Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

“Інформаційна технологія аналізу та передбачення стану хворих на гепатит”

Виконав: студент 2 курсу, групи 2ІСТ-24м
спеціальності 126 – «Інформаційні системи та технології»
_____ Денис ПОЛІЩУК
Керівник: к.т.н., ас. каф. САІТ
_____ Ігор ШТЕЛЬМАХ
«29» 11 _____ 2025 р.
Опонент: к.т.н., доц. каф. КН
_____ Володимир ОЗЕРАНСЬКИЙ
«03» 12 _____ 2025 р.

Допущено до захисту
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«28» 11 _____ 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – другий (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 126 Інформаційні системи та технології
Освітньо-професійна програма – Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ

Завідувач кафедри САІТ

д.т.н., проф. Віталій МОКІН

«25» 09 2025 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Поліщуку Денису Миколайовичу

1. Тема роботи: «Інформаційна технологія аналізу та передбачення стану хворих на гепатит», керівник роботи: Ігор ШТЕЛЬМАХ, к.т.н., ас. каф. САІТ, затверджено наказом ВНТУ від "24" 09 2025 року №313
2. Строк подання студентом роботи «28» 11 2025 року
3. Вихідні дані до роботи:
   Kaggle Dataset «Hepatitis C Prediction Dataset» https://www.kaggle.com/datasets/fedesoriano/hepatitis-c-dataset.
4. Зміст текстовової частини:
   - Загальна характеристика об'єкту дослідження.
   - Постановка задачі та вибір оптимальних інформаційних технологій для її розв'язання.
   - Розвідувальний аналіз даних та побудова моделей.
   - Розроблення інформаційної технології та її застосування на реальних даних.
5. Перелік ілюстративного матеріалу:
   - Графік розподілу значень.
   - Розподіл даних за віком.
   - Гістограма розподілу за віком.
   - Кореляційна матриця.
   - Матриці плутанини.
   - Результати роботи моделей.

6. Консультанти розділів МКР

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийн.
3	Крижановський Євгеній, доцент каф. САІТ	05.10.2025	25.10.2025
4	Олександр ЛЕСЬКО, к.е.н., проф. каф. ЕПВМ	10.11.2025	15.11.2025

7. Дата видачі завдання «25» 09 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	15.09.2025	25.09.2025	виконано
2	Вибір оптимальних інформаційних технологій	25.09.2025	05.10.2025	виконано
3	Вибір мови програмування та середовища розробки	05.10.2025	25.10.2025	виконано
4	Розробка інформаційної технології	25.10.2025	05.11.2025	виконано
5	Експериментальна перевірка роботи програми	05.11.2025	10.11.2025	виконано
6	Економічна частина	10.11.2025	15.11.2025	виконано
7	Оформлення матеріалів до захисту МКР	15.11.2025	25.11.2025	виконано

Студент _____ Денис ПОЛІЩУК

Керівник роботи _____ Ігор ШТЕЛЬМАХ

АНОТАЦІЯ

УДК 004.09:616.36-002

Поліщук Д.М. Інформаційна технологія аналізу та передбачення стану хворих на гепатит. Магістерська кваліфікаційна робота зі спеціальності 126 – інформаційні системи та технології, освітньо-професійна програма – інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2025. 122 с.

На укр. мові. Бібліогр.: 20 назв; рис.: 40; табл.: 13.

У магістерській кваліфікаційній роботі проаналізовано проблематику аналізу та передбачення стану хворих на гепатит, яка є актуальною у сфері медицини. Досліджено методи та підходи до аналізу та передбачення стану хворих на гепатит, що дозволяють підвищити точність передбачення на основі медичних даних по пацієнтах.

Запропоновані технології машинного навчання, на основі яких побудовано моделі класифікації з використанням ансамблевого підходу, який підвищив точність передбачення стану хворих на гепатит. Результати досліджень можуть бути використані для розробки сучасних інструментів обробки зображень, які сприятимуть ефективному управлінню міськими ресурсами. Об’єкт досліджень – процес автоматизованої класифікації медичних даних пацієнтів, хворих на гепатит, з використанням відповідного набору даних. Галузь застосування – системи передбачення стану хворих.

Ілюстративна частина складається з 6 плакатів.

В економічній частині було проведено оцінювання наукового ефекту розробки та розрахунок витрат на здійснення науково-дослідної роботи.

Ключові слова: машинне навчання, інтелектуальні технології, передбачення, розвідувальний аналіз даних, гепатит, класифікація.

ABSTRACT

Polishchuk D.M. Information technology for analysis and prediction of the condition of patients with hepatitis. Master's Qualification Thesis in the specialty 126 – Information Systems and Technologies, Educational and Professional Program – Information Technologies for Data and Image Analysis. Vinnytsia: VNTU, 2025. 122 pages.

In Ukrainian. Bibliography: 20 references; figures: 40; tables: 13.

The master's qualification work focuses on the issues of analysis and prediction of the condition of patients with hepatitis, which is relevant in the field of medicine. Methods and approaches to the analysis and prediction of the condition of patients with hepatitis are studied, which allow to increase the accuracy of prediction based on medical data on patients.

Machine learning technologies are proposed, on the basis of which classification models are built using the assembly approach, which increased the accuracy of prediction of the condition of patients with hepatitis. The results of the research can be used to develop modern image processing tools that will contribute to the effective management of urban resources. The object of the research is the process of automated classification of medical data of patients with hepatitis, using the appropriate data set. The field of application is patient condition prediction systems.

The illustrative part consists of 6 posters.

In the economic part, an assessment of the scientific effect of the development and a calculation of the costs of carrying out scientific and research work were carried out.

Keywords: machine learning, intelligent technologies, prediction, intelligence data analysis, hepatitis, classification.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасний світ стрімко рухається в напрямку цифровізації, і розвиток інформаційних технологій відіграє в цьому ключову роль. Збільшення обсягів даних в інформаційних мережах створює нові можливості для аналізу, класифікації та передбачення. Завдяки сучасним IT-рішенням можна виявляти закономірності в даних, що дозволяє ефективніше приймати рішення та оптимізувати функціонування різних систем. Однією з актуальних медичних проблем сьогодення є гепатит — захворювання, яке потребує своєчасного виявлення. Використання технологій аналізу медичних даних дає змогу значно покращити діагностику та підвищити шанси на ефективне лікування, особливо якщо захворювання виявлено на ранній стадії.

Мета і задачі дослідження. Метою роботи є підвищення точності передбачення типу гепатиту у пацієнтів шляхом застосування інформаційних технологій для обробки та аналізу їх медичних даних.

Для досягнення цієї мети необхідно виконати такі задачі:

- провести аналіз медичної предметної області з акцентом на проблематику гепатиту;
- здійснити обробку й аналіз наявних медичних даних;
- вибрати оптимальні технології та програмні середовища для розробки;
- створити та протестувати моделі класифікації на основі відібраних ознак.

Об’єкт дослідження – процес автоматизованого передбачення медичних даних пацієнтів, хворих на гепатит, з використанням відповідного набору даних.

Предмет дослідження – методи інформаційних технологій, що застосовуються для передбачення типу гепатиту на основі медичних показників пацієнтів.

Методи дослідження. У дослідженнях використовувались методи статистичного аналізу даних, методи кореляційного аналізу даних, методи тюнінгу моделей.

Новизна одержаних результатів.

Подальшого розвитку набула інформаційна технологія аналізу та передбачення стану хворих на гепатит, яка, на відміну від існуючих, використовує моделі Random Forest та Gradient Boosting для передбачення типу гепатиту на основі медичних показників пацієнтів. Це забезпечує високу точність передбачення.

Практична цінність одержаних результатів.

Результати магістерської кваліфікаційної роботи можуть бути застосовані для передбачення типу гепатиту на основі медичних показників пацієнтів. Використання розробленої технології дозволить значно скоротити час і ресурси, необхідні для підготовки прогнозів та аналізу даних захворюваності на гепатит.

Апробація результатів магістерської кваліфікаційної роботи.

За результатами роботи подано до друку тези на «LV Всеукраїнську науково-технічну конференцію підрозділів Вінницького національного технічного університету (ВНТКП ВНТУ) (Вінниця, 2025-2026 рр.).

Публікації результатів магістерської кваліфікаційної роботи.

Подано до друку тези на «LV Всеукраїнську науково-технічну конференцію підрозділів Вінницького національного технічного університету (ВНТКП ВНТУ) (Вінниця, 2025-2026 рр.). [1]

1 ХАРАКТЕРИСТИКА ПОСТАВЛЕНОЇ ЗАДАЧІ ТА ВИБІР ОПТИМАЛЬНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ЇЇ РОЗВ'ЯЗАННЯ

1.1 Опис об'єкта досліджень та постановка задачі

Запалення печінки, відоме як гепатит, може розвиватися внаслідок різноманітних факторів: від вірусних патогенів до впливу токсичних сполук (зокрема спиртних напоїв чи медикаментів), аутоімунних реакцій або метаболічних збоїв. Серед найпоширеніших форм виділяють вірусні варіанти захворювання типів A, B, C, D і E. Кожна форма характеризується унікальним механізмом зараження, характером розвитку, рівнем ризику для здоров'я та терапевтичними підходами [1].

Форма типу A зазвичай поширюється через контаміновані продукти харчування чи питну воду за фекально-оральним маршрутом. Така варіація найчастіше реєструється у регіонах з недостатнім санітарним забезпеченням. Характеризується гострим перебігом без трансформації у хронічний стан. Переважна більшість пацієнтів досягає повного відновлення без негативних наслідків.

Тип B являє собою серйознішу загрозу, передаючись через біологічні рідини, статеві контакти або вертикальним шляхом під час народження дитини. Може проявлятися як у гострій, так і хронічній формі. Хронічний варіант несе ризик розвитку тяжких патологій печінки, включаючи цироз та онкологічні новоутворення. Доступна профілактична імунізація забезпечує надійний захист від інфікування.

Варіант C також розповсюджується через контакт з інфікованою кров'ю, особливо при використанні нестерильного медичного обладнання, трансфузії крові або при ін'єкційному вживанні наркотичних речовин. Ця форма часто непомітно прогресує до хронічної стадії, перебігаючи латентно протягом тривалого періоду, але поступово спричиняючи серйозні пошкодження органу. Незважаючи на відсутність профілактичної вакцини, сучасні противірусні

засоби демонструють високу ефективність у досягненні повного одужання у більшості випадків.

Тип D розвивається виключно у осіб, які вже мають інфекцію типу B, оскільки для свого життєвого циклу потребує присутності вірусу HBV. Коінфекція цими двома патогенами зазвичай характеризується більш важким клінічним перебігом. Тип E, аналогічно до варіанту A, поширюється через забруднені водні джерела і типовий для територій з незадовільними санітарними умовами. Зазвичай має гострий самообмежувальний перебіг, проте становить особливу небезпеку для вагітних.

Рання діагностика має критичне значення, адже на початкових етапах захворювання часто перебігає безсимптомно або з мінімальними проявами, які легко пропустити. Чим триваліший період латентного перебування вірусу в організмі, тим більші незворотні зміни він може спричинити у тканинах печінки, включаючи фіброзні процеси, циротичні трансформації або розвиток гепатоцелюлярної карциноми — однієї з найнебезпечніших форм онкологічних захворювань печінки [2].

Для виявлення патології застосовують комплексний підхід лабораторної діагностики: імунологічні дослідження для ідентифікації специфічних імуноглобулінів або вірусних протеїнів, молекулярно-генетичні методи для детекції нуклеїнових кислот патогену, а також біохімічне профілювання крові для моніторингу функціонального стану печінки (включаючи концентрацію трансаміназ ALT, AST та білірубіну). Важливе місце займають також апаратні методи дослідження, як-от ультразвукова діагностика або еластометрія, що дозволяють визначити морфологічні характеристики та щільність паренхіми органу.

Медична наука досягла вражаючих успіхів у терапії гепатиту, особливо типу C, який донедавна вважався практично невиліковною хворобою. Завдяки розробці прямих противірусних агентів (DAAs), значна частина пацієнтів із хронічною формою може досягти повної елімінації вірусу протягом декількох місяців лікування з мінімальними небажаними реакціями. Терапія типу B складніша через здатність вірусу інтегруватися у генетичний матеріал

гепатоцитів, що ускладнює його повне видалення. Втім, противірусна терапія дозволяє ефективно контролювати реплікацію вірусу та попереджати розвиток ускладнень.

Особливої уваги потребує превентивна стратегія. Імунопрофілактика типів А і В є дієвим засобом запобігання інфікуванню, особливо для медперсоналу, дитячого контингенту та осіб із групи ризику. Дотримання гігієнічних стандартів, застосування стерильного інструментарію, захищена статева активність та уникнення ін'єкційного наркоспоживання суттєво мінімізують імовірність зараження [3-4].

Проблема гепатиту виходить за межі індивідуального здоров'я, становлячи загрозу епідеміологічного характеру. У численних державах функціонують національні стратегії боротьби з вірусними гепатитами, що включають масовий скринінг, безоплатне медикаментозне забезпечення та превентивні ініціативи. Всесвітня організація охорони здоров'я визначила амбітну ціль елімінувати гепатит як епідеміологічну загрозу до 2030 року, що вимагає консолідованих зусиль медичної спільноти, державних структур і самих пацієнтів.

У контексті активної цифровізації медичної галузі особливої актуальності набувають інтелектуальні аналітичні платформи, які дозволяють не тільки констатувати наявність гепатиту, а й формувати прогностичні моделі щодо його типологічної належності, стадії розвитку, ризику прогресування та ефективності специфічних терапевтичних втручань. Це забезпечує перехід до більш прецизійного, індивідуалізованого підходу в діагностичних та лікувальних процесах, що критично важливо при хронічних формах захворювання.

Сучасні медичні інформаційні платформи здатні консолідувати масивні обсяги різнорідної інформації — від результатів клінічних досліджень до історії хвороби пацієнтів, генетичних предиспозицій, демографічних параметрів та соціальних чинників, які можуть впливати на патогенез. Застосовуючи алгоритми машинного навчання, такі системи виявляють латентні кореляції між параметрами, недоступні при конвенційному аналізі. Наприклад, можлива автоматична стратифікація пацієнтів за рівнем ризику прогресування патології

на основі клінічних маркерів або передбачення найвірогіднішого типу патогену навіть до отримання остаточного лабораторного підтвердження.

Подібні технології особливо цінні в умовах обмеженого ресурсного забезпечення — коли клініцист не може провести повномасштабну діагностику всім пацієнтам, але отримує попередні висновки через аналітичну систему. Це також оптимізує швидкість прийняття клінічних рішень, що має життєво важливе значення при гострих або загрозливих станах.

Іншим перспективним напрямом є впровадження електронної медичної документації та хмарних технологій, що забезпечують збереження та обмін пацієнтськими даними між різними лікувальними закладами та фахівцями. Така архітектура гарантує неперервність моніторингу, що надзвичайно важливо для спостереження за хронічними формами гепатиту, коли пацієнт може отримувати медичну допомогу в різних установах або навіть у різних державах.

Не менш важливою є просвітницька діяльність серед громадськості. Дисемінація науково обґрунтованої інформації про механізми трансмісії вірусу, необхідність імунізації та важливість регулярного скринінгу може знизити рівень захворюваності. У цьому напрямку також корисні інформаційні технології — мобільні застосунки, віддалені медичні консультації, інтерактивні освітні ресурси, що роблять знання доступними та зручними для засвоєння [4].

Для розробки інформаційної технології аналізу та передбачення стану хворих на гепатит необхідно розв’язати такі задачі

- здійснити вибір оптимальних інформаційних технологій для її розв’язання;
- здійснити розвідувальний аналіз даних;
- побудувати моделі передбачення стану хворих на гепатит;
- розробити інформаційну технологію аналізу та передбачення стану хворих на гепатит та її застосування на реальних даних.

1.2 Аналіз інформаційних технологій

Машинне навчання є складовою частиною штучного інтелекту, яка дає змогу комп'ютерним системам самостійно здобувати знання з наявних даних і приймати рішення без необхідності жорсткого програмування. Суть цього підходу полягає в тому, що програма аналізує вхідні дані, виявляє приховані закономірності та створює модель, здатну прогнозувати або класифікувати нову інформацію. Наприклад, система може навчитися розпізнавати обличчя, розуміти усне мовлення або прогнозувати зміни валютного курсу, якщо їй надати достатню кількість прикладів для навчання [5].

Основою будь-якої моделі машинного навчання є дані — чим вони численніші та якісніші, тим вищою буде точність результату. Розрізняють кілька типів навчання: з учителем, без учителя та з підкріпленням. У першому випадку моделі навчаються на даних, де відомий правильний результат (наприклад, «це кіт», «це пес»). У безвчительному навчанні алгоритм самостійно шукає структуру в даних, наприклад, групує схожі товари або клієнтів. Навчання з підкріпленням передбачає, що система вдосконалює свою поведінку шляхом проб і помилок, отримуючи винагороду або покарання, подібно до того, як людина навчається на власному досвіді.

Методи машинного навчання сьогодні використовуються практично в усіх сферах людської діяльності — від рекомендаційних систем YouTube і Netflix до автономного транспорту, медицини, фінансів, кібербезпеки, перекладу мов, голосових асистентів та навіть цифрового мистецтва. Проте машинне навчання не є магічним процесом — воно базується на складних математичних моделях, статистиці та значних обчислювальних потужностях. Ефективність його застосування залежить від правильної постановки задачі, якості вхідних даних і доцільного вибору алгоритму для конкретної проблеми.

Моделі класифікації — це окремий клас алгоритмів машинного навчання, призначених для розпізнавання об'єктів або явищ за певними категоріями. Вони «вчаться» на прикладах із відомими класами і згодом можуть визначати, до якої категорії належить новий об'єкт. Наприклад, така модель здатна класифікувати

електронну пошту як «спам» чи «звичайне повідомлення», визначати хворобу за клінічними показниками або розпізнавати зображення тварин — «кіт», «пес» чи «птах» [6].

Створення моделі класифікації розпочинається з підготовки навчального набору даних, у якому кожен приклад має відому мітку класу. Алгоритм аналізує зв’язки між ознаками об’єкта (наприклад, розміром, кольором, формою) та відповідними класами. Після завершення навчання модель застосовується для прогнозування класів нових об’єктів, базуючись на набутому досвіді [7].

У даній роботі для класифікації даних пацієнтів із гепатитом було обрано такі моделі машинного навчання: Random Forest, Support Vector Machine (SVM), Logistic Regression та Gradient Boosting. Алгоритм Random Forest (випадковий ліс) є одним із найпоширеніших і найефективніших методів для задач класифікації та регресії. Його суть полягає в побудові ансамблю численних дерев рішень, кожне з яких навчається на випадковій підмножині навчальних даних [8-19].

Під час формування вузлів дерева використовуються лише випадкові підмножини ознак, що забезпечує різноманітність серед дерев і зменшує ризик перенавчання — тобто ситуації, коли модель занадто точно запам’ятовує тренувальні дані й не може узагальнювати нову інформацію [8]. Такий підхід дозволяє отримати більш стабільні та точні результати навіть на складних або зашумлених наборах даних.

Принцип роботи Random Forest показано на рисунку 1.1.

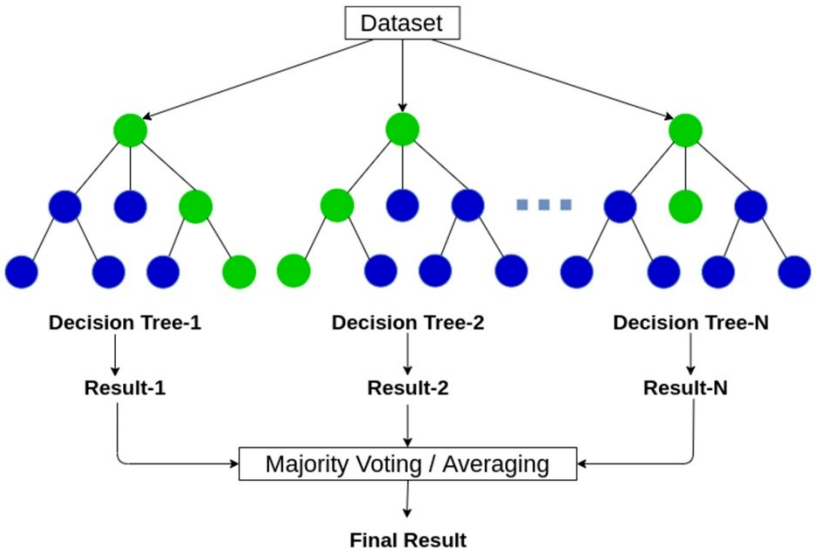

Рисунок 1.1 – Принцип роботи Random Forest

Після завершення процесу навчання, коли модель Random Forest складається з десятків або навіть сотень дерев рішень, вона функціонує за принципом колективного голосування. Для задач класифікації кожне дерево робить свій вибір — визначає, до якого класу належить об’єкт. Після цього підраховується кількість «голосів», і клас, який отримав найбільшу підтримку, приймається як остаточне рішення. У випадку регресійних задач модель обчислює середнє значення прогнозів усіх дерев. Такий ансамблевий підхід робить Random Forest менш чутливим до шуму в даних та випадкових помилок, порівняно з окремим деревом рішень [13-15].

