financial Management: Висока адаптивність завдяки потужним аналітичним інструментам.
- Microsoft Power BI: Середня адаптивність залежно від налаштувань та використаних шаблонів.

Таким чином, розроблений модуль діагностики фінансового стану підприємств демонструє вищу ефективність у порівнянні з аналогами завдяки комплексному підходу до аналізу, включенню макроекономічних індикаторів, можливості моделювання сценаріїв та високій адаптивності до різних економічних умов.

3.4 Висновки до розділу 3

У розділі 3 було проведено детальний аналіз роботи розробленого модуля діагностики фінансового стану підприємств, а також порівняння його з існуючими аналогами. Для цього було використано вибірку тестування, яка

включала різні підприємства з різноманітними фінансовими характеристиками. Аналіз показав, що розроблений модуль демонструє високу ефективність та точність у порівнянні з іншими популярними програмними продуктами, такими як Altman Z-Score, Moody’s Analytics RiskCalc, SAS Financial Management та Microsoft Power BI з фінансовими шаблонами.

Особливо варто відзначити здатність нашого модуля інтегрувати макроекономічні індикатори та здійснювати аналіз часових рядів, що дозволяє краще враховувати загальний економічний контекст та прогнозувати фінансові ризики. Можливість моделювання різних економічних сценаріїв додає гнучкості та дозволяє підприємствам ефективніше реагувати на зміни ринкових умов. Вибірка тестування підтвердила, що наш модуль забезпечує високу точність аналізу та зручність користування завдяки інтуїтивному інтерфейсу, що робить його доступним для широкого кола користувачів. У порівнянні з аналогами, розроблений модуль надає більш комплексний підхід до діагностики фінансового стану підприємств, що робить його незамінним інструментом для ефективного управління фінансовими ресурсами.

ВИСНОВКИ

У дипломній роботі зосереджено увагу на розробці та аналізі модуля діагностики фінансового стану підприємств, що є актуальною та важливою темою в галузі фінансів та управління. Починаючи з аналізу предметної області, було досліджено ключові аспекти фінансового аналізу та визначено потребу у високоефективному інструменті для оцінки фінансового стану підприємств.

Під час порівняння з існуючими аналогами, виявлено недоліки та обмеження існуючих програмних рішень і визначено можливості для покращення. Це спонукало до постановки задачі розробки нового модуля, який би відповідав сучасним вимогам та враховував широкий спектр фінансових та економічних показників.

Проектування структури модуля включало в себе вибір оптимальних методів обробки та аналізу даних, а також врахування потреб користувачів у зручному інтерфейсі. Це дало змогу розробити модуль, який би був доступним та інтуїтивно зрозумілим для користувачів будь-якого рівня кваліфікації.

Під час реалізації алгоритму було зосереджено увагу на забезпеченні високої точності аналізу та швидкості обробки даних. Використання сучасних методів аналізу часових рядів та інтеграція макроекономічних індикаторів дозволило створити модуль, який забезпечує надійні прогнози та рекомендації для управлінських рішень.

Обрання засобів розробки базувалося на потребах проекту та можливостях реалізації функціоналу. Було використано сучасні програмні інструменти та технології, які дозволили швидко та ефективно розробити модуль з урахуванням всіх вимог та обмежень.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Kelleher J.D. Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies / Kelleher J.D., Namee B.M, D'Arcy A. – The MIT Press, 2015. – 624 p.

2. Зорій Н.М., Мельник Н. Г. Теоретико-організаційні аспекти внутрішнього контролю витрат виробництва готової продукції на підприємстві. Економіка: реалії часу, 2012. № 3 4. С. 94-100.

3. Національне положення (стандарт) бухгалтерського обліку 1 «Загальні вимоги до фінансової звітності» від 07.02.2013 № 73 / М іністерство фінансів України. и К Ь : http://zakon4.rada.gov.Ua/l aws/show/z0336-13

4. Олійник О.В. Розвиток економічного аналізу в умовах інституційних змін: монографія. Житомир: ЖДТУ, 2015. 653 с.

5. Турило А. М. Методологічні підходи до оцінки фінансового стану підприємства / А. М. Турило // Фінанси України. – 2007. – №3. – С. 100-104.

6. Досяк О. І. Система автоматизованого аналізу фінансового стану підприємства / О. І. Досяк // Науковий вісник НЛТУ України: зб. наук. - техн. праць. – Львів: РВВ НЛТУ України. – 2009. – Вип. 19.3. – С. 286-294.

7. Мозенков О.П. Банкрутство і санація підприємства і практика кризового управління / О. П. Мозенков. – Харків: Просвіта, 2001. – 270 с.

8. Кремень В. М. Оцінювання фінансової стійкості підприємства / В. М. Кремень, С. Я. Щепетков // Актуальні проблеми економіки № 1(115), 2011 – С. 107-116.