Однією з головних переваг Random Forest є його здатність ефективно працювати з великими наборами даних, що містять як числові, так і категоріальні змінні. Алгоритм не потребує попереднього масштабування ознак і демонструє високі результати навіть без складного налаштування параметрів. Крім того, він може оцінювати важливість ознак, визначаючи, наскільки кожна з них впливає

на якість передбачення. Це робить Random Forest корисним не лише як інструмент для прогнозування, але й як засіб для інтерпретації даних.

Метод опорних векторів (Support Vector Machine, або SVM) є одним із найефективніших алгоритмів машинного навчання, що широко застосовується для класифікаційних задач, а також може використовуватись для регресії чи виявлення аномалій. Його основна ідея полягає у пошуку оптимальної межі — гіперплощини, яка найкраще розділяє дані різних класів.

На відміну від простих алгоритмів, SVM не задовольняється будь-якою роздільною межею між класами. Він прагне знайти гіперплощину, яка забезпечує найбільшу відстань (margin) до найближчих точок кожного класу. Ці точки називаються опорними векторами, і саме вони визначають положення та орієнтацію роздільної межі. Такий підхід робить модель більш стійкою до нових даних і зменшує ризик помилок у прикордонних зонах [17].

Алгоритм SVM демонструє високу ефективність у випадках, коли дані мають велику кількість ознак і відносно невеликий рівень шуму. У ситуаціях, коли класи не можна розділити прямою лінією або площиною, використовується спеціальний прийом — ядрове перетворення (kernel trick). Цей метод дозволяє відобразити дані у простір більшої розмірності, де вони стають лінійно роздільними. Таким чином, SVM може вирішувати складні нелінійні задачі класифікації, зберігаючи при цьому високу точність і стійкість моделі [16].

Принцип роботи алгоритму SVM показано на рисунку 1.2.

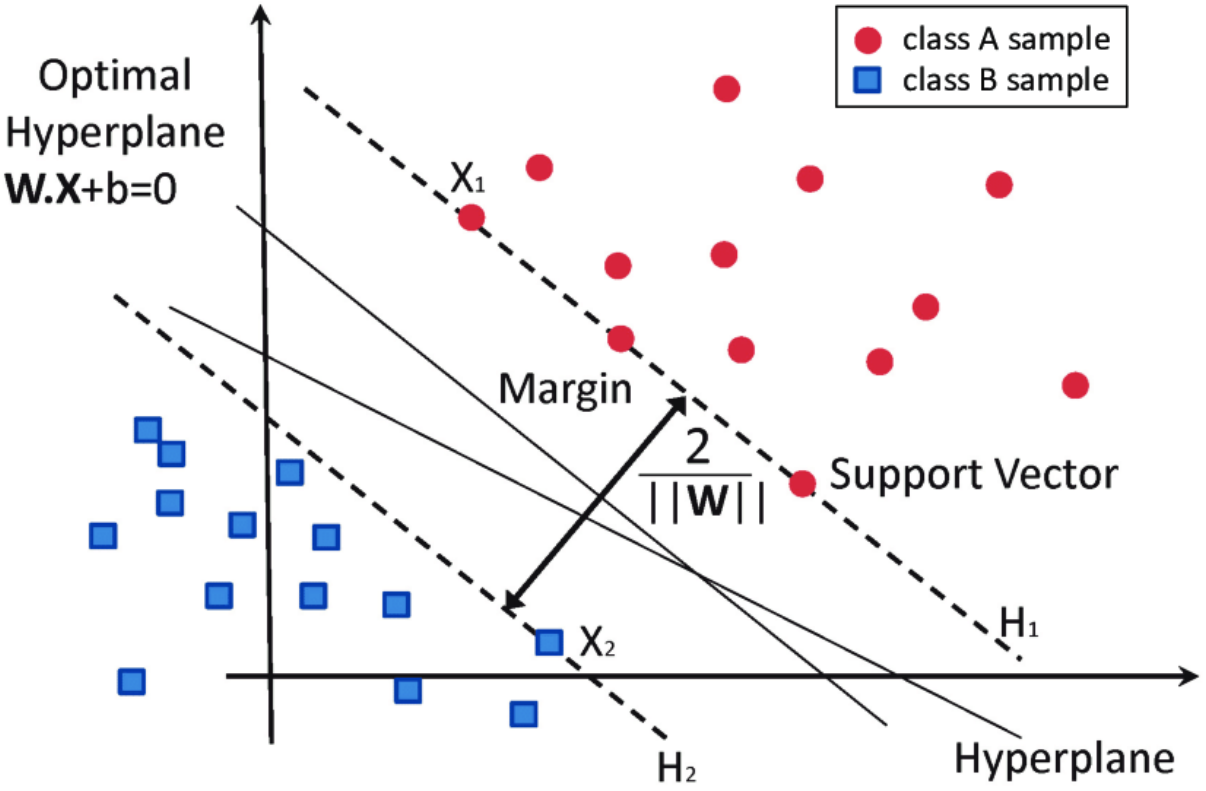

Рисунок 1.2 – Принцип роботи SVM

Серед найбільш популярних ядер — лінійне, поліноміальне, радіальне базисне функціональне (RBF) та сигмоїдальне. Вибір ядра сильно впливає на поведінку моделі, і тому часто потребує налаштування або підбору через крос-валідацію.

SVM має кілька сильних сторін: він добре працює в задачах, де кількість ознак значно перевищує кількість об'єктів, наприклад, у біоінформатиці чи текстовій класифікації. Також він досить ефективний навіть при обмежених обсягах навчальних даних. З іншого боку, він не дуже масштабований для дуже великих наборів даних і може бути повільним у навчанні, особливо з використанням складних ядер [16].

Logistic Regression (логістична регресія) — це базовий і водночас дуже важливий алгоритм машинного навчання, який використовується для задач бінарної класифікації, тобто коли потрібно визначити, до якого з двох класів належить об'єкт. Попри назву, логістична регресія — це не алгоритм для прогнозування чисел, як звичайна лінійна регресія, а метод для передбачення ймовірності належності до певного класу.

Суть логістичної регресії полягає в тому, що вона будує лінійну модель на основі вхідних ознак, але замість того, щоби напряму видавати числовий результат, вона застосовує логістичну (сигмоїдну) функцію, яка «стискає» результат у діапазон від 0 до 1. Це дозволяє інтерпретувати результат як імовірність належності до одного з класів. Наприклад, якщо результат моделі — 0.8, це означає, що ймовірність того, що об’єкт належить до позитивного класу, становить 80%.

На рисунку 1.3 показано порівняння LogisticRegression та LinearRegression.

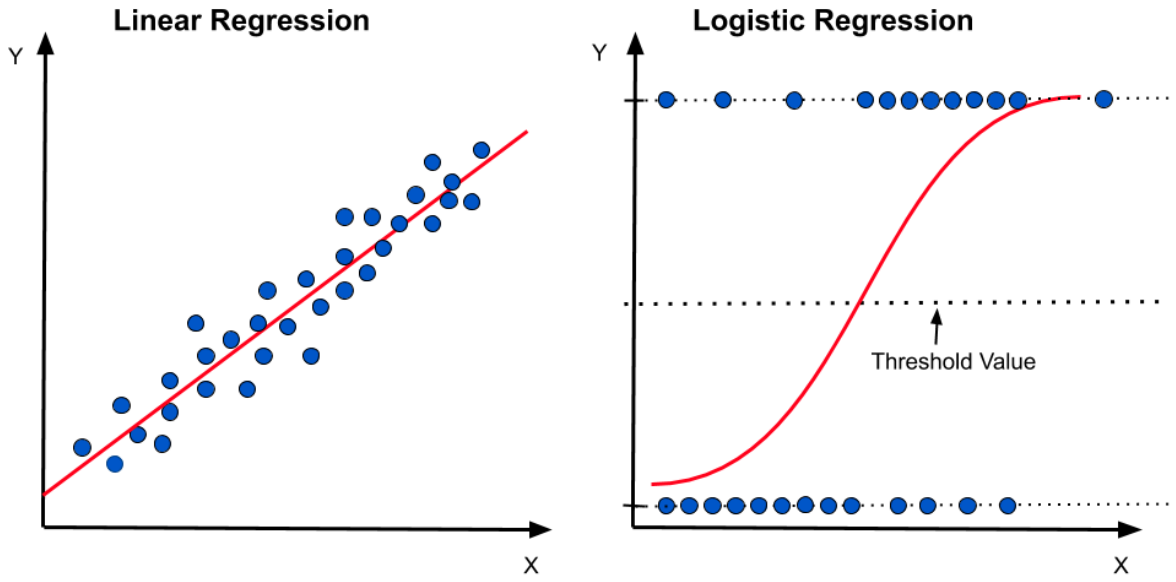

Рисунок 1.3 – Порівняння LogisticRegression та LinearRegression

Під час навчання логістична регресія підбирає ваги (коефіцієнти) для кожної ознаки таким чином, щоби мінімізувати логарифмічну функцію втрат (log loss), яка штрафує за помилки у передбаченні ймовірностей. Цей процес зазвичай відбувається за допомогою методу градієнтного спуску [17].

Модель добре працює, коли між ознаками та класом існує приблизно лінійний зв’язок. Вона швидка, проста для реалізації, легко інтерпретується, а також стійка до переобчислення, якщо застосовувати регуляризацію — техніку, яка зменшує вагу менш важливих ознак і запобігає перенавчанню.

Gradient Boosting — це потужний і гнучкий метод машинного навчання, який використовується як для класифікації, так і для регресії. Його основна ідея полягає в тому, щоби поступово створювати сильну модель шляхом об'єднання великої кількості слабких моделей — зазвичай це прості дерева рішень. Кожне нове дерево не просто працює незалежно, а вчиться на помилках попередніх моделей, намагаючись скоригувати їхні недоліки.

На рисунку 1.4 показано принцип роботи алгоритму Gradient Boosting.

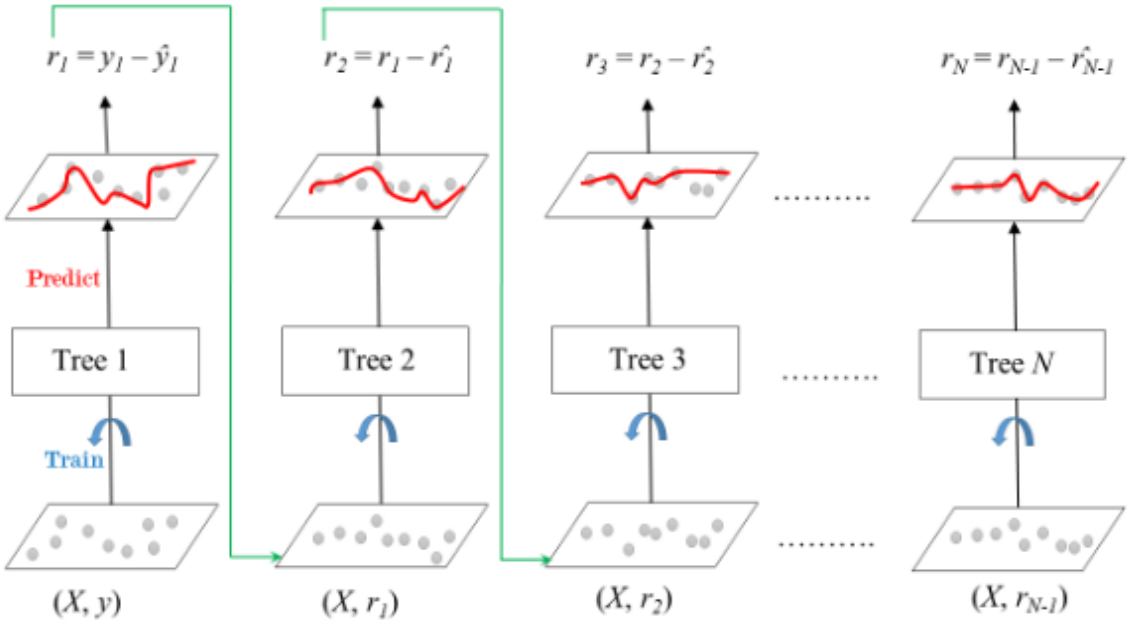

Рисунок 1.4 – Принцип роботи Gradient Boosting

На початку алгоритм створює першу просту модель, яка дає грубе передбачення. Потім обчислюється різниця між цим передбаченням і фактичним значенням — це і є помилка. Наступне дерево будується таким чином, щоб передбачити саме цю помилку, тобто модель поступово «виправляє сама себе». Цей процес повторюється багато разів, і кожне нове дерево додається до попередніх з певним коефіцієнтом ваги (швидкістю навчання), щоби не "перестаратись" і не зробити різких змін. У підсумку виходить складна модель, яка дуже добре адаптується до структури даних [16, 17].

Цей метод називається градієнтним, бо під час навчання помилки мінімізуються за допомогою методу градієнтного спуску — тобто алгоритм обирає такий напрям зміни, який найкраще зменшує загальну помилку.

Gradient Boosting чудово працює з табличними даними, де інші моделі, як-от нейронні мережі, можуть бути менш ефективні. Він дозволяє точно налаштовувати поведінку моделі через такі параметри, як кількість дерев, їхня глибина, швидкість навчання та вид функції втрат. Водночас цей метод може бути повільнішим у навчанні, особливо на великих наборах даних, і схильним до перенавчання, якщо не обмежувати складність дерев або не використовувати регуляризацію.

1.3 Аналіз мов програмування для розробки інформаційних технологій аналізу медичних даних

Вибір мови програмування для розробки інформаційної технології аналізу та передбачення стану хворих на гепатит є критично важливим рішенням, яке безпосередньо впливає на ефективність розробки, продуктивність системи та можливості подальшого розширення функціоналу. Сучасний ландшафт мов програмування пропонує численні варіанти, кожен з яких має свої переваги та обмеження в контексті обробки медичних даних, машинного навчання та статистичного аналізу.

Для комплексного порівняння було обрано п'ять найбільш поширених мов програмування, які активно використовуються в галузі аналізу даних та машинного навчання: Python, R, Julia, MATLAB та Java. Ці мови представляють різні парадигми програмування та мають різний рівень спеціалізації для задач Data Science. Проведення детального аналізу цих технологій дозволить обґрунтовано підійти до вибору оптимального інструментарію для реалізації поставлених завдань дослідження [8].

Python зарекомендував себе як провідна мова програмування в галузі науки про дані та машинного навчання. Його популярність зумовлена поєднанням простоти синтаксису, потужних можливостей та величезної екосистеми

спеціалізованих бібліотек. Мова була розроблена наприкінці 1980-х років Гвідо ван Россумом з акцентом на читабельність коду та продуктивність розробників. За останнє десятиліття Python став фактичним стандартом для проєктів машинного навчання, що підтверджується його домінуванням у навчальних програмах провідних університетів та широким використанням у дослідницьких інституціях.

Особливістю Python є його інтерпретована природа, що дозволяє швидко тестувати гіпотези та ітеративно розробляти моделі без необхідності компіляції. Динамічна типізація спрощує написання коду, хоча може призводити до помилок часу виконання, які в статично типізованих мовах виявляються на етапі компіляції. Проте для задач дослідження та прототипування ця особливість більше переваги, ніж недолік [9, 12].

Екосистема бібліотек Python для аналізу даних є найбільш розвиненою серед усіх розглянутих мов. NumPy забезпечує ефективну роботу з багатовимірними масивами та математичні операції, при цьому критичні операції виконуються на рівні оптимізованого коду C, що забезпечує високу продуктивність. Pandas надає потужні структури даних DataFrame та Series, які спеціально розроблені для роботи з табличними даними, що є типовим випадком для медичних досліджень.

Бібліотека scikit-learn містить реалізації практично всіх класичних алгоритмів машинного навчання з уніфікованим інтерфейсом, що значно спрощує експериментування з різними підходами. Для глибокого навчання доступні TensorFlow та PyTorch, які надають як високорівневі API для швидкої розробки, так і низькорівневі інструменти для створення нестандартних архітектур. Matplotlib та Seaborn забезпечують комплексні можливості візуалізації, що є критично важливим для розвідувального аналізу даних.

Важливою перевагою Python є його загальна універсальність. На відміну від спеціалізованих мов, Python дозволяє не тільки проводити аналіз даних, але й розробляти повноцінні вебдодатки, автоматизувати робочі процеси, інтегруватися з базами даних та створювати REST API. Це особливо цінно при

переході від прототипу до промислового рішення, оскільки не потрібно переписувати код на іншу мову.

Спільнота Python є однією з найбільших та найактивніших у світі програмування. Це означає наявність величезної кількості навчальних матеріалів, відповідей на Stack Overflow, готових рішень типових проблем та регулярних оновлень бібліотек. Для медичних досліджень існують спеціалізовані пакети, такі як Lifelines для аналізу виживання, Statsmodels для статистичного моделювання та Scikit-survival для аналізу часу до події.

Проте Python має і певні обмеження. Найсуттєвішим з них є відносно низька швидкість виконання чистого Python-коду порівняно з компільованими мовами. Global Interpreter Lock (GIL) обмежує можливості паралелізму на рівні потоків, хоча це частково компенсується використанням процесів або спеціалізованих бібліотек. Для ресурсномістких обчислень часто доводиться звертатися до оптимізованих бібліотек, написаних на C або Fortran, що може створювати залежності від системних компонентів [12].

Мова програмування R була спеціально розроблена для статистичного обчислення та візуалізації даних. Створена в середині 1990-х років Россом Ігакою та Робертом Джентлменом як вільна реалізація мови S, R швидко набула популярності в академічному середовищі, особливо серед статистиків та біоінформатиків. Мова орієнтована на векторні операції та функціональний стиль програмування, що природно відповідає потребам статистичного аналізу.

Ключовою особливістю R є те, що багато сучасних статистичних методів спочатку реалізуються саме в R, часто самими авторами відповідних наукових публікацій. Це означає, що дослідники отримують доступ до найновіших методів значно швидше, ніж в інших мовах. Comprehensive R Archive Network (CRAN) містить понад 18000 пакетів, які охоплюють практично всі області статистики, від класичних методів до найсучасніших розробок [8].

Для медичного аналізу R пропонує унікальні можливості. Пакет survival містить всебічну функціональність для аналізу виживання, що особливо важливо для клінічних досліджень. Пакет caret надає уніфікований інтерфейс до понад 200 алгоритмів машинного навчання, автоматизуючи процеси попередньої

обробки, вибору моделей та оцінки продуктивності. Біоінформатичний проєкт Bioconductor пропонує спеціалізовані інструменти для аналізу геномних даних, експресії генів та протеоміки.

Система візуалізації ggplot2, заснована на граматиці графіків Лілана Уілкінсона, забезпечує декларативний підхід до створення складних багатовимірних візуалізацій. Цей підхід дозволяє створювати публікаційно готові графіки з мінімальними зусиллями, що особливо цінно для наукових звітів та статей. Інтерактивні візуалізації можна створювати за допомогою пакетів Shiny, plotly та htmlwidgets.

R має природну інтеграцію з LaTeX через пакет knitr, що дозволяє створювати відтворювані дослідницькі документи, де код, результати та текст пояснень поєднані в єдиному документі. Це забезпечує прозорість наукового процесу та спрощує оновлення аналізу при надходженні нових даних або виявленні помилок.

Однак R має і суттєві недоліки для задач загального призначення. Синтаксис мови може здаватися незвичним програмістам, які звикли до більш традиційних мов. Множина способів вирішення однієї задачі може бути джерелом плутанини для початківців. Управління пам'яттю в R менш ефективне, ніж у Python, що може призводити до проблем при роботі з великими датасетами. База даних зберігається в оперативній пам'яті, що обмежує розмір аналізованих даних доступною RAM [8].

Продуктивність чистого R-коду часто поступається Python, особливо для задач, які важко векторизувати. При цьому перехід від аналітичного прототипу до промислової системи часто вимагає переписування коду на іншу мову, оскільки R менш підходить для розробки повноцінних додатків, вебсервісів або інтеграції з корпоративними системами.

Julia є відносно новою мовою програмування, представленою в 2012 році з амбітною метою поєднати швидкість виконання мов на кшталт C та Fortran зі зручністю динамічних мов, таких як Python та R. Розробники Julia прагнули вирішити проблему "двох мов", коли прототипи створюються на зручній

високорівневій мові, а потім критичні частини переписуються на компільовану мову для досягнення належної продуктивності.