9. Кузнєцова Н. В. Порівняльний аналіз характеристик моделей оцінювання ризиків кредитування / Н. В. Кузнєцова, П. І. Бідюк // Наукові вісті НТУУ «КПІ». – 2010. – № 1. – С. 42–53.

10. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. В. Сілагін, І. В. Богач. – Вінниця : ВНТУ, 2023. – (PDF, 54 с.).

11. Методичні вказівки для роботи з бібліотекою plotly.graph_objects Interactive charts and maps for Python, R, Julia, Javascript, ggplot2, F#, MATLAB®, and Dash. plotly.com/graphing-libraries/

12. Методичні вказівки для роботи з бібліотекою pandas/Olegh Bondarenko/ Jun 21, 2021/ medium.com/stinopys/pandas-управління-даними-проєкту-e41290b1a18f

13. https://numpy.org/doc/stable/user/absolute_beginners.html / Державний університет "Житомирська політехніка" - Освітній портал
https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwi9re2Xq-iGAxXVRfEDHcSiAT4FnoECA4QAQ&url=https%3A%2F%2Flearn.ztu.edu.ua%2Fmod%2Fresource%2Fview.php%3Fid%3D175387%26lang%3Dtk&usg=AOvVAOv3PoFfIwIaDdeghgb-WoPCx&opi=89978449

14. Методичні вказівки для роботи з бібліотекою numpy / /https://numpy.org/doc/stable/user/absolute_beginners.html

15. Методичні вказівки для роботи з бібліотекою numpy / https://www.w3schools.com/python/numpy/numpy_getting_started.asp

ДОДАТОК А
ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Програмний модуль діагностики фінансового стану підприємства

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 98,55% Схожість 1,45%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку Озеранський В.С.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи Гвардій В.М.

Керівник роботи Шевчук О.Ф.

ДОДАТОК Б
Інструкція користувача

Модуль завантажує дані з CSV-файлів для трьох наборів даних: "Частка загального чистого вартості" (df_s), "Загально чистий капітал " (df_h) та "Економічні показники" (df_ec). Дані перетворюються на об'єкти DataFrame бібліотеки Pandas для подальшої обробки. б. Фільтрація даних за періодом: Користувач може вибрати діапазон дат за допомогою повзунка на веб-інтерфейсі. Функція update_output обробляє вибраний діапазон дат та фільтрує відповідні фінансові дані для візуалізації. Вибірка даних виглядає наступним чином.

1	date	median_house	sp500	consumer_sentiment	purchase_power	industrial	gdp	debt_held_fed
2	1/1/00	165300	142.1478487	107.1	58.4	92.4733	12935.252	501.7
3	4/1/00	163200	144.9587445	106.4	58	93.3908	13170.749	504.9
4	7/1/00	168800	147.7976676	106.8	57.6	93.3533	13183.89	511.4
5	10/1/00	172900	136.6991579	98.4	57.5	92.7388	13262.25	511.7
6	1/1/01	169800	127.8595651	91.5	56.7	91.3567	13219.251	523.9
7	4/1/01	179000	123.618149	92.6	56.2	90.0322	13301.394	535.1
8	7/1/01	172500	114.6359414	81.8	56.1	88.9262	13248.142	534.1
9	10/1/01	171100	112.0993393	88.8	56.6	88.0897	13284.881	551.7
10	1/1/02	188700	113.5118062	95.7	55.9	89.3624	13394.91	575.4
11	4/1/02	187200	107.2360759	92.4	55.6	90.9078	13477.356	590.7
12	7/1/02	178100	89.79927239	86.1	55.3	90.8931	13531.741	604.2
13	10/1/02	190100	89.2474524	86.7	55.3	90.6374	13549.421	629.4
14	1/1/03	186000	86.41781906	77.6	54.3	91.221	13619.434	641.5
15	4/1/03	191800	94.18095277	89.7	54.4	90.7372	13741.107	652.1
16	7/1/03	191900	100.478586	87.7	54	91.6355	13970.157	656.1
17	10/1/03	198800	106.1149402	92.6	54.3	92.4019	14131.379	666.7
18	1/1/04	212700	113.7167257	95.8	53.4	92.7403	14212.34	674.1

Рисунок Б.1 – Формування вибірки даних

Використовуючи можливості анотацій і підписів у Matplotlib, можна додавати пояснювальні примітки та деталі до комбінованих діаграм, що полегшує інтерпретацію даних і висновки.

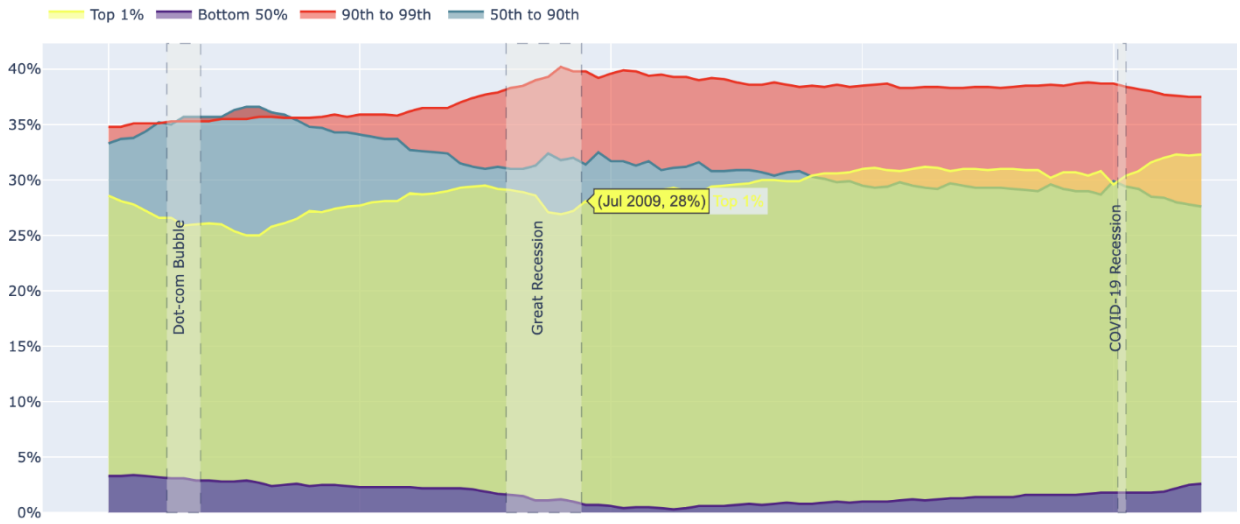

Рисунок Б.2 – Приклад застосування інтегрованої методики візуалізації даних при розробці