Центральною ідеєю Julia є використання Just-In-Time (JIT) компіляції через LLVM. При першому виклику функції з певними типами аргументів Julia компілює її в оптимізований машинний код, який потім кешується для подальшого використання. Це дозволяє досягати продуктивності, порівнянної з C, зберігаючи при цьому інтерактивність та динамічність, характерну для інтерпретованих мов. Множинна диспетчеризація (multiple dispatch) є основою системи типів Julia, що дозволяє елегантно вирішувати багато проблем, які в інших мовах вимагають складних архітектурних рішень [8].

Для наукових обчислень Julia пропонує багату стандартну бібліотеку, яка включає вбудовану підтримку лінійної алгебри, статистики, випадкових чисел та паралелізму. Екосистема пакетів швидко розвивається: Plots.jl забезпечує уніфікований інтерфейс до різних бекендів візуалізації, DataFrames.jl надає функціональність, подібну до pandas, а MLJ.jl пропонує уніфікований доступ до численних алгоритмів машинного навчання.

Особливо цінною можливістю Julia є природна підтримка паралельних та розподілених обчислень. Мова була розроблена з урахуванням сучасних багатоядерних процесорів та кластерних систем, що дозволяє легко масштабувати обчислення без складних модифікацій коду. Підтримка GPU обчислень через пакет CUDA.jl дозволяє ефективно використовувати графічні процесори для прискорення навчання моделей глибокого навчання.

Незважаючи на свої переваги, Julia все ще має певні обмеження для широкого впровадження. Екосистема пакетів, хоч і швидко розвивається, все ще значно менша за Python або R. Це означає, що для деяких спеціалізованих задач може не існувати готових рішень. Час першої компіляції може бути суттєвим, що уповільнює інтерактивну роботу та ускладнює швидке тестування невеликих змін у коді.

Спільнота Julia менша, ніж у більш зрілих мов, що означає меншу кількість навчальних матеріалів, прикладів та відповідей на типові питання. Багато бібліотек все ще знаходяться в активній розробці, що може призводити до

breaking changes між версіями. Для промислового використання це може створювати ризики щодо довгострокової підтримки коду.

MATLAB (Matrix Laboratory) є комерційним середовищем та мовою програмування, розробленою компанією MathWorks, яка домінує в галузі інженерних та наукових обчислень. Історія MATLAB починається з кінця 1970-х років, коли Клів Молер створив його як інтерфейс до бібліотек LINPACK та EISPACK. З того часу MATLAB еволюціонував у повнофункціональне середовище з величезною кількістю спеціалізованих інструментальних пакетів (toolboxes).

Основою MATLAB є матричні операції, що відображено навіть у назві. Всі дані в MATLAB розглядаються як матриці, що робить код природним та лаконічним для задач лінійної алгебри та чисельних методів. Вбудовані функції високо оптимізовані та використовують багатопоточність за замовчуванням, що забезпечує високу продуктивність без необхідності явного програмування паралелізму [8].

Для задач машинного навчання MATLAB пропонує Statistics and Machine Learning Toolbox, який містить реалізації широкого спектру алгоритмів класифікації, регресії, кластеризації та зменшення розмірності. Deep Learning Toolbox підтримує створення, навчання та розгортання нейронних мереж різних архітектур. Важливою перевагою є наявність App Designer — візуального середовища для створення інтерактивних додатків без необхідності глибоких знань розробки GUI.

Інтеграція MATLAB з Simulink дозволяє моделювати складні динамічні системи та виконувати симуляції, що може бути корисним для прогнозування розвитку захворювань або моделювання ефективності лікування. Автоматична генерація коду C/C++ або HDL з моделей MATLAB дозволяє переносити розроблені алгоритми на вбудовані системи або спеціалізоване обладнання.

MATLAB широко використовується в академічних інституціях завдяки комплексним ліцензіям для освітніх закладів, що робить його знайомим для багатьох дослідників. Документація та технічна підтримка від MathWorks є вичерпними та високої якості, що спрощує вирішення технічних проблем.

Однак найсуттєвішим недоліком MATLAB є його комерційна природа. Вартість базової ліцензії досить висока, а спеціалізовані toolbox-и продаються окремо, що може зробити повнофункціональне середовище надзвичайно дорогим. Для невеликих дослідницьких груп або індивідуальних розробників це може бути вирішальним фактором проти використання MATLAB.

Відкритість коду також обмежена — більшість вбудованих функцій є закритими, що ускладнює розуміння деталей реалізації та налагодження складних проблем. Інтеграція MATLAB з іншими технологіями може бути складнішою, ніж у відкритих мов. Розгортання розроблених рішень також вимагає наявності MATLAB Runtime або придбання додаткових ліцензій для користувачів.

Java є об'єктно-орієнтованою мовою програмування, яка зарекомендувала себе як надійна платформа для розробки великомасштабних корпоративних систем. Розроблена в середині 1990-х років компанією Sun Microsystems, Java базується на принципі "напиши один раз, запускай всюди" завдяки використанню Java Virtual Machine (JVM), яка забезпечує портабельність скомпільованого байт-коду між різними операційними системами.

Для задач науки про дані Java пропонує бібліотеку Weka, яка містить колекцію алгоритмів машинного навчання для задач інтелектуального аналізу даних. Deeplearning4j є потужною бібліотекою глибокого навчання, розробленою спеціально для Java та інтегрованою з екосистемою великих даних. Apache Spark з API для Java дозволяє обробляти величезні обсяги даних у розподіленому середовищі, що критично важливо для промислових застосувань.

Статична типізація Java забезпечує ранню детекцію помилок на етапі компіляції, що знижує ймовірність runtime помилок у продуктивному середовищі. Строга система типів може здаватися обтяжливою при швидкому прототипуванні, але вона надає додаткову безпеку для великих кодових баз, що підтримуються командами розробників. Автоматичне управління пам'яттю через збирання сміття спрощує розробку, хоча може призводити до непередбачуваних пауз у виконанні програми.

Java має чудову підтримку багатопоточності на рівні мови, що дозволяє ефективно використовувати сучасні багатоядерні процесори. Велика кількість зрілих бібліотек для роботи з базами даних, вебсервісами, обміну повідомленнями та іншими корпоративними технологіями робить Java природним вибором для інтеграції аналітичних компонентів у існуючі інформаційні системи [8].

Проте для типових задач дослідницького аналізу даних Java має суттєві недоліки. Багатослівність синтаксису вимагає написання значної кількості коду для виконання простих операцій, що сповільнює цикл експериментування. Екосистема бібліотек для науки про дані значно менша та менш зріла порівняно з Python або R. Візуалізація даних у Java менш зручна та потужна, ніж спеціалізовані інструменти інших мов.

Відсутність інтерактивного режиму (REPL — Read-Eval-Print Loop) у стандартній Java робить швидке тестування гіпотез незручним. Хоча існують проєкти на кшталт JShell, вони не є настільки інтегрованими в робочий процес, як Jupyter для Python. Навчальна крива для Java крутіша, особливо для людей без міцного програмістського бекграунду, які приходять зі статистики або предметних галузей [8].

Для систематизації отриманої інформації було проведено порівняльний аналіз розглянутих мов програмування за ключовими критеріями, важливими для розробки інформаційної технології аналізу медичних даних. Результати аналізу представлено в таблиці 1.1, де оцінка здійснювалась за п'ятибальною шкалою: 5 — відмінно, 4 — добре, 3 — задовільно, 2 — погано, 1 — дуже погано.

Таблиця 1.1 — Порівняльна оцінка мов програмування

Критерій	Python	R	Julia	MATLAB	Java
Зручність для прототипування	5	5	4	4	2
Продуктивність обчислень	3	2	5	4	4
Екосистема ML бібліотек	5	4	3	4	2
Статистичні можливості	4	5	4	4	2
Візуалізація даних	5	5	3	4	2
Інтеграційні можливості	5	3	3	3	5
Розмір спільноти	5	4	2	3	5
Навчальні ресурси	5	4	2	4	5
Вартість використання	5	5	5	1	5
Промислова зрілість	5	3	2	5	5

Аналіз таблиці 1.1 показує, що Python займає лідируючі позиції за більшістю критеріїв, особливо тих, що є критичними для дослідницьких проєктів. Висока оцінка за зручність прототипування відображає інтуїтивний синтаксис та інтерактивне середовище розробки. Найбільша екосистема бібліотек машинного навчання забезпечує доступ до найсучасніших алгоритмів та архітектур. Величезна спільнота гарантує швидке вирішення проблем та постійний розвиток інструментів.

R демонструє найвищі показники саме в статистичному аналізі, що є його історичною сильною стороною. Для задач, які вимагають складних статистичних моделей або специфічних біостатистичних методів, R може бути оптимальним вибором. Проте його обмежена придатність для побудови повноцінних додатків знижує загальну привабливість для комплексних проєктів.

Julia виділяється винятковою продуктивністю, що робить її перспективною для обчислювально інтенсивних задач. Однак молодість мови та відносно невелика екосистема створюють ризики щодо доступності необхідних інструментів та стабільності API. Для дослідницьких проєктів з довгостроковою перспективою це може бути проблематично.

MATLAB зберігає сильні позиції в інженерних застосуваннях та академічній сфері, проте висока вартість є серйозною перешкодою. Для індивідуальних дослідників або невеликих груп використання MATLAB може бути економічно недоцільним, особливо коли існують безкоштовні альтернативи з порівнянними можливостями.

Java, незважаючи на відмінні інтеграційні можливості та промислову зрілість, не є оптимальним вибором для дослідницької фази проєкту. Багатослівність синтаксису та обмежена екосистема для науки про дані роблять його менш привабливим порівняно з спеціалізованими мовами. Проте на етапі промислового впровадження Java може стати корисною для побудови масштабованих сервісів.

Детальний аналіз переваг та недоліків кожної мови представлено в таблиці 1.2, яка систематизує ключові характеристики для прийняття обґрунтованого рішення щодо вибору технологічного стеку проєкту.

Таблиця 1.2 — Переваги та недоліки мов програмування для аналізу даних

Мова	Переваги	Недоліки
Python	- Найбільша екосистема ML бібліотек. - Простий та читабельний синтаксис. - Величезна спільнота та ресурси. - Універсальність застосування. - Безкоштовні інструменти. - Jupyter Notebooks для документування. - Легка інтеграція з іншими системами.	- Повільніше виконання чистого Python коду. - GIL обмежує багатопоточність. - Динамічна типізація може призводити до runtime помилок. - Управління залежностями може бути складним. - Версійна несумісність (Python 2 vs 3).

Продовження таблиці 1.2.

Мова	Переваги	Недоліки
R	- Найпотужніші статистичні можливості. - Спеціалізовані пакети для біостатистики. - Чудова візуалізація (ggplot2) - Відтворюваність досліджень (knitr, RMarkdown). - Безкоштовні інструменти. - Активна наукова спільнота.	- Синтаксис може здаватися незвичним. - Повільніше виконання коду. Неефективне управління пам'яттю. - Обмежена придатність для production. - Складніша інтеграція з іншими системами. - Менша екосистема для глибокого навчання.
Julia	• Висока продуктивність. - Природна підтримка паралелізму. - Сучасний дизайн мови. - Зручність динамічної мови. - Безкоштовні інструменти. - Ефективна робота з GPU.	- Невелика екосистема пакетів. - Мала спільнота. - Час першої компіляції. - Недостатньо навчальних матеріалів. - Нестабільність API деяких пакетів. - Обмежена підтримка IDE.
MATLAB	- Потужні вбудовані функції. - Чудова документація. - Simulink для моделювання. - Оптимізовані обчислення. - Професійна технічна підтримка. - Генерація коду C/C++.	- Висока вартість ліцензій. - Закритий вихідний код. - Обмежена інтеграція з іншими технологіями. - Складніше розгортання додатків. - Менша гнучкість порівняно з відкритими мовами.
Java	- Відмінна промислова зрілість. - Статична типізація підвищує надійність. - Чудова підтримка багатопоточності. - Велика екосистема корпоративних бібліотек. - Портабельність через JVM. - Автоматичне управління пам'яттю	- Багатослівний синтаксис. - Повільне прототипування. - Обмежена екосистема для науки про дані - Слабкіші можливості візуалізації. - Відсутність зручного REPL.

На основі проведеного комплексного аналізу можна зробити висновок, що для задачі розробки інформаційної технології аналізу та передбачення стану хворих на гепатит найбільш оптимальним вибором є мова програмування Python. Це рішення обґрунтовується наступними ключовими факторами:

По-перше, Python надає найбільш збалансоване поєднання зручності розробки та функціональних можливостей для всіх етапів проєкту — від розвідувального аналізу даних до побудови та тестування моделей машинного навчання. Бібліотеки Pandas, NumPy, Scikit-learn та інші забезпечують повний інструментарій для реалізації поставлених задач без необхідності звертатися до додаткових технологій. По-друге, величезна спільнота та велика кількість навчальних матеріалів у ріхих форматах

1.4 Висновки

У першому розділі було розглянуто об’єкт дослідження та визначено його актуальність.Здійснено аналіз мов програмування та середовищ, які можуть бути використанні для програмної реалізації інформаціної технології аналізу та передбачення стану хворих на гепатит. Також було досліджено можливості їх реалізації з використанням сучасних інформаційних технологій, зокрема мови програмування Python і платформи Kaggle. На основі цього обрано оптимальні інструменти для ефективного розв’язання поставленої задачі.

2 РОЗВІДУВАЛЬНИЙ АНАЛІЗ

2.1 Аналіз середовищ розробки для реалізації інформаційної технології

Вибір середовища розробки є не менш важливим рішенням, ніж вибір мови програмування, оскільки воно безпосередньо впливає на продуктивність розробника, якість коду, швидкість налагодження та загальну ефективність процесу створення програмного забезпечення. Для мови програмування Python існує широкий спектр інтегрованих середовищ розробки (IDE) та текстових редакторів, кожен з яких має свої унікальні особливості, переваги та недоліки. У контексті розробки інформаційної технології для аналізу медичних даних та машинного навчання особливого значення набувають такі характеристики середовища, як підтримка інтерактивного виконання коду, можливості візуалізації, інтеграція з науковими бібліотеками, засоби налагодження та можливість співпраці з іншими розробниками.

Сучасний ландшафт інструментів для розробки на Python включає як традиційні IDE з повним набором функцій, так і легковагі текстові редактори з розширеннями, а також спеціалізовані середовища для інтерактивної роботи з даними. Для проведення комплексного аналізу було обрано шість найбільш популярних та функціональних середовищ: Jupyter Notebook/JupyterLab, PyCharm, Visual Studio Code, Spyder, Google Colab та Kaggle Notebooks. Ці інструменти представляють різні підходи до організації робочого процесу та мають різний рівень спеціалізації для задач Data Science [8-12].

Jupyter Notebook є веборієнтованим додатком (рис. 2.1), що дозволяє створювати та ділитися документами, які містять виконуваний код, рівняння, візуалізації та пояснювальний текст. Назва "Jupyter" походить від трьох основних мов програмування, які воно підтримувало спочатку: Julia, Python та R. Проєкт виріс з IPython Notebook, створеного Фернандо Пересом у 2001 році, і став стандартом де-факто для інтерактивної роботи з даними в науковій спільноті.

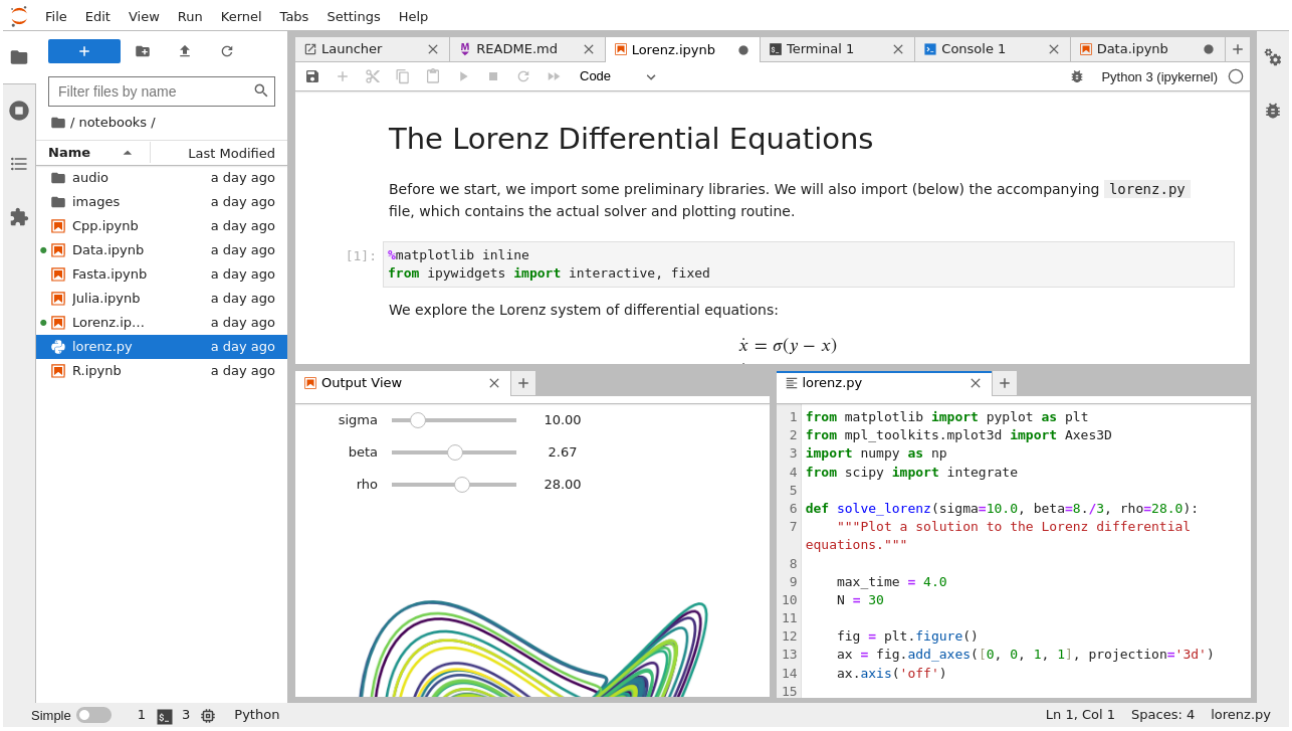

Рисунок 2.1 – Загальний вигляд Jupyter Notebook

Архітектура Jupyter базується на моделі клієнт-сервер, де kernel (ядро) виконує код, а інтерфейс користувача працює в веббраузері. Це дозволяє запускати обчислення на віддалених серверах, що особливо корисно для роботи з великими датасетами або обчислювально інтенсивними моделями. Документи Jupyter називаються notebooks (блокнотами) і складаються з комірок, які можуть містити код, текст у форматі Markdown, HTML, LaTeX для математичних формул або виводи виконання коду, включно з графіками.

Ключовою особливістю Jupyter є можливість виконання коду по частинах (cell-by-cell execution). Це означає, що дослідник може запустити певний фрагмент коду, проаналізувати результати, внести зміни та виконати наступний фрагмент без необхідності перезапуску всієї програми. Такий підхід ідеально підходить для розвідувального аналізу даних, коли потрібно швидко тестувати різні гіпотези та переглядати результати. Змінні зберігаються в пам'яті між виконаннями комірок, що дозволяє поступово будувати аналіз крок за кроком.

Інтеграція з бібліотеками візуалізації, такими як Matplotlib, Seaborn та Plotly, робить Jupyter надзвичайно потужним інструментом для створення графіків.

Magic commands (магічні команди) є спеціальними директивами Jupyter, які починаються з символу % (для однієї лінії) або %% (для всієї комірки). Вони надають доступ до додаткової функціональності, такої як вимірювання часу виконання коду (%timeit), завантаження зовнішніх скриптів (%run), налагодження (%debug), робота з різними мовами програмування в одному блокноті та багато іншого. Це значно розширює можливості базового Python без необхідності імпорту додаткових модулів.

JupyterLab є наступним поколінням інтерфейсу для Jupyter, який пропонує більш гнучке та інтегроване середовище. На відміну від класичного Jupyter Notebook, JupyterLab надає повноцінне IDE-подібне середовище з можливістю розташування декількох блокнотів, терміналів, текстових редакторів та переглядачів файлів у вкладках або поруч у розділеному вікні. Це дозволяє працювати над кількома аспектами проєкту одночасно, наприклад, переглядати дані в одному блокноті, розробляти модель в іншому та відслідковувати метрики в третьому [8].

Система розширень JupyterLab дозволяє додавати нову функціональність, таку як інтеграція з Git, перегляд змінних у пам'яті, форматування коду, підтримка таблиць та багато іншого. Завдяки відкритій архітектурі та активній спільноті регулярно з'являються нові розширення, які покращують функціональність середовища. Підтримка тем дозволяє налаштувати візуальний вигляд відповідно до особистих переваг.