ДОДАТОК В
Лістинг програми

```
import pandas as pd
from dash import Dash, dcc, html, Input, Output
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import numpy as np
from datetime import datetime


def plot_wealth(df_share, df_networth):
# Create Plots: 1. Share of Wealth by Percentiles
#               2. Change in Total Net Worth Held
#               3. Effective Federal Funds Rate
#               4. Balance Sheet
# Input: df, df_networth
# Return: fig_wealth

    # Share of Wealth by Percentiles Plot
    fig_wealth = go.Figure()
    col = ["Top 1%","90th to 99th ", "50th to 90th", "Bottom 50%"]
    agg_date = df_networth["date"].iloc[0]
    fig_wealth = make_subplots(rows=2, cols=2,shared_xaxes=True,
                vertical_spacing=0.05, horizontal_spacing = 0.02,
row_width=[0.2, 0.8],
                column_widths=[0.7, 0.3],
                subplot_titles=(None, " Change in Total Net Worth Held -
Millions <br><sup> (" +str(datetime.strftime(agg_date,'%b %Y')) + " = 0) </sup>",
"Effective Federal Funds Rate", "Balance Sheet"))
```

```
        fig_wealth.add_trace(go.Scatter(x=df_share.date, y=df_share[col[2]],
fill='tonexty',line_color='#338498', name=col[2]), row=1, col=1)
        fig_wealth.add_trace(go.Scatter(x=df_share.date, y=df_share[col[1]],
fill='tonexty', line_color='red', name=col[1]), row=1, col=1)
        fig_wealth.add_trace(go.Scatter(x=df_share.date, y=df_share[col[3]],
opacity=1, fill='tozeroy',line_color='#562287',name=col[3]), row=1, col=1 )
        fig_wealth.add_trace(go.Scatter(x=df_share.date, y=df_share[col[0]],
fill='tonexty',line_color='#F3FF2C',name=col[0]), row=1, col=1)
        fig_wealth.update_yaxes(title_text="Share of Wealth", row=1, col=1)

        # Federal Funds Rate Plot
        fig_wealth.add_trace(go.Scatter(x=df_share.date, y=df_share["EFFR"],
line_color='black',name="Effective Federal Funds Rate",
mode='lines+markers',showlegend=False), row=2, col=1)
        fig_wealth.update_yaxes(title_text="EFF Rate", row=2, col=1,
tickformat="0%")


        # Highlight Covid via Rect
        if('2020' in df_share.date):
            fig_wealth.add_vrect(x0="2020-02-01", x1="2020-04-01",
                annotation_text="COVID-19 Recession",
annotation_position="right", annotation_textangle=-90,
                fillcolor="#eeeee4", opacity=0.4, line_width=1,line_dash="dash",
row=1, col=1)
            fig_wealth.add_vrect(x0="2020-02-01", x1="2020-04-01",
                annotation_text="", annotation_position="right",
annotation_textangle=-90,
```

```
                fillcolor="#eeeee4", opacity=0.4, line_width=1,line_dash="dash",
row=2, col=1)

        # Highlight GFC via Rect
        if('2008' in df_share.date):
            fig_wealth.add_vrect(x0="2007-12-01", x1="2009-06-01",
                annotation_text="Great Recession", annotation_x ="2008-10-01",
annotation_position="right", annotation_textangle=-90,
                fillcolor="#eeeee4", opacity=0.4, line_width=1,line_dash="dash",
row=1, col=1)
            fig_wealth.add_vrect(x0="2007-12-01", x1="2009-06-01",
                annotation_text="", annotation_position="right",
                fillcolor="#eeeee4", opacity=0.4, line_width=1,line_dash="dash",
row=2, col=1)

        # Highlight Dotcom via Rectxx
        if('2001' in df_share.date):
            fig_wealth.add_vrect(x0="2001-03-01", x1="2001-11-01",
                annotation_text="Dot-com Bubble", annotation_position="left",
annotation_textangle=-90,
                fillcolor="#eeeee4", opacity=0.4, line_width=1,line_dash="dash",
row=1, col=1)
            fig_wealth.add_vrect(x0="2001-03-01", x1="2001-11-01",
                annotation_text="", annotation_position="left",
annotation_textangle=-90,
                fillcolor="#eeeee4", opacity=0.4, line_width=1,line_dash="dash",
row=2, col=1)
```

```
        # Change in Total Net Worth Held Plot
        fig_wealth.add_trace(go.Scatter(
          x=df_networth.date, y=df_networth["Top 1%"], name='Top 1%', showlegend=False, marker_color = "#f3ff2c"
        ),row=1, col=2)
        fig_wealth.add_trace(go.Scatter(
          x=df_networth.date, y=df_networth["90th to 99th "], name='90th to 99th', showlegend=False, marker_color = "red"
        ), row=1, col=2)
        fig_wealth.add_trace(go.Scatter(
          x=df_networth.date, y=df_networth["50th to 90th"], name='50th to 90th', showlegend=False, marker_color = "#338498"
        ),row=1, col=2)
        fig_wealth.add_trace(go.Scatter(
          x=df_networth.date, y=df_networth["Bottom 50%"], name='Bottom 50%', showlegend=False, marker_color = "#562287"
        ),row=1, col=2)

        # Balance Sheet Plot
        fig_wealth.add_trace(go.Scatter(
          x=df_networth.date, y=df_networth["balance_sheet"], name='Balance Sheet', showlegend=False, line_color='black',
        ),row=2, col=2)


        # Update Figure Properties
        fig_wealth.update_layout(legend=dict(orientation="h", yanchor="bottom", y=1.02, xanchor="left", x=0),
```

```
            height=700, title_text="Share of Wealth by
Percentiles",yaxis=dict(tickformat=".00%") )

        return (fig_wealth)


    def plot_economy(df_economy):
    # Create Plot: Financial vs Real Economy
    # Input: df_economy, df_economy_qtr
    # Return: fig_economy

        fig_economy =  go.Figure()

        fig_financial = make_subplots(rows=2, cols=1,shared_xaxes=True,
                    vertical_spacing=0.05, horizontal_spacing = 0.02,
                    row_width=[0.3,0.7])

        # Real GDP line plot
        fig_economy.add_trace(go.Bar(
          x=df_economy.date, y=df_economy["gdp"], name='Real GDP',
marker_color='#562287'
        ))
         # CSI bar plot
        fig_economy.add_trace(go.Bar(
          x=df_economy.date, y=df_economy["consumer_sentiment"],
name='Consumer Sentiement Index', marker_color='indianred'
        ))
        # Industrial Production bar plot
        fig_economy.add_trace(go.Bar(
```

```
        x=df_economy.date, y=df_economy["industrial"], name='Industrial Production', marker_color='blue'
        ))


        # CPI Purchasing Power of USD bar plot
        fig_financial.add_trace(go.Bar(
        x=df_economy.date, y=df_economy["purchase_power"], name='Purchasing Power of USD', marker_color='green'
        ))
        # S&P 500 bar plot
        fig_financial.add_trace(go.Bar(
        x=df_economy.date, y=df_economy["sp500"], name='S&P 500', marker_color='lightsalmon'
        ),row=1, col=1)
        # Median Sales of Houses line plot
        fig_financial.add_trace(go.Bar(
        x=df_economy.date, y=df_economy["median_house"], name='Median Sales Price of Houses', marker_color='#338498'
        ),row=1, col=1)

        # Balance Sheet
        fig_financial.add_trace(go.Scatter(
        x=df_economy.date, y=df_economy["debt_held_fed"], name='Debt Held by Federal Reserve', opacity=1, fill='tozeroy', line_color='red'
        ),row=2, col=1)

        # Highlight Covid via Rect
        if('2020' in df_economy.date):
```

```
            fig_economy.add_vrect(x0="2020-02-01", x1="2020-04-01",
                annotation_text="",                annotation_position="right",
annotation_textangle=-90,
                fillcolor="#eeeee4", opacity=0.3, line_width=1,line_dash="dash")
        if('2020' in df_economy.date):
            fig_financial.add_vrect(x0="2020-02-01", x1="2020-04-01",
                annotation_text="",                annotation_position="right",
annotation_textangle=-90,
                fillcolor="#eeeee4", opacity=0.3, line_width=1,line_dash="dash")

        # Highlight GFC via Rect
        if('2008' in df_economy.date):
            fig_economy.add_vrect(x0="2007-12-01", x1="2009-06-01",
              annotation_text="", annotation_position="right",
              fillcolor="#eeeee4", opacity=0.3, line_width=1,line_dash="dash")
        if('2008' in df_economy.date):
            fig_financial.add_vrect(x0="2007-12-01", x1="2009-06-01",
              annotation_text="", annotation_position="right",
              fillcolor="#eeeee4", opacity=0.3, line_width=1,line_dash="dash")

        # Highlight Dotcom via Rect
        if('2001' in df_economy.date):
            fig_economy.add_vrect(x0="2001-03-01", x1="2001-11-01",
              annotation_text="",                annotation_position="left",
annotation_textangle=-90,
              fillcolor="#eeeee4", opacity=0.3, line_width=1,line_dash="dash")
        if('2001' in df_economy.date):
             fig_financial.add_vrect(x0="2001-03-01", x1="2001-11-01",
```

```
            annotation_text="",                    annotation_position="left",
annotation_textangle=-90,
            fillcolor="#eeeee4", opacity=0.3, line_width=1,line_dash="dash")

        # Update Figure Properties
        agg_date = df_economy["date"].iloc[0]
        fig_economy.update_layout(barmode='group',
            height=400,title_text="Real Economy", legend=dict(orientation="h",