Одним з найбільших переваг Jupyter є його роль у відтворюваності наукових досліджень. Блокнот містить не тільки код та результати, але й пояснення логіки аналізу, що робить дослідження прозорим та зрозумілим для інших. Можливість експорту блокнотів у різні формати (HTML, PDF, LaTeX, презентації) дозволяє легко ділитися результатами з колегами або публікувати їх. Платформи на кшталт nbviewer дозволяють переглядати блокноти безпосередньо в браузері без необхідності запуску Jupyter на локальній машині.

Проте Jupyter має і певні обмеження. Найсуттєвішим недоліком є те, що нелінійний порядок виконання комірок може призводити до проблем з відтворюваністю результатів. Якщо комірки виконуються не в порядку зверху

вниз, стан змінних може стати неочікуваним, що ускладнює налагодження. Блокноти можуть стати занадто великими та неорганізованими, якщо не дотримуватися дисципліни в структуруванні коду.

Контроль версій для блокнотів є проблематичним, оскільки файли .ipynb зберігаються у форматі JSON з великою кількістю метаданих, включно з виводами виконання. Це робить diff між версіями важко читабельним і може призводити до конфліктів при злитті змін у команді. Хоча існують інструменти на кшталт nbdime для покращення роботи з версіями блокнотів, проблема залишається не повністю вирішеною.

Відсутність традиційних IDE функцій, таких як потужний рефакторинг коду, перехід до визначення функцій, інтелектуальне автодоповнення та аналіз якості коду, робить Jupyter менш придатним для розробки великих програмних систем. Для складних проєктів зазвичай код виноситься в окремі Python модулі, які потім імпортуються в блокнот, що створює додаткову складність у підтримці.

PyCharm, розроблений компанією JetBrains, є повнофункціональним інтегрованим середовищем розробки, спеціально створеним для Python. Перша версія була випущена в 2010 році, і з того часу PyCharm став одним з найпопулярніших IDE для Python розробників. Середовище доступне у двох редакціях: безкоштовна Community Edition з базовим набором функцій та комерційна Professional Edition з розширеними можливостями для веброзробки, наукових обчислень та роботи з базами даних.

Інтелектуальний редактор коду PyCharm забезпечує контекстно-залежне автодоповнення, яке аналізує не тільки синтаксис мови, але й структуру проєкту, імпортовані бібліотеки та типи змінних. Система інспекцій коду в реальному часі виявляє потенційні помилки, стилістичні проблеми відповідно до PEP 8, невикористані імпорти та змінні, що значно підвищує якість коду. Швидкі виправлення (quick fixes) дозволяють автоматично вирішити багато виявлених проблем одним натисканням клавіші.

Потужні можливості рефакторингу включають безпечне перейменування змінних, функцій та класів по всьому проєкту, витягування методів, переміщення коду між файлами, зміну сигнатур функцій та багато іншого. PyCharm

відслідковує всі використання перейменованих сутностей та автоматично оновлює їх, мінімізуючи ризик помилок при реорганізації коду. Це особливо цінно при роботі над великими проєктами, де ручне відстеження всіх залежностей було б надзвичайно складним [9].

Вбудований налагоджувач (debugger) PyCharm є одним з найпотужніших інструментів для розробників. Він дозволяє встановлювати точки зупинки (breakpoints), виконувати код покроково, переглядати та змінювати значення змінних під час виконання, оцінювати довільні вирази в контексті поточного стану програми. Підтримка умовних точок зупинки дозволяє зупиняти виконання тільки при виконанні певних умов, що значно прискорює налагодження складних сценаріїв.

Professional Edition PyCharm включає спеціальну підтримку для науки про дані через інтеграцію Jupyter Notebooks безпосередньо в IDE. Це поєднує переваги інтерактивних обчислень Jupyter з потужними можливостями PyCharm, такими як рефакторинг та налагодження. Інтерактивні графіки можна переглядати в окремій панелі, а SciView надає зручний інтерфейс для роботи з NumPy масивами та Pandas DataFrames.

Інтеграція з системами контролю версій (Git, GitHub, GitLab, Bitbucket) дозволяє виконувати всі операції з версіями безпосередньо з IDE: створювати коміти, переглядати історію змін, розв'язувати конфлікти злиття, створювати та переключатися між гілками. Вбудований редактор відмінностей (diff viewer) з підсвічуванням змін значно спрощує перегляд та злиття коду.

Підтримка віртуальних середовищ та менеджерів пакетів (virtualenv, conda, pipenv, poetry) інтегрована безпосередньо в інтерфейс. PyCharm може автоматично виявляти requirements.txt або environment.yml файли та пропонувати встановити необхідні залежності. Панель Python Packages дозволяє легко шукати, встановлювати та оновлювати пакети без необхідності використання командного рядка.

Однак PyCharm є досить ресурсномістким додатком, що може бути проблематично на машинах з обмеженими ресурсами. Для індексації великих проєктів потрібен значний час, особливо при першому відкритті. Professional

Edition є комерційним продуктом, хоча надає безкоштовні ліцензії для освітніх цілей та опенсорс проєктів. Для простих скриптів або швидких експериментів запуск повноцінного IDE може здаватися надмірним.

Інтерфейс PyCharm з великою кількістю панелей, меню та налаштувань може бути overwhelming для початківців. Навчальна крива досить крута, і потрібен час, щоб освоїти всі можливості середовища. Проте для серйозної розробки складних проєктів інвестиція часу в освоєння PyCharm повністю окупається підвищеною продуктивністю.

Visual Studio Code (VS Code) є безкоштовним редактором вихідного коду з відкритим кодом, розробленим компанією Microsoft. Випущений у 2015 році, VS Code швидко набув популярності завдяки поєднанню легковаговості текстового редактора з потужними можливостями IDE через систему розширень. Побудований на основі Electron framework, VS Code працює на всіх основних операційних системах та забезпечує однаковий досвід роботи незалежно від платформи.

Центральною ідеєю VS Code є модульність через розширення. Базова установка включає тільки основну функціональність редактора коду, а всі специфічні можливості для різних мов програмування, фреймворків та інструментів додаються через extensions marketplace. Для Python існує офіційне розширення від Microsoft, яке забезпечує підсвічування синтаксису, автодоповнення, linting, форматування коду, підтримку налагодження та інтеграцію з Jupyter Notebooks.

Розширення Python для VS Code включає Pylance — сервер мови, заснований на Pyright, який забезпечує швидке та точне автодоповнення на основі аналізу типів. IntelliSense в VS Code надає інформацію про параметри функцій, типи змінних та документацію прямо в редакторі при наведенні курсору. Підтримка type hints дозволяє отримати переваги статичної перевірки типів у динамічно типізованому Python [8].

Вбудована підтримка Jupyter Notebooks у VS Code дозволяє працювати з .ipynb файлами безпосередньо в редакторі з усіма перевагами IDE:

автодоповнення, навігація по коду, рефакторинг та налагодження комірок. Інтерактивний режим Python дозволяє виконувати виділений код у інтерактивному вікні, що поєднує переваги скриптів та блокнотів. Змінні з виконаного коду можна переглядати в окремій панелі Variable Explorer.

Вбудована інтеграція з Git є однією з найсильніших сторін VS Code. Source Control панель надає зручний візуальний інтерфейс для всіх операцій з версіями: перегляд змін, створення коммітів, робота з гілками, розв'язання конфліктів. Підтримка GitHub Pull Requests та Issues через розширення дозволяє переглядати та обговорювати код безпосередньо в редакторі.

Remote Development є унікальною можливістю VS Code, яка дозволяє розробляти код на віддалених машинах, контейнерах Docker або Windows Subsystem for Linux (WSL), при цьому інтерфейс редактора працює локально. Це надзвичайно корисно для машинного навчання, коли навчання моделей відбувається на потужних серверах з GPU, а розробка коду — на локальному комп'ютері. VS Code автоматично встановлює необхідні компоненти на віддаленій машині та забезпечує безшовний досвід роботи [9].

Інтегрований термінал дозволяє запускати команди без виходу з редактора, що значно прискорює робочий процес. Можна відкрити декілька терміналів, використовувати різні оболонки (bash, PowerShell, cmd) та організовувати їх у вкладки або розділені панелі. Підтримка task runner дозволяє автоматизувати повторювані команди, такі як запуск тестів, linting або побудова проєкту.

Marketplace VS Code містить тисячі розширень для різних цілей: теми оформлення, snippets для швидкого введення коду, інтеграція з різними сервісами та інструментами, підтримка додаткових мов програмування та форматів файлів. Можливість створювати власні розширення дозволяє адаптувати редактор під специфічні потреби проєкту або команди.

Проте VS Code, незважаючи на свою популярність, має і недоліки. Велика кількість розширень може призводити до конфліктів або зниження продуктивності. Необхідність окремого встановлення та налаштування розширень для кожної мови чи фреймворку може бути незручною для

початківців. Хоча VS Code позиціонується як легковаговий редактор, з великою кількістю встановлених розширень він може споживати значні ресурси системи.

Деякі просунуті функції рефакторингу, доступні в повноцінних IDE на кшталт PyCharm, в VS Code працюють менш надійно або відсутні взагалі. Налагоджувач, хоч і функціональний, не має деяких специфічних можливостей, корисних для розробки складних систем. Для великих корпоративних проєктів з суворими вимогами до якості коду VS Code може виявитися недостатнім без додаткового інструментарію.

Spyder (Scientific Python Development Environment) є інтегрованим середовищем розробки, спеціально створеним для науковців, інженерів та аналітиків даних, які працюють з Python. Проєкт був розпочатий П'єром Рапін у 2009 році як частина дистрибутива Python(x,y) і пізніше став самостійним продуктом. Spyder часто порівнюють з MATLAB IDE або RStudio, оскільки він надає подібний досвід роботи, орієнтований на інтерактивний науковий аналіз.

Інтерфейс Spyder (рис. 2.2) організований навколо чотирьох основних компонентів: редактор коду, інтерактивна консоль IPython, explorer змінних та панель довідки. Таке розташування створює ефективний робочий процес для наукових обчислень: код пишеться в редакторі, виконується в консолі, результати переглядаються в explorer змінних, а документація доступна в панелі довідки. Всі панелі можна переміщувати та налаштовувати відповідно до особистих переваг.

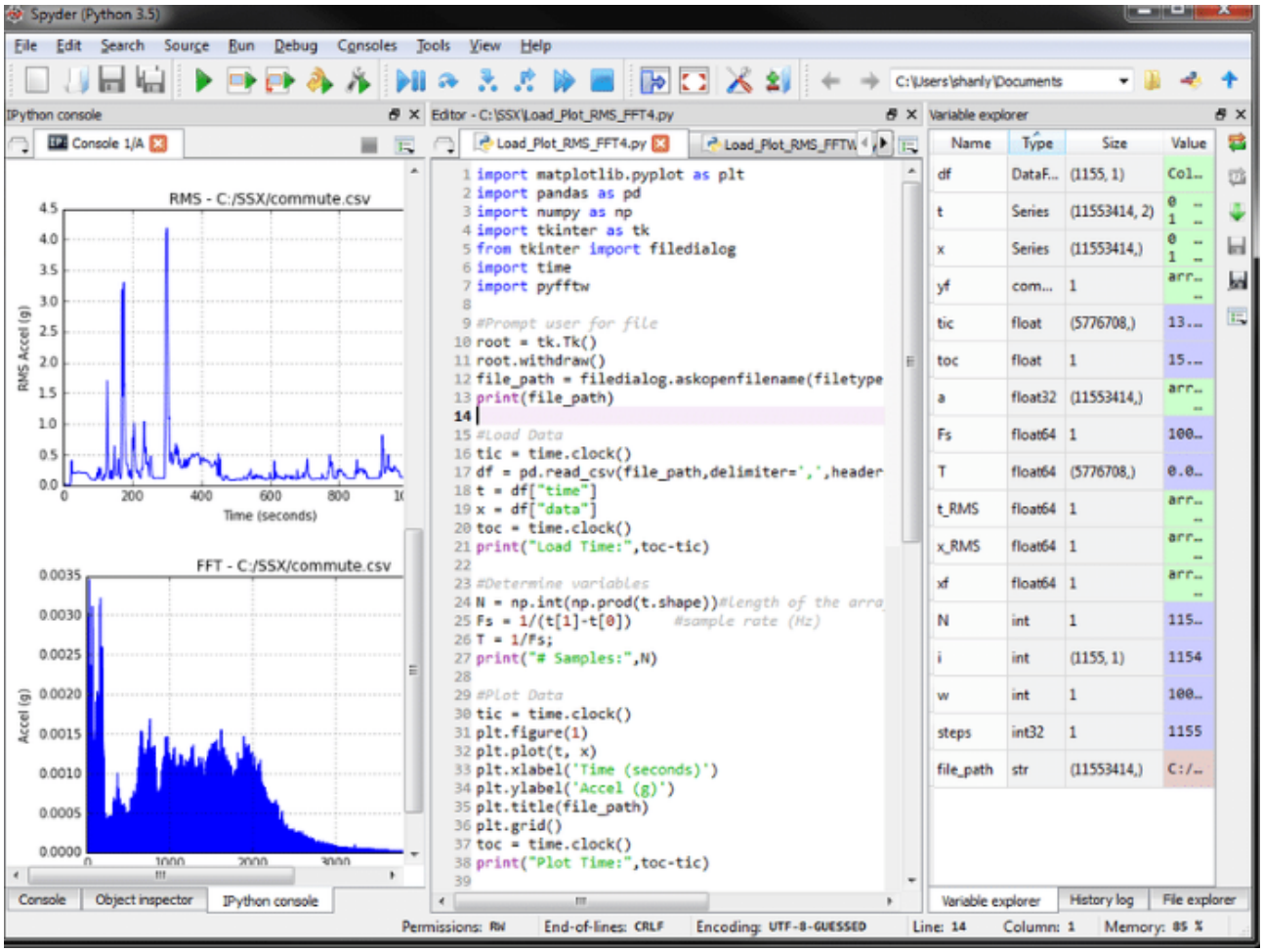

Рисунок 2.2 – Інтерфейс Spyder

Variable Explorer є однією з найпотужніших функцій Spyder. Він надає табличне представлення всіх змінних у поточному просторі імен з інформацією про їхній тип, розмір та значення. Для NumPy масивів та Pandas DataFrames доступний вбудований переглядач, який дозволяє інтерактивно досліджувати дані, сортувати, фільтрувати та редагувати значення. Для зображень доступний спеціальний переглядач, який відображає їх візуально. Це значно спрощує процес налагодження та розуміння стану програми в будь-який момент часу.

Інтерактивна консоль IPython в Spyder надає всі переваги IPython: підсвічування синтаксису, автодоповнення, magic commands, історію команд та багато іншого. Можна виконувати окремі фрагменти коду з редактора в консолі, що дозволяє тестувати функції без запуску всього скрипта. Підтримка декількох консолей дозволяє працювати з різними інтерпретаторами Python або різними проєктами одночасно [8].

Інтегрований налагоджувач в Spyder підтримує всі стандартні функції: точки зупинки, покрокове виконання, перегляд стеку викликів та змінних. Унікальною особливістю є можливість налагодження в інтерактивному режимі — коли виконання зупинено на точці зупинки, можна виконувати довільні команди в консолі в контексті поточного стану програми. Це надзвичайно корисно для розуміння причин помилок або експериментування з виправленнями.

Profiler та статичний аналізатор коду допомагають оптимізувати продуктивність програм. Profiler показує, скільки часу виконується кожна функція та скільки разів вона викликається, що допомагає ідентифікувати вузькі місця. Статичний аналізатор (на основі Pylint) виявляє потенційні помилки, порушення стилю коду та неоптимальні конструкції.

Spyder входить до складу дистрибутива Anaconda, що робить його установку та налаштування надзвичайно простими — середовище готове до роботи одразу після встановлення Anaconda. Інтеграція з conda дозволяє легко перемикатися між різними Python середовищами безпосередньо з інтерфейсу Spyder. Для користувачів, які переходять з MATLAB, Spyder надає звичний досвід роботи, що спрощує перехід.

Проте Spyder має і обмеження. Інтерфейс може здаватися застарілим порівняно з сучасними редакторами на кшталт VS Code. Можливості розширення функціональності через плагіни є обмеженими порівняно з іншими IDE. Продуктивність може знижуватися при роботі з дуже великими файлами або проєктами. Функції рефакторингу коду менш розвинені, ніж у PyCharm.

Система контролю версій інтегрована менш зручно — хоча існує плагін для Git, його функціональність базова, і багато операцій доводиться виконувати через зовнішні інструменти. Підтримка Jupyter Notebooks відсутня нативно, хоча є плагіни, які додають цю функціональність. Для веброзробки або створення складних додатків Spyder не є оптимальним вибором, оскільки орієнтований саме на науковий аналіз даних.

Google Colaboratory, відомий як Colab, є безкоштовним хмарним сервісом (рис. 2.3), що надає середовище Jupyter Notebook з доступом до обчислювальних ресурсів, включно з GPU та TPU. Запущений у 2017 році, Colab швидко став

популярним серед дослідників та студентів завдяки можливості виконувати обчислювально інтенсивні завдання машинного навчання без необхідності володіння власним потужним обладнанням.

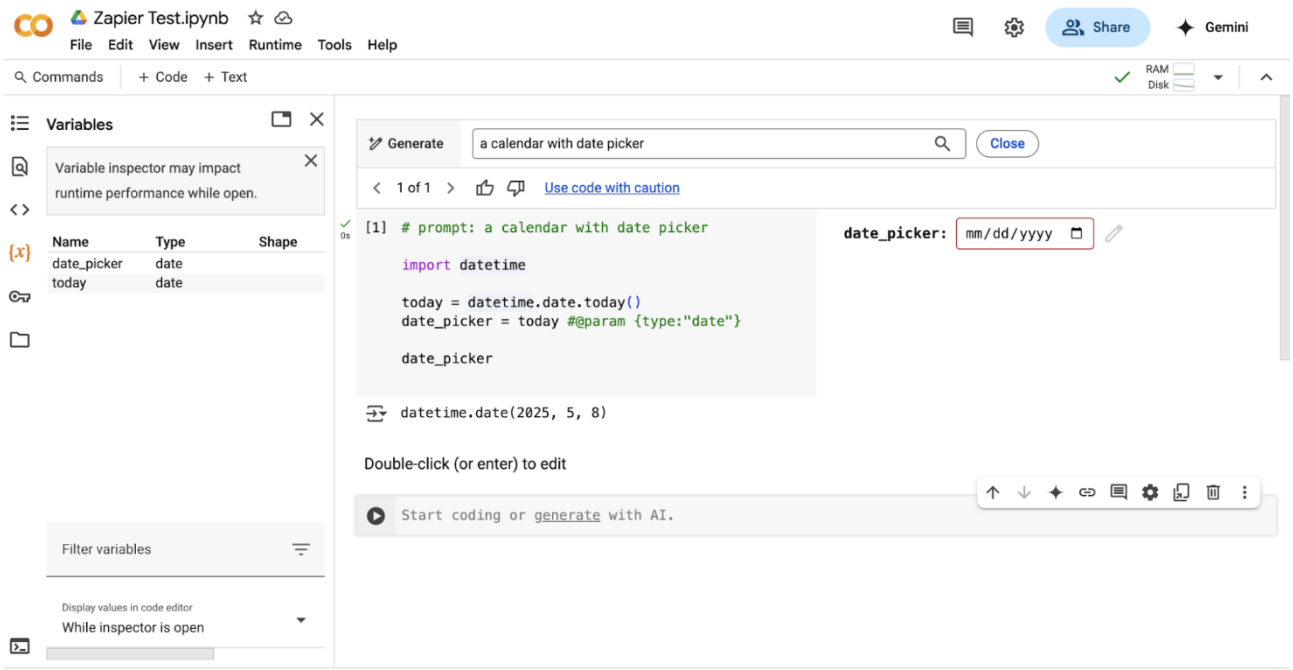

Рисунок 2.3 – Google Colab

Найбільшою перевагою Colab є безкоштовний доступ до апаратних прискорювачів. Користувачі можуть переключатися між CPU, GPU (зазвичай NVIDIA Tesla T4 або K80) та TPU (Tensor Processing Unit від Google) залежно від потреб завдання. Для навчання нейронних мереж доступ до GPU може прискорити обчислення в десятки разів порівняно з CPU. TPU надають ще більшу продуктивність для специфічних операцій глибокого навчання, особливо при роботі з TensorFlow.