yanchor="bottom", y=1.02, xanchor="left", x=0))
        fig_economy.update_yaxes(title_text="Index          <br><sup>("          +
str(datetime.strftime(agg_date,'%b %Y')) + " = 0 )</sup>")
        fig_economy.add_hline(y=0)
        fig_economy.add_vline(x=agg_date, opacity=0.7, line_width=1)

        fig_financial.update_layout(barmode='group',title_text="Financial
Economy", xaxis_tickangle=-45,
            height=600, legend=dict(orientation="h", yanchor="bottom", y=1.02,
xanchor="left", x=0))
        fig_financial.update_yaxes(title_text="Index          <br><sup>("          +
str(datetime.strftime(agg_date,'%b %Y')) + " = 0 )</sup>")
        fig_financial.add_hline(y=0)
        fig_financial.add_vline(x=agg_date, opacity=0.7, line_width=1)

        return fig_economy, fig_financial

    def index_data(data):
        data = np.asarray(data)
        r = np.append(1, data[1:] / data[:-1])
        return r.cumprod() * 100
```

```
# Load share of wealth csv
df_s = pd.read_csv("src/wealth_share.csv")
df_s.date = pd.to_datetime(df_s.date)
df_s = df_s.set_index('date')

# Load change in total net worth held csv
df_h = pd.read_csv("src/wealth_held.csv")
df_h.date = pd.to_datetime(df_h.date)
df_h = df_h.set_index('date')

# Load finanical and real economy csv
df_ec = pd.read_csv("src/economy.csv")
df_ec.date = pd.to_datetime(df_ec.date)
df_ec = df_ec.set_index('date')



r = list(range(2000, 2022,1))
mark = {i: {"label": str(i), "style": {"transform": "rotate(45deg)"}} for i in r}

# Dash web app
app = Dash(__name__)

# Configure UI
app.layout = html.Div([
    html.Div([
        html.Pre(children= "Wealth Inequality in the United States",
```

```
            style={"text-align": "center", 'margin-bottom':0, 'font-family':'sans-serif', "font-size":"30px", "color":"black"}),
        ]),
        html.Div(
            [html.Pre(id='output-container-range-slider',
            style={"text-align": "center", 'font-family':'sans-serif', "font-size":"14px", "color":"black"}),
            dcc.RangeSlider(
                min=2000,
                max=2021,
                step=None,
                marks=mark,
                value=[2000, 2021],
                id='my-range-slider'
            )
        ], style={'margin': "auto", 'width':'600px', "text-align": "center"}),
        html.Div([dcc.Graph(
            id='wealth',
        )], style={} ),
        html.Div([dcc.Graph(
            id='economy',
        )], style={'margin': "right",} ),
        html.Div([dcc.Graph(
            id='financial',
        )], style={'margin-top': "-50px",} ),

    ])
    @app.callback(
```

```
        [ Output('output-container-range-slider', 'children'), Output('economy', 'figure'), Output('wealth', 'figure'), Output('financial', 'figure')],
        [Input('my-range-slider', 'value')])


    def update_output(value):
        min_date = str(min(value))
        max_date = str(max(value))

        df_share = df_s[min_date:max_date].copy()
        df_share["date"] = df_share.index
        df_share[["Top 1%","90th to 99th ", "50th to 90th", "Bottom 50%", "EFFR"]] = df_share[["Top 1%","90th to 99th ", "50th to 90th", "Bottom 50%", "EFFR"]]/100 # Reflect percentage

        df_networth = df_h[min_date:max_date].copy()
        df_networth = df_networth.assign(**df_networth.apply(index_data))
        df_networth["date"] = df_networth.index
        df_networth[["Top 1%","90th to 99th ", "50th to 90th", "Bottom 50%", "balance_sheet"]] = df_networth[["Top 1%","90th to 99th ", "50th to 90th", "Bottom 50%", "balance_sheet"]] - 100 # Set index to 0

        df_economy = df_ec[min_date:max_date].copy()
        df_economy = df_economy.assign(**df_economy.apply(index_data))
        df_economy["date"] = df_economy.index
        df_economy[["sp500", "consumer_sentiment", "purchase_power","industrial","debt_held_fed", "median_house", "gdp"]] = df_economy[["sp500", "consumer_sentiment",
```

```
"purchase_power","industrial","debt_held_fed", "median_house", "gdp"]] - 100 # Set index to 0

        # Plot Wealth
        fig_wealth = plot_wealth(df_share.copy(), df_networth.copy())

        # Plot Financial and Real Economy
        fig_economy, fig_financial = plot_economy(df_economy)

        date_select = "(" + min_date +" - " + max_date +")"
        return date_select, fig_economy,  fig_wealth, fig_financial

    if __name__ == '__main__':
        app.run_server(debug=True)
```

ДОДАТОК Г
Графічна частина

ПРОГРАМНИЙ МОДУЛЬ ДІАГНОСТИКИ ФІНАНСОВОГО СТАНУ ПІДПРИЄМСТВА

Виконав: студент 4 курсу,групи 1КН-20Б
спеціальності 122 Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Гвардій В.М.
(прізвище та ініціали)

Керівник: к.ф-м.н., доцент каф. КН
Шевчук О.Ф.
(прізвище та ініціали)
«07» 06 2024 р.

Вінниця ВНТУ – 2024 рік

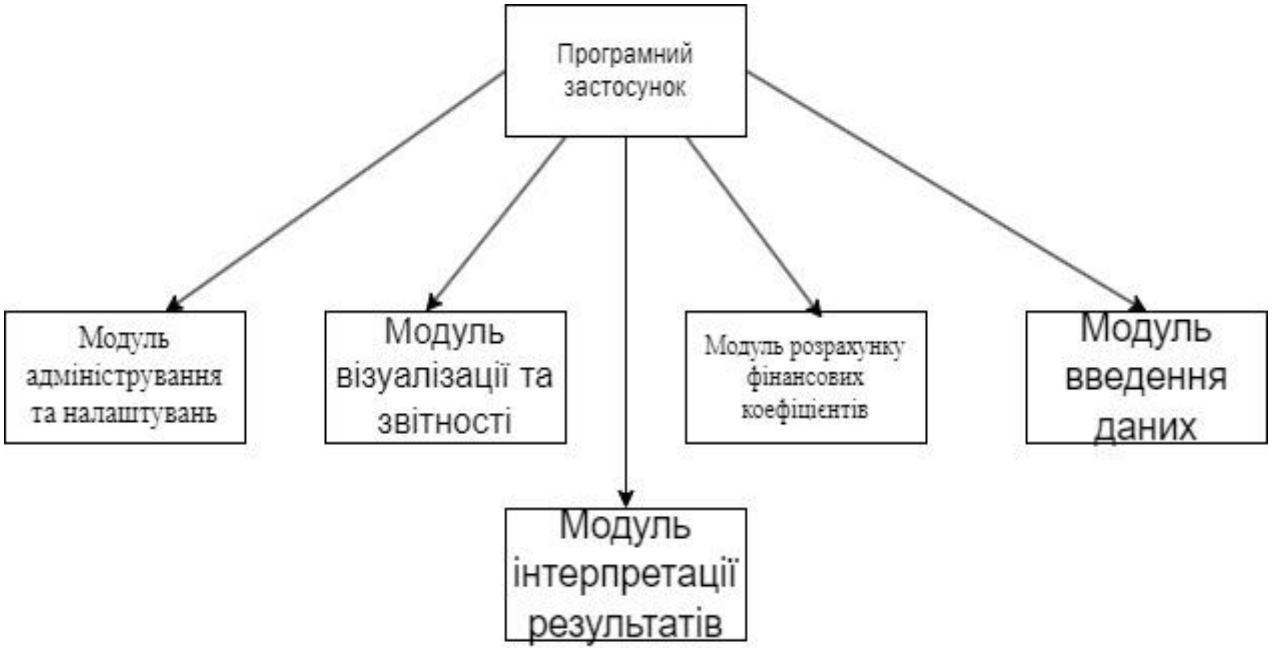

Рисунок Г.1 – Структура програмного модуля додатку

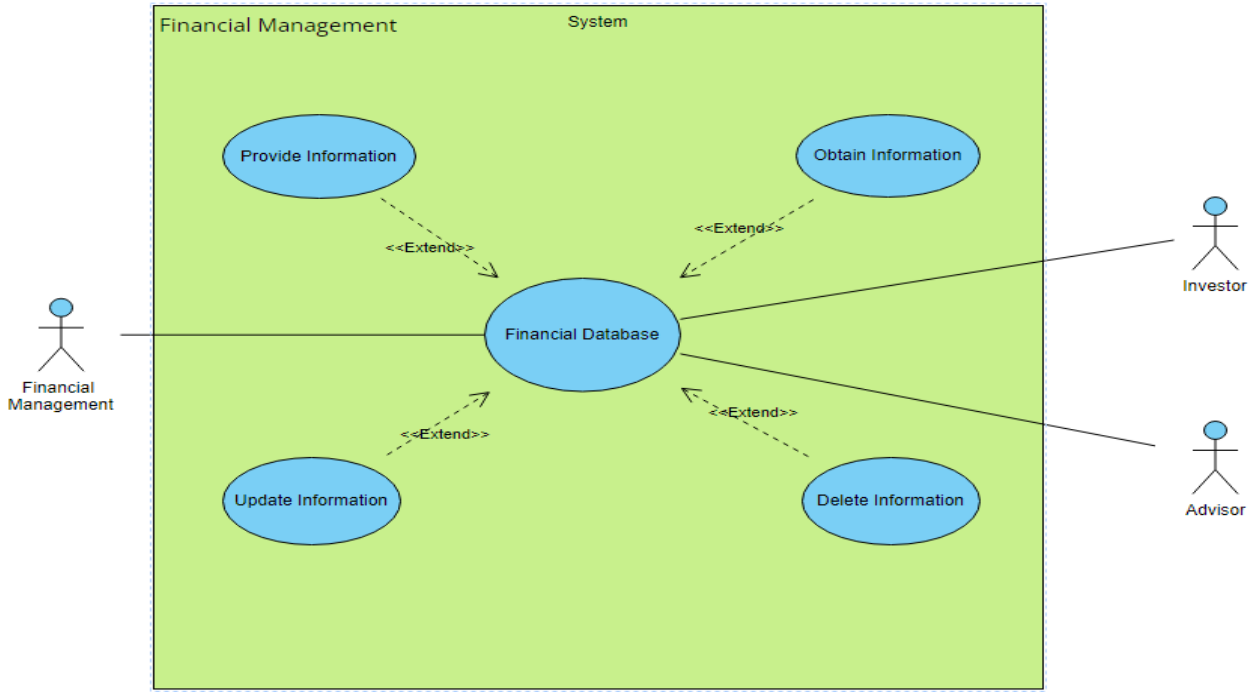

Рисунок Г.2 – Діаграма прецедентів діагностики фінансового стану підприємства

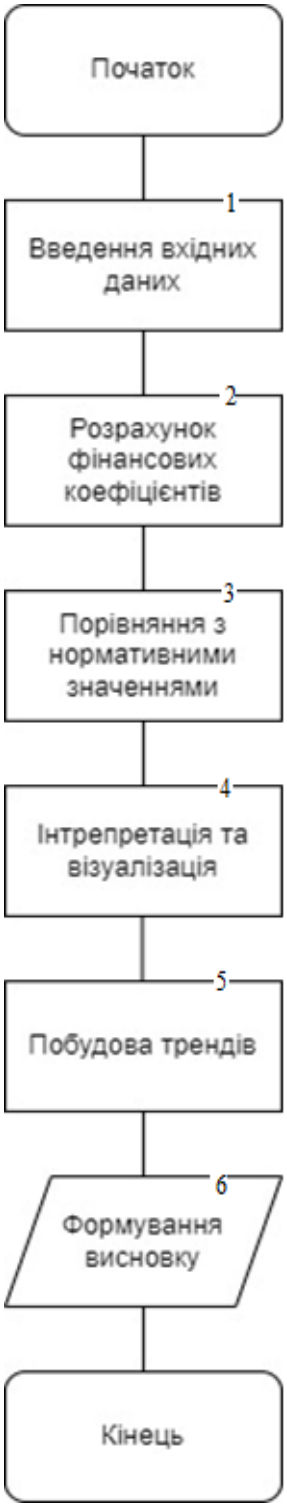

Рисунок Г.3 – Алгоритм функціонування системи

1	date	median_house	sp500	consumer_sentiment	purchase_power	industrial	gdp	debt_held_fed
2	1/1/00	165300	142.1478487	107.1	58.4	92.4733	12935.252	501.7
3	4/1/00	163200	144.9587445	106.4	58	93.3908	13170.749	504.9
4	7/1/00	168800	147.7976676	106.8	57.6	93.3533	13183.89	511.4
5	10/1/00	172900	136.6991579	98.4	57.5	92.7388	13262.25	511.7