Середовище Colab базується на Jupyter Notebook, тому користувачі, знайомі з Jupyter, відчують себе комфортно. Notebooks зберігаються в Google Drive, що забезпечує автоматичне збереження, контроль версій через історію ревізій та легке спільне використання. Можна надати доступ до блокнота іншим користувачам з різними рівнями прав: перегляд, коментування або редагування. Це робить Colab чудовим інструментом для співпраці та навчання [8, 9].

Більшість популярних бібліотек для машинного навчання та аналізу даних попередньо встановлені: TensorFlow, PyTorch, Keras, Scikit-learn, Pandas, NumPy, Matplotlib, Seaborn та багато інших. Це дозволяє відразу почати роботу без налаштування середовища. Якщо потрібна бібліотека відсутня, її можна легко встановити за допомогою команди !pip install. Інтеграція з Google Drive дозволяє легко завантажувати та зберігати дані, моделі та результати.

Colab надає спеціальні форми (Colab Forms), які дозволяють створювати інтерактивні елементи управління для параметрів без написання коду інтерфейсу. Користувачі можуть змінювати значення через текстові поля, слайдери, випадаючі списки та прапорці, що робить блокноти більш зручними для нетехнічних користувачів. Інтеграція з TensorBoard дозволяє візуалізувати метрики навчання моделей у реальному часі безпосередньо в блокноті.

Можливість публікації блокнотів через GitHub або пряме посилання робить Colab ідеальним для поширення дослідницьких результатів, навчальних матеріалів або демонстрацій алгоритмів. Будь-хто з посиланням може відкрити блокнот, виконати код та відтворити результати, що значно підвищує відтворюваність досліджень. Багато наукових статей та курсів з машинного навчання надають супровідні Colab блокноти для практичних експериментів [18].

Проте Colab має суттєві обмеження. Найголовніше — обмежений час виконання: безкоштовні сесії автоматично закриваються після певного періоду неактивності (зазвичай 90 хвилин) або після максимального часу виконання (близько 12 годин). Це означає, що дуже довгі обчислення можуть бути перервані, а стан змінних втрачається. Хоча існує платна версія Colab Pro з довшими сесіями та кращими GPU, вона все одно має обмеження.

Ресурси GPU та TPU надаються на основі доступності, і немає гарантії їх отримання, особливо у пікові години. Безкоштовні користувачі можуть зіткнутися з чергами або обмеженнями на кількість часу використання прискорювачів. Продуктивність GPU може варіюватися в залежності від того, яка саме модель доступна в даний момент.

Відсутність традиційних IDE функцій, таких як налагодження з точками зупинки, рефакторинг коду, інтеграція з системами контролю версій (крім базового Git через команди терміналу), робить Colab менш зручним для розробки складних проєктів.

З огляду на те, що робота присвячена задачі класифікації даних, раціонально використовувати мову програмування Python, яка завдяки великій кількості бібліотек забезпечує широкі можливості для дослідження, аналізу та реалізації методів класифікації.

Python є однією з найпопулярніших мов програмування у сфері машинного навчання та обробки даних. Її популярність зумовлена не лише простим і зрозумілим синтаксисом, а й наявністю потужних бібліотек, таких як NumPy для чисельних обчислень, pandas для роботи з табличними даними, scikit-learn для реалізації алгоритмів машинного навчання, matplotlib і seaborn для візуалізації результатів, а також TensorFlow та PyTorch для створення глибоких нейронних мереж [9].

Крім того, Python має активну спільноту розробників і дослідників, що сприяє швидкому поширенню нових рішень та підтримці актуальних інструментів. Завдяки цьому Python є зручним середовищем як для прототипування моделей, так і для розробки повноцінних програмних рішень. У контексті курсової роботи це дозволяє зосередитися не на реалізації низькорівневих алгоритмів, а на дослідженні, порівнянні та вдосконаленні існуючих методів класифікації.

Також, для реалізації інформаційної технології було обрано платформу Kaggle.

Kaggle — це онлайн-платформа для змагань, навчання та співпраці у сфері аналізу даних і машинного навчання. Вона належить компанії Google і є одним із найпопулярніших ресурсів серед дослідників, аналітиків, дата-сайентістів та розробників, які хочуть удосконалювати свої навички, ділитися знаннями або змагатися з іншими [10].

Одна з головних особливостей Kaggle — це проведення відкритих змагань. Компанії або організації публікують реальні задачі (наприклад, передбачення

відтоку клієнтів або діагностика хвороби на основі медичних зображень), надаючи набір даних, а учасники з усього світу створюють моделі та змагаються за найкращі результати. Переможці часто отримують грошові призи, а також визнання в професійній спільноті.

Головна сторінка Kaggle показана на рисунку 2.4.

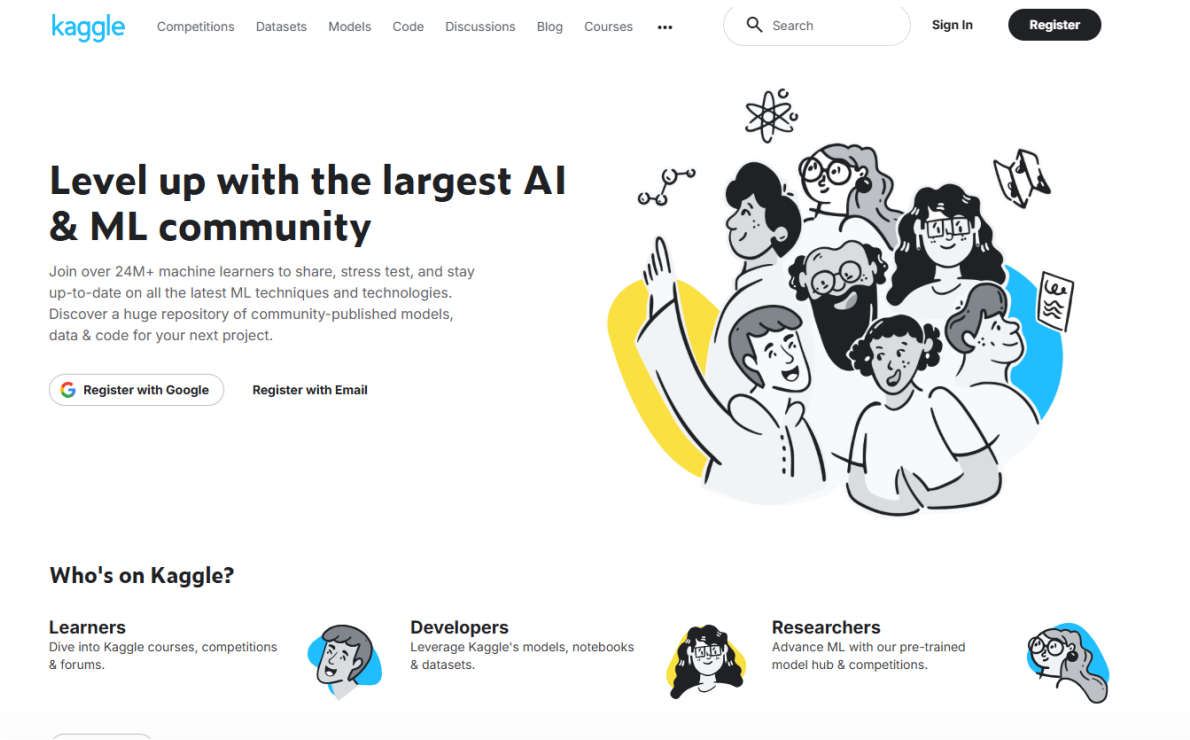

Рисунок 1.8 – Головна сторінка Kaggle

Крім змагань, Kaggle надає інструменти для навчання, зокрема інтерактивні курси з Python, машинного навчання, глибокого навчання, аналізу даних та інших тем. Початківці можуть спробувати свої сили в невеликих проєктах або kernel-ах (ноутбуках), які виконуються прямо в браузері без необхідності налаштовувати середовище на своєму комп’ютері.

Платформа також дозволяє ділитися власними наборами даних або використовувати чужі — в бібліотеці Kaggle зібрано десятки тисяч відкритих датасетів на будь-який смак: від фінансових звітів до зображень та текстів. Крім того, тут можна створювати та публікувати свої ноутбуки з кодом, поясненнями, візуалізаціями та результатами, які бачать інші користувачі.

Numpy — це фундаментальна бібліотека Python для наукових обчислень, яка забезпечує підтримку великих багатовимірних масивів і матриць, а також широкий набір математичних функцій для їх обробки. Вона є базою для багатьох інших бібліотек у сфері машинного навчання, аналізу даних та штучного інтелекту. Завдяки високій продуктивності і оптимізації на рівні C, Numpy дозволяє ефективно виконувати числові операції, що значно прискорює роботу з даними у порівнянні з традиційними структурами даних Python [17].

Крім того, Numpy має зручний інтерфейс для роботи з масивами, включаючи операції над усіма елементами, індексацію, маскування та трансформації. Це робить бібліотеку ідеальним інструментом для підготовки даних, чисельних розрахунків і наукових експериментів. Завдяки своїй гнучкості і потужності, Numpy широко використовується в академічних дослідженнях і промислових проєктах [11].

Pandas — це бібліотека для аналізу та обробки даних, що надає високорівневі структури даних, такі як DataFrame та Series, які значно полегшують роботу з табличними даними. Вона дозволяє легко імпортувати, очищати, фільтрувати, агрегувати і трансформувати дані, а також виконувати складні операції групування і злиття. Pandas активно використовується для обробки як невеликих наборів даних, так і великих датасетів, що робить її незамінним інструментом у світі аналізу даних.

Pandas також інтегрується з іншими популярними бібліотеками Python, такими як Numpy, Matplotlib і Scikit-learn, що забезпечує повний цикл роботи з даними — від їх підготовки до візуалізації і моделювання. Зручність використання і багатий функціонал роблять Pandas першочерговим вибором для аналітиків, дослідників і розробників, які працюють з даними у будь-якій галузі [12].

Matplotlib — це найбільш популярна бібліотека Python для створення графіків і візуалізації даних. Вона підтримує широкий спектр типів графіків, включно з лінійними, гістограмами, діаграмами розсіяння, тепловими картами та багатьма іншими. Matplotlib дозволяє створювати як прості, так і складні

візуалізації з високою деталізацією, налаштовуючи всі елементи графіків, такі як осі, легенди, підписи та кольори.

Однією з переваг Matplotlib є її гнучкість і інтеграція з іншими бібліотеками, що дозволяє використовувати її як базовий інструмент для наукових досліджень, машинного навчання та розробки інтерфейсів. Вона часто використовується разом із Pandas і Numpy для наочного представлення результатів обробки даних, що робить процес аналізу більш інтуїтивним і зрозумілим [13].

Модуль Datetime у Python забезпечує потужні інструменти для роботи з датами і часом. Він дозволяє легко створювати, форматувати, порівнювати і виконувати арифметичні операції над об'єктами часу. Це особливо важливо для обробки часових рядів, планування завдань, логування і роботи з часовими зонами. Модуль підтримує як прості операції з датами, так і складніші функції, такі як обчислення різниці між датами та конвертація між різними форматами.

Datatime широко використовується в різних сферах, починаючи від фінансової аналітики і закінчуючи розробкою веб-додатків, де коректне відображення і обробка часу має ключове значення. Завдяки простому API та гнучкості, цей модуль є невід’ємною частиною багатьох Python-проєктів, що працюють із часом [14].

2.2 Попередня підготовка даних

Перед тим, як розпочати розвідувальний аналіз даних, важливо забезпечити належну підготовку середовища для роботи. Це включає імпорт всіх необхідних бібліотек, які дозволяють ефективно обробляти, аналізувати та візуалізувати інформацію. На початковому етапі роботи було підключено набір інструментів, що є ключовими для виконання поставлених задач — вони представлені на рисунку 2.5.

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import style
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OrdinalEncoder
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
```

Рисунок 2.5 – Підготовка бібліотек

На наступному етапі було здійснено завантаження набору даних, після чого відобразили його частину для попереднього ознайомлення та перевірки коректності. Крім того, було отримано загальну інформацію про структуру датасету, що дозволило краще зрозуміти його склад та характерні особливості (див. рис. 2.6).

In [4]:

```
df
```

Out[4]:

	Unnamed: 0	Category	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT
0	1	0=Blood Donor	32	m	38.5	52.5	7.7	22.1	7.5	6.93	3.23	106.0	12.1	69.0
1	2	0=Blood Donor	32	m	38.5	70.3	18.0	24.7	3.9	11.17	4.80	74.0	15.6	76.5
2	3	0=Blood Donor	32	m	46.9	74.7	36.2	52.6	6.1	8.84	5.20	86.0	33.2	79.3
3	4	0=Blood Donor	32	m	43.2	52.0	30.6	22.6	18.9	7.33	4.74	80.0	33.8	75.7
4	5	0=Blood Donor	32	m	39.2	74.1	32.6	24.8	9.6	9.15	4.32	76.0	29.9	68.7
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
610	611	3=Cirrhosis	62	f	32.0	416.6	5.9	110.3	50.0	5.57	6.30	55.7	650.9	68.5
611	612	3=Cirrhosis	64	f	24.0	102.8	2.9	44.4	20.0	1.54	3.02	63.0	35.9	71.3
612	613	3=Cirrhosis	64	f	29.0	87.3	3.5	99.0	48.0	1.66	3.63	66.7	64.2	82.0
613	614	3=Cirrhosis	46	f	33.0	NaN	39.0	62.0	20.0	3.56	4.20	52.0	50.0	71.0
614	615	3=Cirrhosis	59	f	36.0	NaN	100.0	80.0	12.0	9.07	5.30	67.0	34.0	68.0

615 rows × 14 columns

Рисунок 2.6 – Аналіз обраного набору даних

Датасет містить лабораторні показники крові донорів та пацієнтів з гепатитом С, а також демографічні дані, такі як вік і стать. Джерелом даних є UCI Machine Learning Repository.

Усі атрибути, крім категорії (Category) та статі (Sex), є числовими. Перші чотири атрибути описують основну інформацію про пацієнта: унікальний ідентифікатор (ID), діагноз, вік у роках і стать (чоловіча або жіноча).

Наступні десять атрибутів — це лабораторні показники крові, які включають рівні таких речовин, як альбумін (ALB), лужна фосфатаза (ALP), аланінова та аспартатамінотрансферази (ALT, AST), білірубін (BIL), холінестераза (CHE), холестерин (CHOL), креатинін (CREA), гамма-глутамілтрансфераза (GGT) та загальний білок (PROT).

Цільовою змінною для задачі класифікації є атрибут Category, який розподіляє пацієнтів на кілька груп: здорові донори крові, підозрювані донори та різні стадії хвороби — гепатит, фіброз і цироз. Метою аналізу є розділення здорових донорів та пацієнтів із гепатитом С (включно з прогресуванням захворювання) на основі наявних даних.

Після загального огляду датасету, було проведено перевірку на пусті значення (рисунок 2.7).

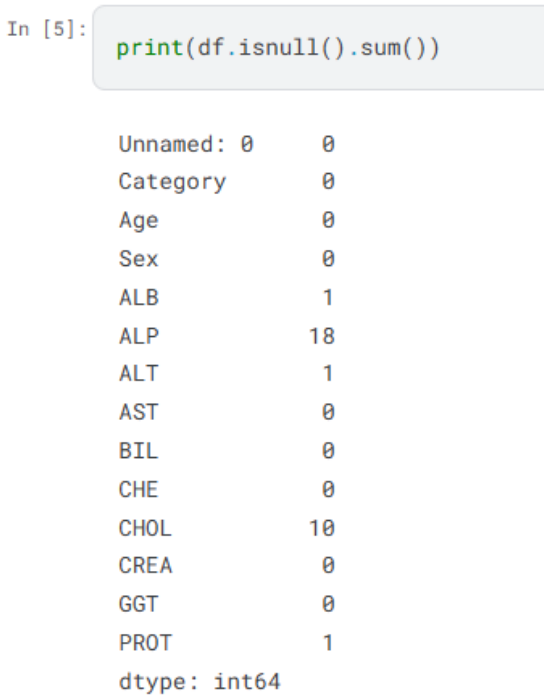

Рисунок 2.7 – Перевірка на пусті значення

Аналіз пропущених значень у DataFrame показав, що декілька стовпців містять відсутні дані. Повністю заповненими є стовпці Unnamed: 0, Category,

Age, Sex, AST, BIL, CHE, CREA та GGT. Водночас деякі стовпці мають пропуски різної кількості. Найбільше пропущених значень зафіксовано у стовпці ALP — їх налічується 18. По одному пропущеному значенню виявлено у стовпцях ALB, ALT та PROT. Крім того, стовпець CHOL містить 10 відсутніх значень. Виявлені пропуски потребують заповнення для забезпечення коректності подальшого аналізу. Ігнорування цих пропусків може призвести до неточностей у моделюванні та спотворення результатів. Тому було прийнято рішення застосувати метод заповнення пропущених значень середнім арифметичним, що дозволить зберегти узгодженість даних і підвищити якість моделі. Це важливий крок для підготовки датасету до ефективного навчання алгоритмів класифікації (рисунок 2.8).

In [6]:
```
df['ALB'].fillna(df['ALB'].mean(), inplace=True)
df['ALP'].fillna(df['ALP'].mean(), inplace=True)
df['CHOL'].fillna(df['CHOL'].mean(), inplace=True)
df['PROT'].fillna(df['PROT'].mean(), inplace=True)
df['ALT'].fillna(df['ALT'].mean(), inplace=True)
df = df.drop('Unnamed: 0', axis=1)

print(df.isnull().sum())
```

```
Category    0
Age         0
Sex         0
ALB         0
ALP         0
ALT         0
AST         0
BIL         0
CHE         0
CHOL        0
CREA        0
GGT         0
PROT        0
dtype: int64
```

Рисунок 2.8 – Заповнення порожніх значень

Таким чином було підготовано датасет до подальших досліджень, що дозволило забезпечити коректність і якість аналізу, уникнути помилок через

відсутні дані та підвищити точність побудованих моделей класифікації. Такий підхід сприяв більш стабільним і надійним результатам у процесі навчання алгоритмів. Завдяки цьому можна було перейти до детального розгляду методів класифікації та їх застосування на підготовлених даних.

2.3 Розвідувальний аналіз

Розвідувальний аналіз даних (Exploratory Data Analysis, EDA) — це перший етап роботи з набором даних, під час якого дослідник знайомиться з даними, вивчає їх основні властивості та структуру. Мета цього етапу — зрозуміти, які дані є, виявити закономірності, аномалії, відсутні або некоректні значення, а також встановити взаємозв'язки між різними змінними.

Під час розвідувального аналізу застосовують різні статистичні методи та візуалізаційні інструменти, такі як гістограми, діаграми розсіювання, коробкові діаграми, кореляційні матриці тощо. Це допомагає сформувати гіпотези про структуру даних і визначити, які методи подальшої обробки і моделювання будуть найбільш ефективними.

На рисунку 2.9 приведено код для побудови графіка розподілу значень.

In [11]:
```
import matplotlib.pyplot as plt

# create a list of the columns to plot
columns_to_plot = ['ALB', 'ALP', 'ALT', 'AST', 'BIL', 'CHE', 'CHOL', 'CREA', 'GGT', 'PROT']