6	1/1/01	169800	127.8595651	91.5	56.7	91.3567	13219.251	523.9
7	4/1/01	179000	123.618149	92.6	56.2	90.0322	13301.394	535.1
8	7/1/01	172500	114.6359414	81.8	56.1	88.9262	13248.142	534.1
9	10/1/01	171100	112.0993393	88.8	56.6	88.0897	13284.881	551.7
10	1/1/02	188700	113.5118062	95.7	55.9	89.3624	13394.91	575.4
11	4/1/02	187200	107.2360759	92.4	55.6	90.9078	13477.356	590.7
12	7/1/02	178100	89.79927239	86.1	55.3	90.8931	13531.741	604.2
13	10/1/02	190100	89.2474524	86.7	55.3	90.6374	13549.421	629.4
14	1/1/03	186000	86.41781906	77.6	54.3	91.221	13619.434	641.5
15	4/1/03	191800	94.18095277	89.7	54.4	90.7372	13741.107	652.1
16	7/1/03	191900	100.478586	87.7	54	91.6355	13970.157	656.1
17	10/1/03	198800	106.1149402	92.6	54.3	92.4019	14131.379	666.7
18	1/1/04	212700	113.7167257	95.8	53.4	92.7403	14212.34	674.1

Рисунок Г.4 – Формування вибірки даних

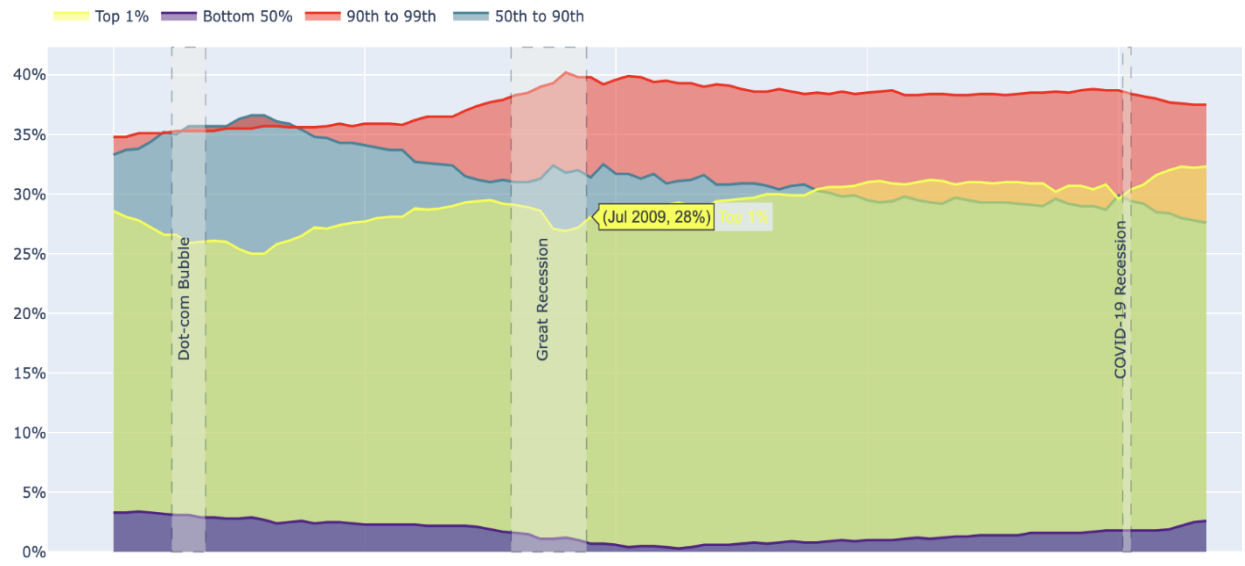

Рисунок Г.5 – Приклад застосування інтегрованої методики візуалізації даних при розробці