# create a box plot for each column
plt.figure(figsize=(10,6))
plt.boxplot(df[columns_to_plot].values, labels=columns_to_plot, showfliers=True)
plt.xticks(rotation=45)
plt.show()
```

Рисунок 2.9 – Код для побудови графіка розподілу значень

На рисунку 2.10 показано графік що відображає розподіл значень показників.

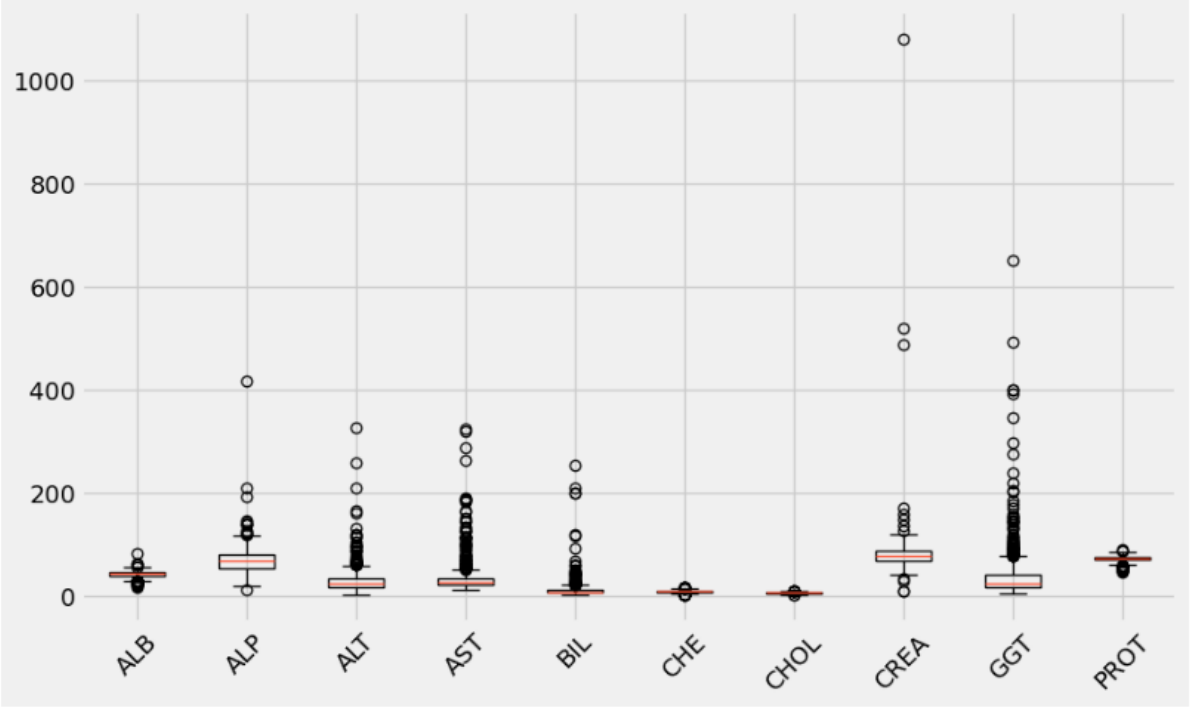

Рисунок 2.10 – Графік розподілу значень

Аналіз графіку boxplot показав розподіл біохімічних показників у датасеті. Більшість змінних, таких як ALB та PROT мають відносно невеликий розкид значень. Водночас показники ALP, GGT та CREA містять значну кількість викидів — точок, які виходять за межі «вусів» скриньки. Особливо виразні викиди спостерігаються у змінних CREA і GGT, де значення перевищують 600–1000, що може свідчити як про можливі патології, так і про технічні помилки у даних. Для подальшого аналізу необхідно перевірити причини виникнення цих викидів, а також розглянути застосування логарифмічного перетворення або обрізки даних для кращої нормалізації розподілу. Врахування цих особливостей є важливим при побудові статистичних моделей, оскільки це допоможе знизити вплив аномальних значень на результати. Графік також підкреслює важливість попередньої обробки даних, включно із заповненням пропусків, щоб уникнути можливих спотворень через аномалії.

Далі було проведено перевірку на викиди у наборі даних (рисунок 2.11).

In [14]:
```
q_low = df[col].quantile(0.01)
q_hi  = df[col].quantile(0.99)

df_outliers = df[(df[col] < q_low) | (df[col] > q_hi)]
outlier_percentage = (df_outliers.sum() / len(df_outliers)) * 100
print(outlier_percentage)
```

```
Category      0.000000
Age         103.089431
Sex           0.000000
ALB           0.040650
ALP           3.701887
ALT          10.292243
AST          21.351176
BIL          30.486427
CHE           0.115138
CHOL          0.790912
CREA         14.107627
GGT          17.872905
PROT         -1.871251
dtype: float64
```

Рисунок 2.11 – Аналіз викидів у даних

Результати свідчать про найбільшу кількість викидів у стовпцях GGT, CREA, AST та ALP, де значення перевищують типові межі в кілька разів. Менш критичні, але все ж помітні викиди виявлені в ALT, BIL та PROT. Стовпці CHOL та CHE мають незначну кількість аномальних значень, тоді як Category та Sex не містять викидів, що пояснюється їх категоріальним характером. Відсотки, більші за 100%, свідчать про наявність дуже екстремальних значень, які можуть бути спричинені як помилками у даних, так і реальними медичними станами, наприклад, нирковою недостатністю у випадку CREA.

Для зменшення кількості викидів було проведено масштабування даних за допомогою RobustScaler (рисунок 2.12).

In [14]:
```
q_low = df[col].quantile(0.01)
q_hi  = df[col].quantile(0.99)

df_outliers = df[(df[col] < q_low) | (df[col] > q_hi)]
outlier_percentage = (df_outliers.sum() / len(df_outliers)) * 100
print(outlier_percentage)
```

```
Category      0.000000
Age         103.089431
Sex           0.000000
ALB           0.040650
ALP           3.701887
ALT          10.292243
AST          21.351176
BIL          30.486427
CHE           0.115138
CHOL          0.790912
CREA         14.107627
GGT          17.872905
PROT         -1.871251
dtype: float64
```

Рисунок 2.12 – Масштабування даних

Цей метод застосовано до біохімічних показників ALB, ALP, ALT, AST, BIL, CHE, CHOL, CREA, GGT та PROT для зменшення впливу екстремальних значень. Після масштабування викиди було визначено як значення поза межами 1-го та 99-го перцентилів, і для кожного стовпця розраховано їх відсоток. Результати показали значне зменшення кількості викидів порівняно з початковими даними. Наприклад, ALB, CHE та CHOL майже не містять аномальних значень (близько 0%), що свідчить про ефективність застосованого масштабування.

Помірну кількість викидів зберегли ALP, ALT, CREA та GGT, проте їх частка суттєво знизилася. Водночас AST та BIL мають порівняно високий відсоток викидів, що може відображати реальні медичні аномалії або залишковий шум у даних. Незвичним є негативне значення викидів для PROT, що вказує на можливі помилки у розрахунках або особливості застосованого

методу масштабування. Загалом, RobustScaler продемонстрував високу ефективність у зменшенні впливу викидів, забезпечуючи більш стабільні та надійні дані для подальшого аналізу.

Наступним кроком було проведено розподіл даних за віком. На рисунку 2.13 приведено код для здійснення побудови гістограми розподілу даних за віком, а на рисунку 2.14 приведено саму гістограму розподілу даних за віком.

```
In [15]:
import matplotlib.pyplot as plt

plt.hist(df['Age'])
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```

Рисунок 2.13 – Код для здійснення побудови гістограми розподілу даних за віком

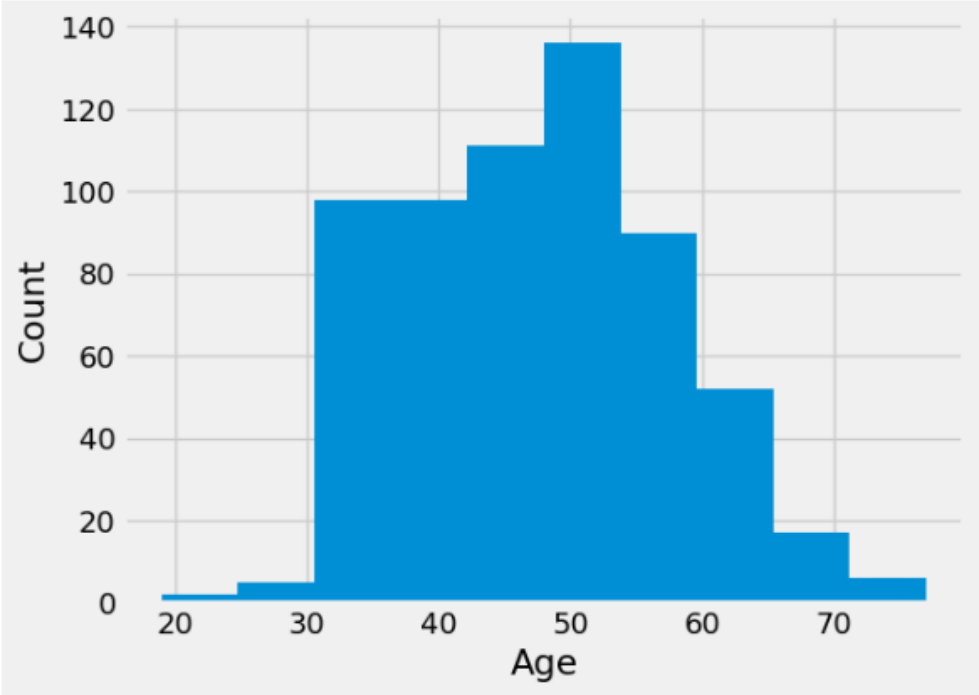

Рисунок 2.14 – Розподіл даних за віком

Графік демонструє розподіл кількості людей за віком, де найбільша концентрація спостерігається приблизно у віці 50 років — близько 140 осіб.

Розподіл має форму дзвоноподібної кривої з піком на цьому віковому значенні, яка поступово зменшується до меж вікових груп 20 та 70 років. Значна частина вибірки, понад 100 осіб, зосереджена у віковому діапазоні від 40 до 60 років, тоді як молодші за 30 і старші за 70 років групи представлені менш ніж 20 особами. Загалом розподіл відповідає нормальному, із чітко вираженим центральним піком навколо 50 років, що свідчить про найбільш поширений вік у досліджуваній вибірці.

На рисунку 2.15 приведено код для побудови кругової діаграми розподілу за статтю, а на рисунку 2.16 показано саму діаграму розподілу за статтю.

```
fig, ax = plt.subplots(figsize=(8,8))
plt.pie(x=df["Sex"].value_counts(),
        colors=["blue","red"],
        labels=["Male","Female"],
        autopct="%1.2f%%",
        )
plt.show()
```

Рисунок 2.15 – Код для побудови кругової діаграми розподілу за статтю

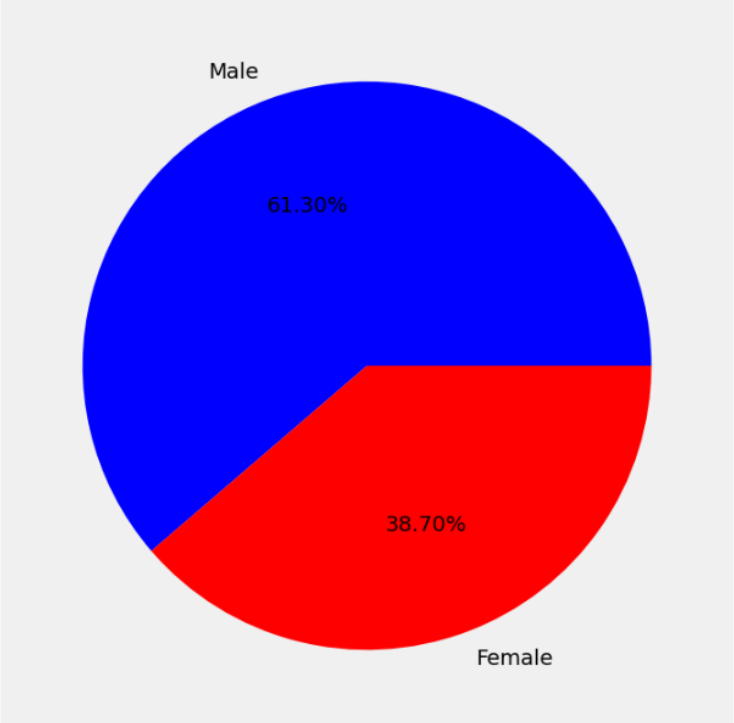

Рисунок 2.16 – Діаграма розподілу за статтю

На рисунку 2.16 показано, що більша частина значень в датасеті припадає на чоловіків, 61.3%, коли на жінок припадає тільки 38.7%.

На рисунку 2.17 показано гістограму розподілу за віком.

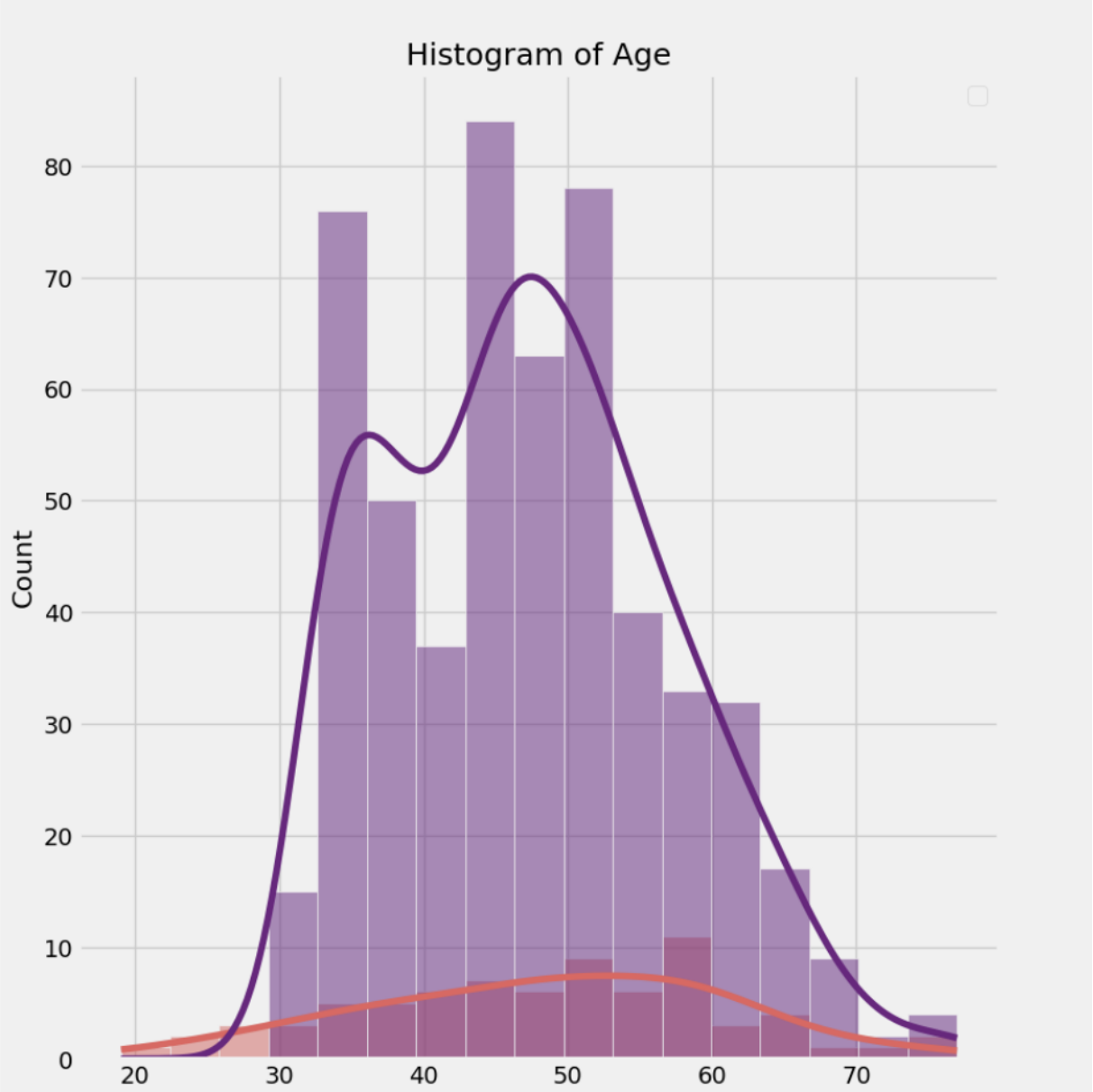

Рисунок 2.17 – Гістограма розподілу за віком

Гістограма віку показує, що найбільша концентрація людей припадає на вікові групи 40-50 років, з піком близько 50 років. Розподіл має тенденцію до симетричного спаду як у молодшій, так і в старшій вікових групах. На рисунку 2.18 показано гістограму розподілу за ALB.

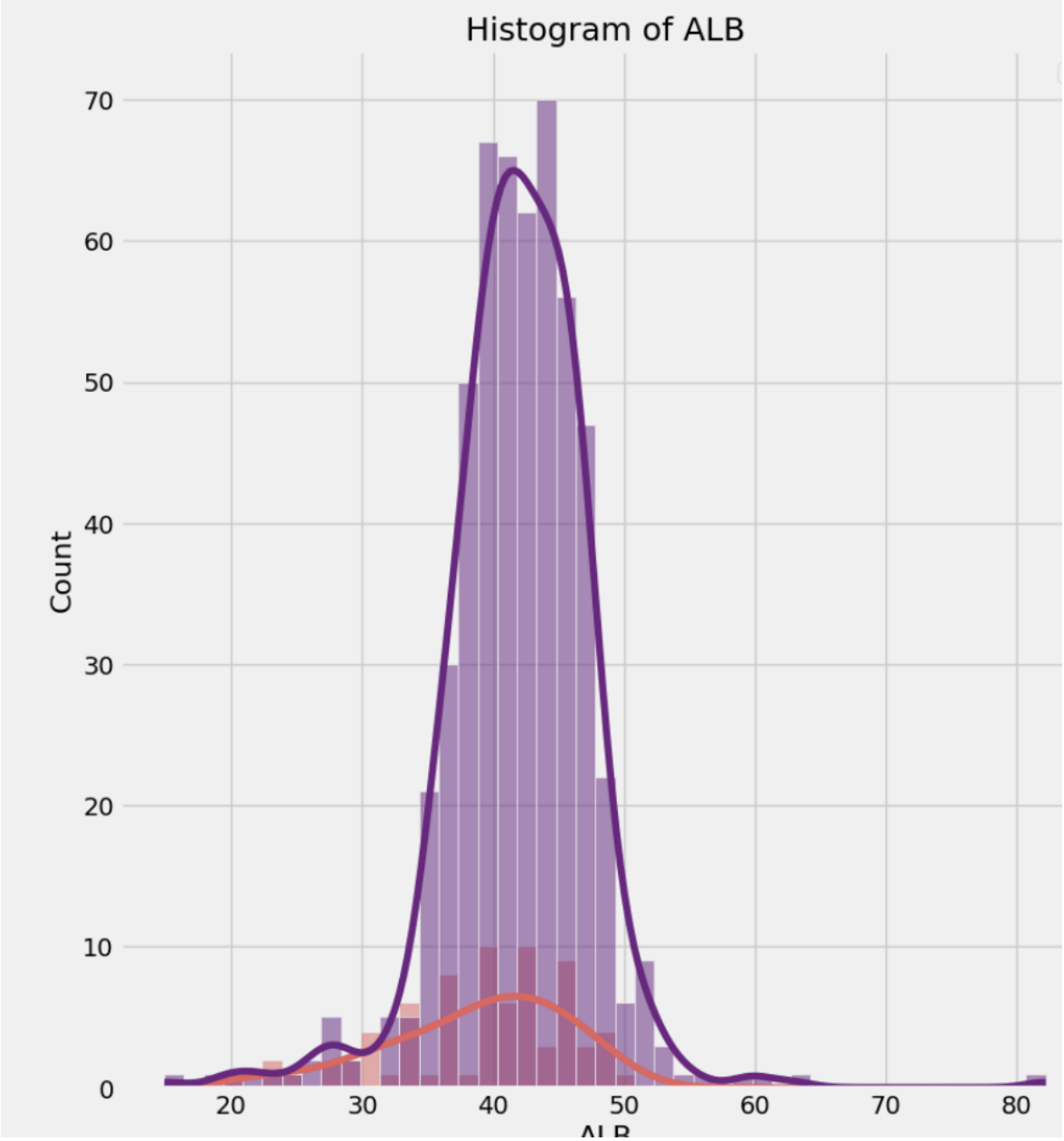

Рисунок 2.18 – Гістограма розподілу за ALB

Гістограма ALB показує, що найбільша концентрація значень припадає на діапазон 40-50 одиниць, з піком кількості близько 60-70. Фіолетова крива підтверджує симетричний розподіл із центром у цьому діапазоні, тоді як оранжева крива вказує на додатковий розподіл із подібним піком, що поступово знижується. На рисунку 2.19 показано гістограму розподілу за ALP.

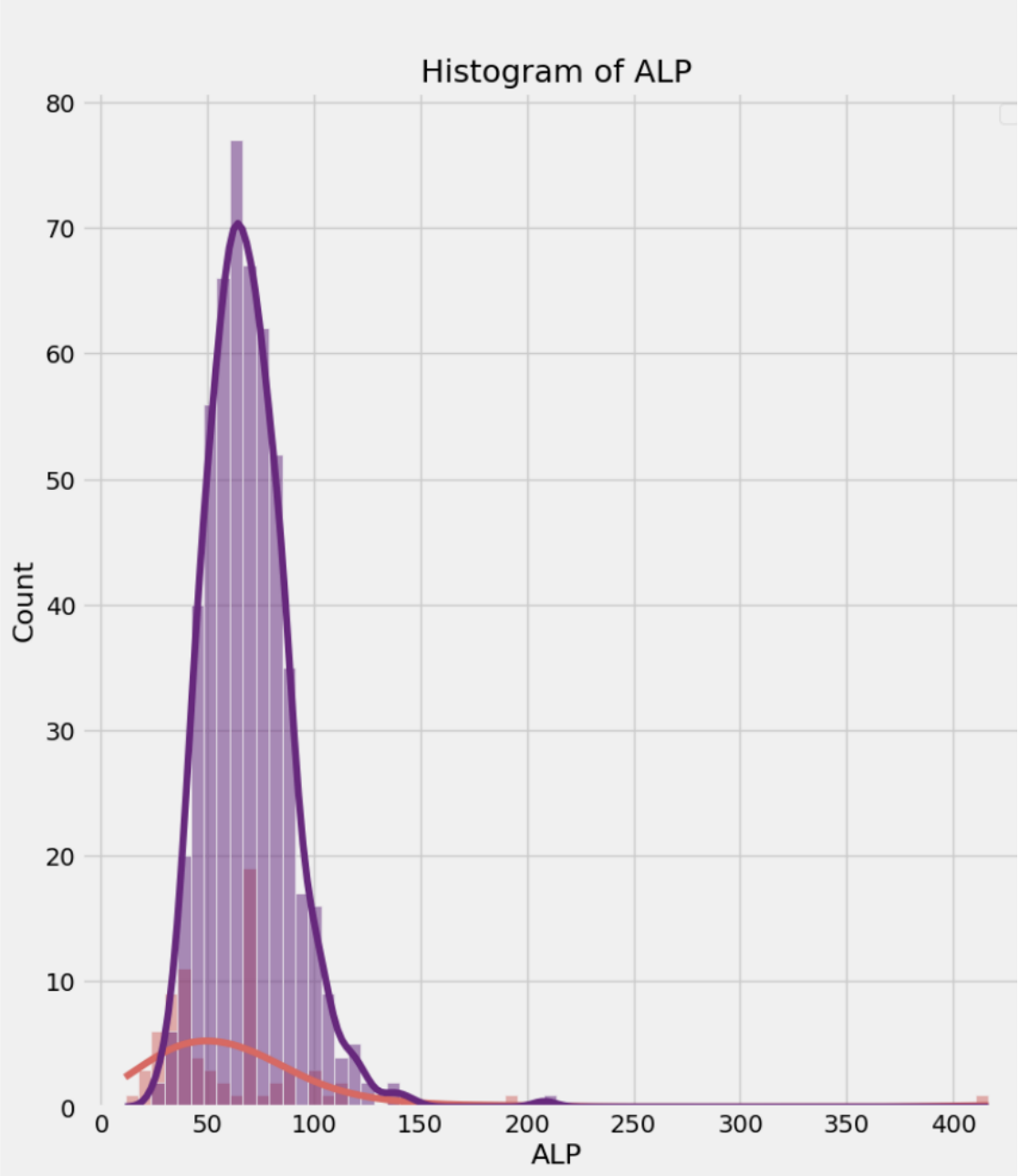

Рисунок 2.19 – Гістограма розподілу за ALP

Гістограма ALP показує, що найбільша концентрація значень припадає на діапазон 0-50 одиниць, з піком кількості близько 70-80. Розподіл має різкий спад після піку, з мінімальними значеннями після 200 одиниць. Оранжева крива вказує на додатковий розподіл із меншою амплітудою, що поступово знижується після 50 одиниць. На рисунку 2.20 показано гістограму розподілу за ALT.

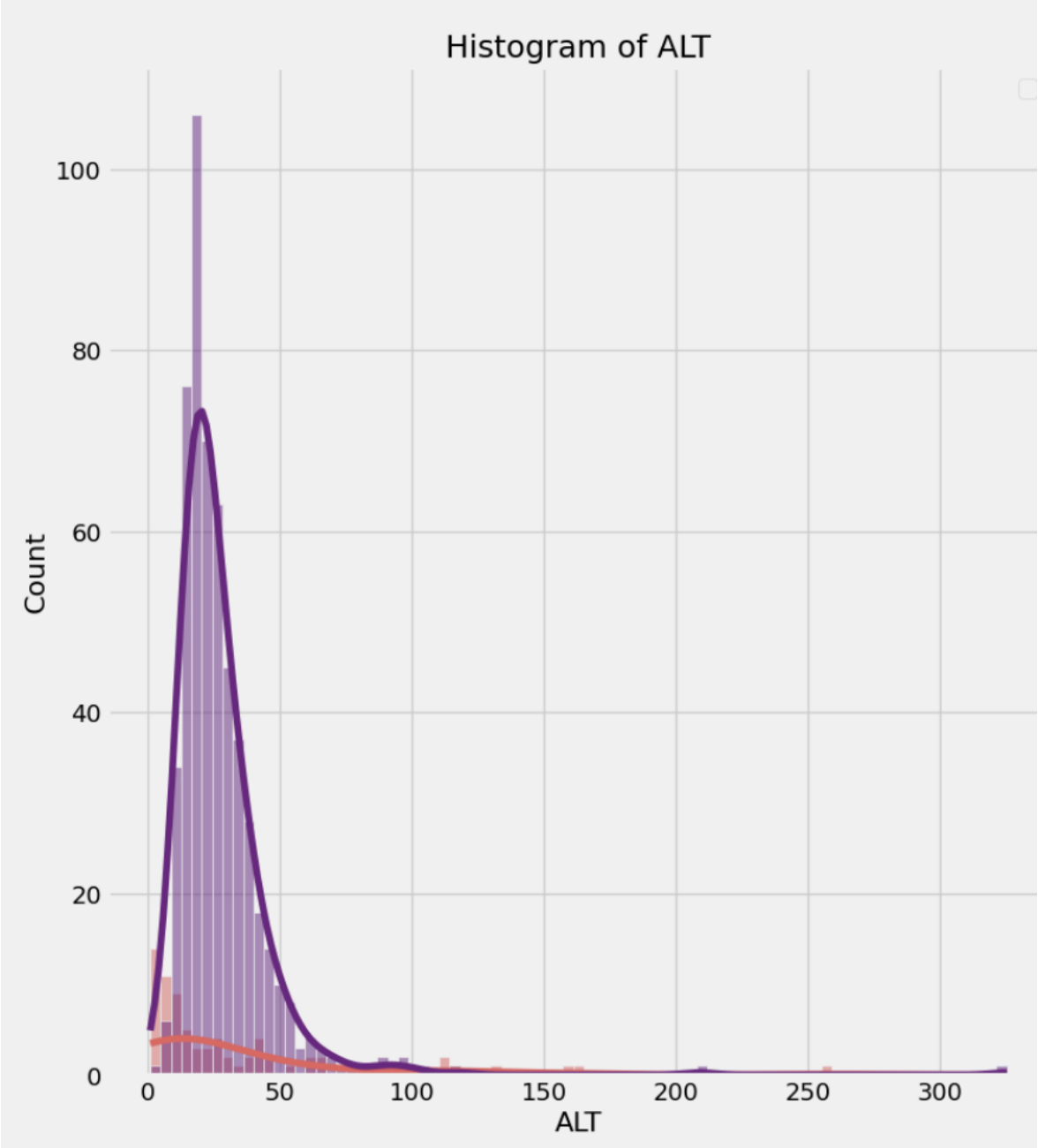

Рисунок 2.20 – Гістограма розподілу за ALT

Гістограма ALT показує, що найбільша концентрація значень припадає на діапазон 0-50 одиниць, з піком кількості близько 80-100. Розподіл різко спадає після 50 одиниць, з мінімальними значеннями після 150 одиниць. Оранжева крива вказує на додатковий розподіл із меншою амплітудою, що поступово знижується після початкового підйому. На рисунку 2.21 приведено код для реалізації матриці кореляції, а на рисунку 2.22 показано результат побудови матриці кореляції.

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.set(style="whitegrid")
correlation_matrix = df.corr()
fig, ax = plt.subplots(figsize=(12, 10))
plt.title("Correlation Matrix Heatmap", fontsize=16)
sns.heatmap(correlation_matrix, annot=True, annot_kws={"size": 12}, cmap='coolwarm', ax=ax)
cbar = ax.collections[0].colorbar
cbar.ax.tick_params(labelsize=12)
cbar.set_label('Correlation Strength', rotation=270, fontsize=14, labelpad=15)
plt.show()
```

Рисунок 2.21 – Код для реалізації матриці кореляції

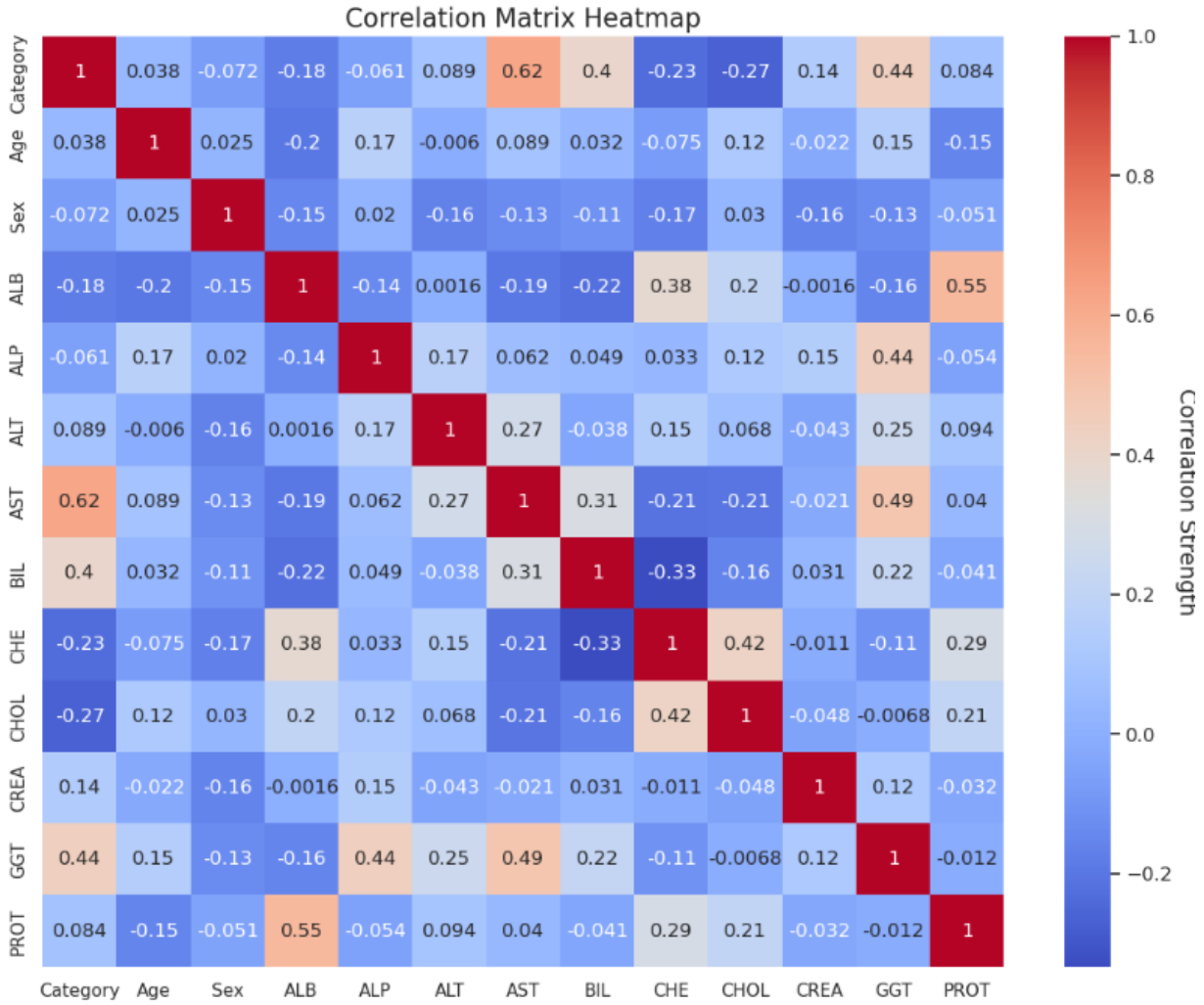

Рисунок 2.22 – Матриця кореляції показників

Матриця кореляції демонструє взаємозв'язки між різними атрибутами пацієнтів та лабораторними показниками, де коефіцієнти кореляції варіюються

від - 1.0 (сильна негативна залежність) до 1.0 (сильна позитивна залежність). Виявлено, що діагноз (Category) має помірну позитивну кореляцію з такими показниками, як AST (0.62) та GGT (0.44), що свідчить про їхній зв'язок із хворобою. Альбумін (ALB) показує сильну кореляцію з PROT (0.55), що підкреслює їхню тісну біохімічну взаємодію. Печінкові ферменти AST та ALT також мають помірну позитивну кореляцію (0.27), підтверджуючи спільну функціональну роль. Білірубін (BIL) пов'язаний позитивно з AST (0.31), але негативно корелює з холінестеразою (CHE) (-0.33), що може вказувати на різні механізми дії. Холестерин (CHOL) помірно корелює з креатиніном (CHE) (0.42) і має слабкі зв'язки з іншими показниками. Демографічні змінні, такі як вік (Age) і стать (Sex), мають загалом слабкі кореляції з лабораторними даними, що свідчить про їх менший вплив у порівнянні з біохімічними показниками. Таким чином, матриця кореляції підкреслює важливість лабораторних параметрів для подальшого аналізу і моделювання.

2.4 Висновки

Проведено аналіз і вибір методів та моделей класифікації, серед яких були описані вище алгоритми, зокрема Random Forest, Support Vector Machine, Logistic Regression та Gradient Boosting.

Також, у цьому розділі виконано комплексну підготовку даних та проведено розвідувальний аналіз, спрямований на виявлення ключових факторів, що впливають на діагноз пацієнта. Завдяки аналізу було ідентифіковано цільові групи захворювання та основні змінні, які підвищують ризик виникнення патології. Це дозволяє глибше зрозуміти взаємозв'язки між біохімічними показниками та станом здоров'я пацієнтів, а також закладає основу для подальшого моделювання та класифікації.

3 РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ АНАЛІЗУ ТА ПЕРЕДБАЧЕННЯ СТАНУ ХВОРИХ НА ГЕПАТИТ

Основою інформаційної технології аналізу та передбачення стану хворих на гепатит є реалізація алгоритму побудови та вдосконалення моделей машинного навчання (рисунок 3.1).

Алгоритм починається з етапу ініціалізації процесу, який позначений блоком «Початок». Після цього виконується імпорт необхідних бібліотек, що містять функції для роботи з даними, побудови моделей, оцінювання результатів та візуалізації. Цей етап забезпечує інструментальну основу для подальших дій.

Наступним кроком є імпорт датасету, тобто завантаження даних із зовнішнього джерела — файлу, бази даних або мережевого ресурсу. Отримані дані стають основою для подальшого аналізу. Далі здійснюється розвідувальний аналіз даних (EDA – Exploratory Data Analysis), під час якого відбувається початкове ознайомлення з даними, виявлення закономірностей, наявності пропусків або аномалій, а також побудова базових статистичних характеристик і візуалізацій.

Після аналізу настає етап обробки даних, який включає очищення від пропусків, перетворення текстових змінних у числові, масштабування числових ознак, а також формування навчальної та тестової вибірок. Ця стадія є критично важливою, оскільки якість підготовлених даних безпосередньо впливає на результати роботи моделей.

Далі виконується побудова моделей машинного навчання. На цьому етапі застосовуються різні алгоритми (наприклад, логістична регресія, дерева рішень, випадкові ліси, градієнтний бустинг тощо). Моделі навчаються на підготовленій навчальній вибірці, а потім використовуються для прогнозування на тестових даних.

Після побудови моделей здійснюється оцінка та порівняння результатів. Для цього обчислюються метрики точності, повноти, або інші показники, які дозволяють визначити, наскільки добре модель справляється з поставленим завданням.

На наступному етапі проводиться перевірка умовної гілки — «Точність задовільна?». Якщо результати моделі є задовільними, алгоритм переходить до завершення процесу («Кінець»). Якщо ж точність є недостатньою, виконується гілка «Ні», де передбачено зміну параметрів моделі та додаткову обробку даних. Це може включати налаштування гіперпараметрів, додавання або видалення ознак, покращення балансу класів, а також використання більш потужних алгоритмів. Після цього цикл повертається до етапу побудови моделей для повторного навчання.

Таким чином, алгоритм є ітераційним процесом, що передбачає поступове вдосконалення моделі шляхом багаторазового повторення етапів навчання, оцінки та налаштування. Такий підхід дозволяє досягти максимальної точності прогнозування та оптимальної узагальнювальної здатності моделей машинного навчання.

На рисунку 3.1 приведено саме узагальнений алгоритм, всі специфічні моменти опрацювання даних та налаштування моделей буде реалізовано безпосередньо в програмному коді.

В даному узагальненому алгоритмі відображено основні етапи реалізації інформаційної технології:

1. Імпорт та налаштування необхідних бібліотек.
2. Імпорт набору даних.
3. Розвідувальний аналіз даних.
4. Обробка даних.
5. Побудова моделей та оцінка їх точності.
6. Зміна параметрів моделей для покращення результатів.

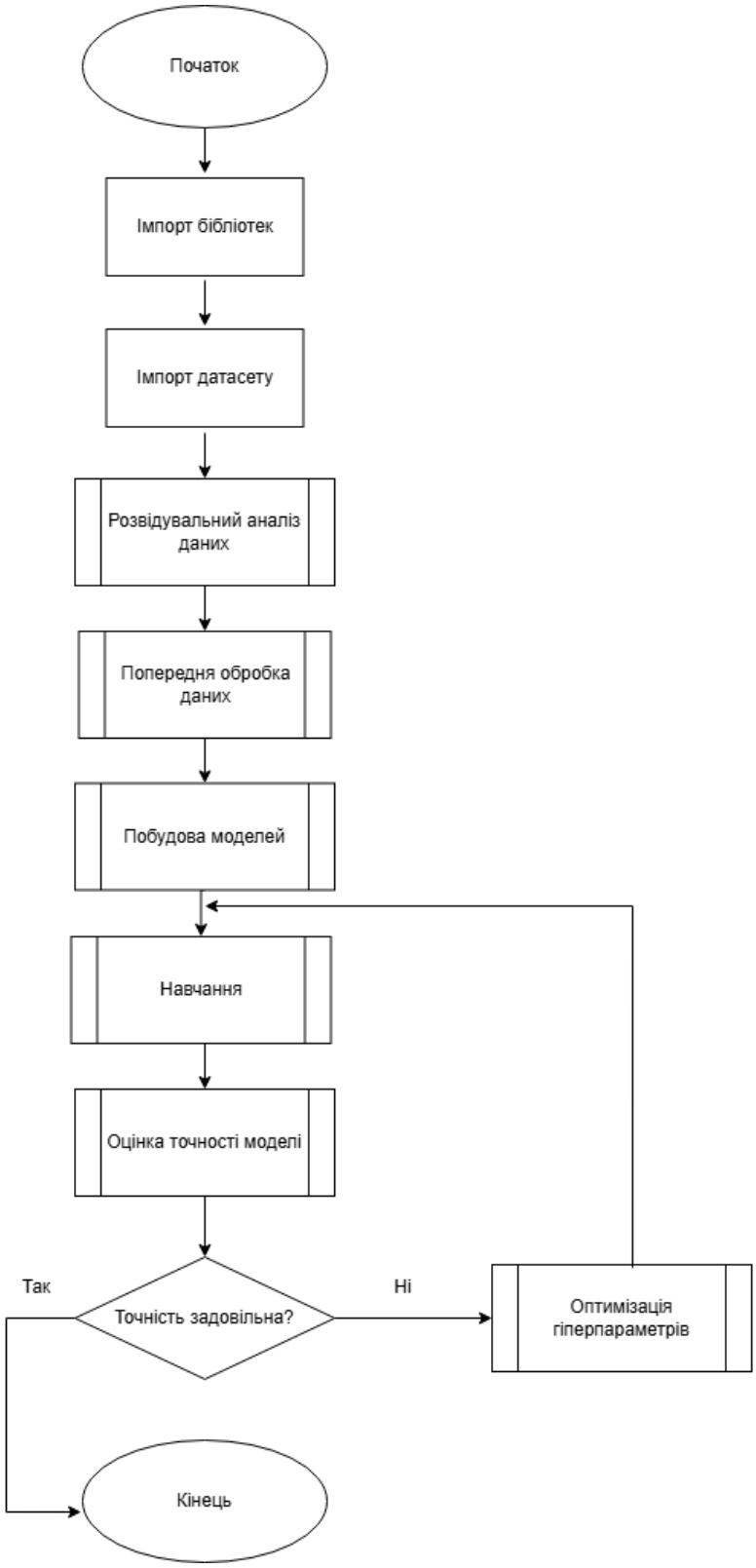

Рисунок 3.1 – Узагальнений алгоритм інформаційної технології

3.1 Підготовка набору даних до машинного навчання

Розбивання даних на тренувальний і тестовий набори потрібно для того, щоб перевірити, наскільки добре модель навчається і як вона буде працювати на

нових, раніше невідомих даних. Тренувальний набір використовується для навчання моделі — вона “вивчає” закономірності та залежності. Тестовий набір служить для оцінки якості моделі: він показує, наскільки точно модель передбачає правильні результати на даних, яких вона раніше не бачила. Це допомагає уникнути перенавчання (overfitting), коли модель занадто точно запам’ятовує тренувальні дані, але погано працює на нових. Таким чином, розбивання даних підвищує надійність і загальну якість моделі.

У результаті підготовки даних було здійснено розподіл вибірки на вхідні ознаки та цільову змінну. Для подальшого навчання моделей дані було розділено на тренувальний та тестовий набори у пропорції 80% до 20%. Це дозволить оцінити ефективність класифікації на нових, невідомих модифікованих даних та забезпечити коректну перевірку моделей. Використання фіксованого параметра random_state гарантує відтворюваність результатів експерименту (рисунок 3.2).

In [22]:
```
from sklearn.model_selection import train_test_split

X = df.drop("Category", axis=1)
y = df["Category"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [23]:
```
X
```

Out[23]:

	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT
0	32	0	-0.531250	-0.538899	-0.918919	-0.336283	0.033898	-0.500942	-1.449477	1.380952	-0.457143	-0.52
1	32	0	-0.531250	0.136622	-0.300300	-0.106195	-0.576271	1.096045	-0.355401	-0.142857	-0.314286	0.704
2	32	0	0.781250	0.303605	0.792793	2.362832	-0.203390	0.218456	-0.076655	0.428571	0.404082	1.163
3	32	0	0.203125	-0.557875	0.456456	-0.292035	1.966102	-0.350282	-0.397213	0.142857	0.428571	0.573
4	32	0	-0.421875	0.280835	0.576577	-0.097345	0.389831	0.335217	-0.689895	-0.047619	0.269388	-0.57
...	...	...	...	...	...	...	...	...	...	...	...	...
610	62	1	-1.546875	13.278937	-1.027027	7.469027	7.237288	-1.013183	0.689895	-1.014286	25.616327	-0.60
611	64	1	-2.796875	1.370019	-1.207207	1.637168	2.152542	-2.531073	-1.595819	-0.666667	0.514286	-0.14
612	64	1	-2.015625	0.781784	-1.171171	6.469027	6.898305	-2.485876	-1.170732	-0.490476	1.669388	1.606
613	46	1	-1.390625	0.060111	0.960961	3.194690	2.152542	-1.770245	-0.773519	-1.190476	1.089796	-0.19
614	59	1	-0.921875	0.060111	4.624625	4.787611	0.796610	0.305085	-0.006969	-0.476190	0.436735	-0.68

Рисунок 3.2 – Розбиття датасету на тренувальний та тестовий

Наступним кроком було проведено побудову та тюнінг моделей для визначення найкращих гіперпараметрів (рисунки 3.3–3.4).

```
models = [
    {
        "name": "Logistic Regression",
        "estimator": LogisticRegression(),
        "hyperparameters": {
            "penalty": ["l2"],
            "C": [0.01, 0.1, 1, 10],
            "max_iter": [500]
        }
    },
    {
        "name": "Random Forest",
        "estimator": RandomForestClassifier(),
        "hyperparameters": {
            "n_estimators": [100, 200, 300],
            "max_depth": [5, 10, 20, None]
        }
    },
    {
        "name": "Gradient Boosting",
        "estimator": GradientBoostingClassifier(),
        "hyperparameters": {
            "n_estimators": [100, 200, 300],
            "learning_rate": [0.01, 0.1, 1],
            "max_depth": [3, 5, 10]
        }
    },
    {
        "name": "Support Vector Machine",
        "estimator": SVC(),
        "hyperparameters": {
            "C": [0.01, 0.1, 1, 10],
            "kernel": ["linear", "rbf", "sigmoid"],
            "gamma": ["scale", "auto"]
        }
    }

]
```

Рисунок 3.3 – Побудова моделей машинного навчання

Було визначено чотири моделі машинного навчання для тренування та оптимізації на основі різних алгоритмів класифікації: логістична регресія, випадковий ліс, градієнтний бустинг та метод опорних векторів. Для кожної моделі було задано набір гіперпараметрів, які підлягають налаштуванню з метою підвищення точності прогнозів.

Застосування різних алгоритмів і варіантів налаштувань дозволяє порівняти їхню ефективність та обрати найбільш оптимальне рішення для конкретної задачі класифікації.

```
# train and tune each model
accuracies = []
best_models = {}
for model in models:
    with warnings.catch_warnings():
        warnings.simplefilter("ignore")
        print(f"Training {model['name']}...")
        grid_search = GridSearchCV(
            estimator=model['estimator'],
            param_grid=model['hyperparameters'],
            scoring='accuracy',
            cv=5
        )
        grid_search.fit(X_train, y_train)

        # evaluate the model's performance
        best_model = grid_search.best_estimator_
        y_pred = best_model.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        conf_matrix = confusion_matrix(y_test, y_pred)

        accuracies.append((model['name'], accuracy))
        best_models[model['name']] = best_model

        print(f"Best parameters for {model['name']}: {grid_search.best_params_}")
        print(f"Accuracy for {model['name']}: {accuracy}")
```

Рисунок 3.4 – Тренування моделей

Результати навчання моделей на тренувальних даних показано на рисунку 3.5.

```
Training Logistic Regression...
Best parameters for Logistic Regression: {'C': 10, 'max_iter': 500, 'penalty': 'l2'}
Accuracy for Logistic Regression: 0.9024390243902439
Training Random Forest...
Best parameters for Random Forest: {'max_depth': 10, 'n_estimators': 200}
Accuracy for Random Forest: 0.926829268292683
Training Gradient Boosting...
Best parameters for Gradient Boosting: {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 300}
Accuracy for Gradient Boosting: 0.9349593495934959
Training Support Vector Machine...
Best parameters for Support Vector Machine: {'C': 10, 'gamma': 'scale', 'kernel': 'linear'}
Accuracy for Support Vector Machine: 0.9024390243902439
```

Рисунок 3.5 – Результат тренування моделей

Результати навчання та налаштування моделей показали високу ефективність різних алгоритмів у задачі класифікації. Найкращі показники точності досягнуто за допомогою Gradient Boosting, який продемонстрував точність близько 93.5%. Це свідчить про високу здатність моделі узагальнювати інформацію та точно класифікувати нові дані.

Random Forest також показав сильні результати з точністю понад 92.6%, підтверджуючи свою надійність у роботі з комплексними та різнорідними даними завдяки ансамблевому підходу.

Logistic Regression і Support Vector Machine показали однакову точність близько 90.2%, що є добрим результатом для моделей із порівняно простішою архітектурою. Вони можуть бути корисні, коли важлива швидкість навчання або інтерпретованість моделі.

Оптимальні гіперпараметри для кожної моделі були знайдені за допомогою пошуку по сітці, що дозволило покращити продуктивність і уникнути перенавчання. Загалом, отримані результати підтверджують ефективність застосованих методів та відкривають можливість для подальшого використання цих моделей у практичних медичних задачах класифікації.

3.2 Машинне навчання

У результаті налаштування та порівняння моделей було сформовано остаточні версії кожного класифікатора з найкращими підібраними гіперпараметрами (рис. 3.6).

```
# create the Logistic Regression model with the best hyperparameters
log_reg_model = LogisticRegression(
    C=10,
    max_iter=500,
    penalty='l2'
)

# create the Random Forest model with the best hyperparameters
rf_model = RandomForestClassifier(
    max_depth=10,
    n_estimators=300
)

# create the Gradient Boosting model with the best hyperparameters
gb_model = GradientBoostingClassifier(
    learning_rate=0.1,
    max_depth=3,
    n_estimators=100
)

# create the Support Vector Machine model with the best hyperparameters
svm_model = SVC(
    C=10,
    gamma='scale',
    kernel='linear'
)
```

Рисунок 3.6 – Підготовка моделей

Для логістичної регресії оптимальною виявилась регуляризація типу L2 з параметром C, рівним 10, та максимальною кількістю ітерацій 500. Модель випадкового лісу показала найкращу продуктивність при використанні 300 дерев з максимальною глибиною 10. Градієнтний бустинг досяг високої точності за умов кількості базових моделей 100, глибини дерев 3 та швидкості навчання 0.1. Для методу опорних векторів найкращим варіантом виявилося використання лінійного ядра з параметром C, рівним 10, та автоматичним масштабуванням

gamma. Визначення цих конфігурацій дало змогу підготувати моделі до фінального етапу аналізу ефективності на тестових даних.

У рамках завершального етапу моделі були натреновані на навчальних даних і застосовані для передбачення на тестовій вибірці. Кожен з чотирьох алгоритмів — логістична регресія, випадковий ліс, градієнтний бустинг та метод опорних векторів — пройшов процес навчання з використанням раніше підібраних найкращих гіперпараметрів (рис. 3.7).

In [26]:
```
# train the Logistic Regression model on the training data
log_reg_model.fit(X_train, y_train)

# make predictions on the testing data
y_pred_log_reg = log_reg_model.predict(X_test)

# train the Random Forest model on the training data
rf_model.fit(X_train, y_train)

# make predictions on the testing data
y_pred_rf = rf_model.predict(X_test)

# train the Gradient Boosting model on the training data
gb_model.fit(X_train, y_train)

# make predictions on the testing data
y_pred_gb = gb_model.predict(X_test)

# train the Support Vector Machine model on the training data
svm_model.fit(X_train, y_train)

# make predictions on the testing data
y_pred_svm = svm_model.predict(X_test)
```

Рисунок 3.7 – Підготовка моделей

Далі було реалізовано візуалізацію результатів класифікації шляхом побудови матриць неточностей для кожної з чотирьох моделей: логістичної регресії, випадкового лісу, градієнтного бустингу та методу опорних векторів. Для кожної моделі було створено окрему матрицю, яка демонструє

співвідношення правильних і помилкових передбачень. Візуалізація виконувалась у вигляді підграфіків на одній фігурі, що дозволяє зручно порівняти ефективність кожного алгоритму (рис. 3.8).

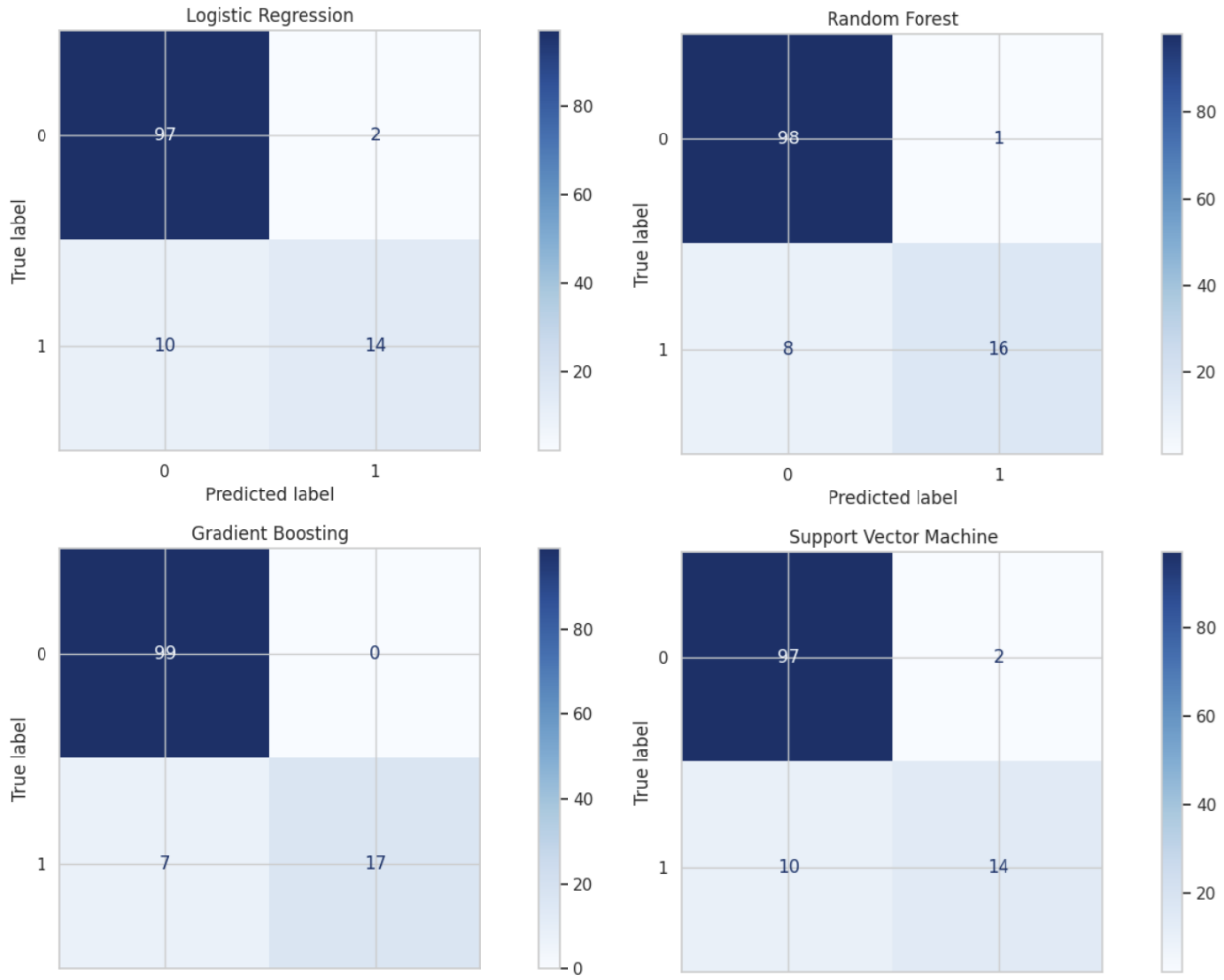

Рисунок 3.8 – Матриці плутанини

Модель логістичної регресії добре справляється з розпізнаванням класу 0, однак має певну кількість помилок при класифікації класу 1. Метод опорних векторів демонструє аналогічну поведінку: високу точність для одного класу й більшу кількість помилок для іншого. Це свідчить про їх обмежену здатність до розпізнавання менш виражених або складніших для класифікації зразків класу 1.

Натомість модель випадкового лісу показала значно вищу точність класифікації. Вона демонструє зменшену кількість помилок як для класу 0, так і для класу 1, що вказує на хорошу збалансованість моделі при класифікації

об’єктів обох категорій. Градієнтний бустинг виявився ще точнішим у визначенні класу 0, не зробивши жодної помилки, водночас допускаючи мінімальні помилки при класифікації класу 1.

Загалом, найкращі результати були досягнуті за допомогою моделей градієнтного бустингу та випадкового лісу. Вони забезпечили найвищу якість класифікації та найменшу кількість хибних передбачень. Це свідчить про їхню здатність краще захоплювати складні взаємозв’язки в даних і адаптуватися до особливостей задачі. Моделі логістичної регресії та опорних векторів, хоч і є простішими, продемонстрували стабільну, але дещо нижчу ефективність, що може бути виправдано їх меншою складністю та меншим ризиком перенавчання.

Загальні результати роботи моделей показано на рисунку 3.9

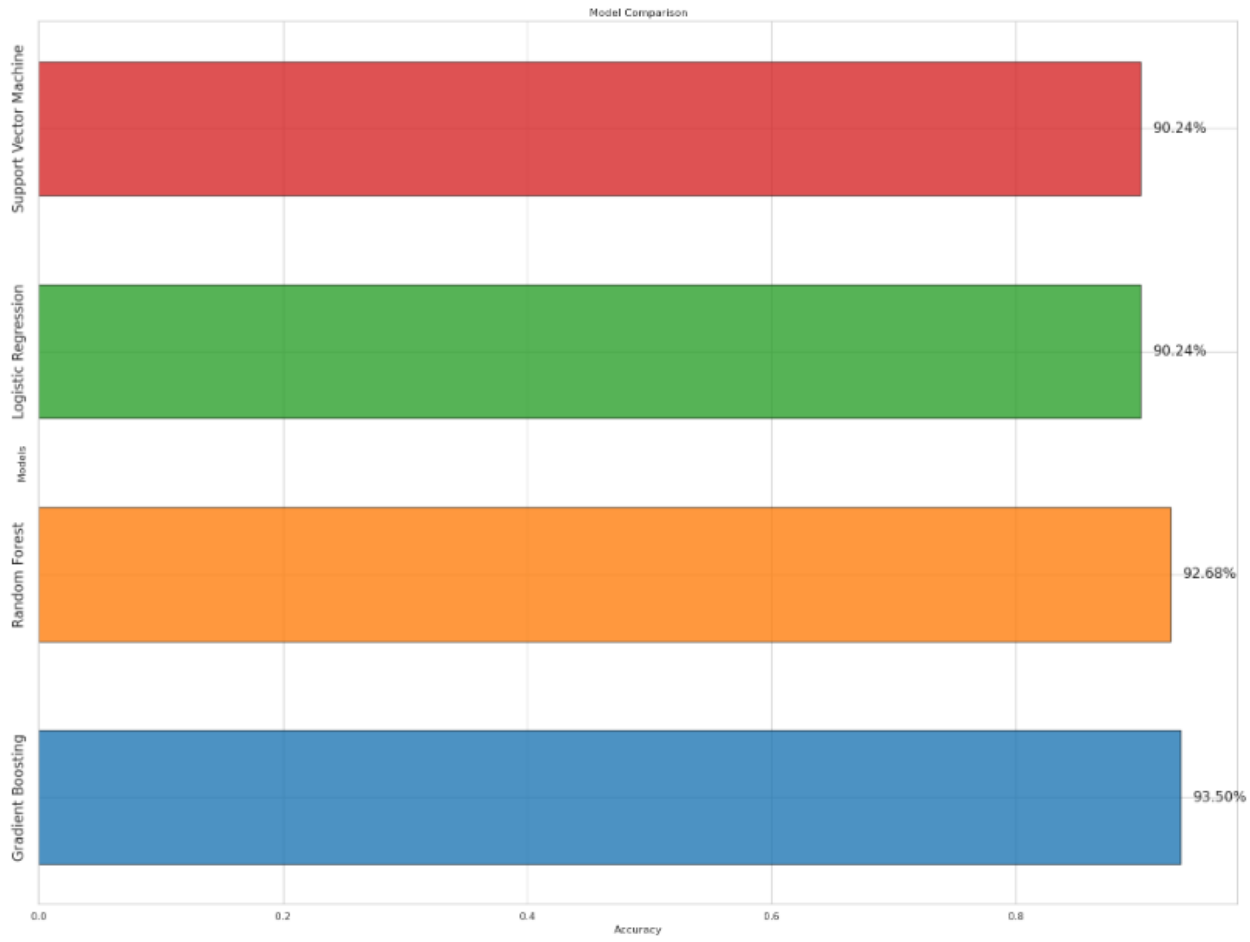

Рисунок 3.9 – Результати роботи моделей

На рисунку 3.9 наведено порівняльний аналіз точності чотирьох моделей машинного навчання, представлений у вигляді горизонтального графіка. Він чітко демонструє ефективність кожної моделі за метрикою точності. Найкращим результатом відзначилася модель Gradient Boosting, яка досягла точності 93.50%, що свідчить про її здатність найбільш ефективно розпізнавати об'єкти обох класів. Трохи нижчий, але також високий результат показала модель Random Forest з точністю 92.68%, що підтверджує її надійність і стабільність у класифікаційних завданнях.

Модель Gradient Boosting було протестовано на тестовій вибірці, що була виділена з початкового датасету під час попереднього поділу даних на навчальну та тестову частини (рис. 3.10).

```
In [29]: accuracy = gb_model.score(X_test, y_test)
         print(f"Accuracy of gb_model: {accuracy:.2f}")

         Accuracy of gb_model: 0.94
```

Рисунок 3.10 – Результат роботи моделі на тестовій вибірці

Точність 94% була отримана саме на цій тестовій вибірці, яка містить раніше невідомі моделі дані, що дозволяє об’єктивно оцінити її здатність до узагальнення та передбачення нових прикладів. Таким чином, модель продемонструвала високу ефективність у класифікації даних, які не використовувались під час навчання.

Надалі було застосовано трифазний підхід (розвідувальний аналіз, Optuna - широкий пошук, GridSearchCV - уточнення) до оптимізації моделей.

На етапі розвідувального аналізу:

- визначено оптимальні діапазони гіперпараметрів на основі розміру датасету (492 тренувальних зразків);
- виявлено дисбаланс класів (540 здорових vs 75 хворих).

На етапі широкого пошуку (Optuna) Bayesian Optimization для дослідження великого простору параметрів виявлено найважливіші гіперпараметри та їх оптимальні діапазони.

На етепі уточнення (GridSearchCV) створили звужені сітки навколо знайдених Optuna оптимумів

- ±50 для integer параметрів
- ±0.05/0.1 для float параметрів

Застосований підхід для тюнінгу моделей дозволив покращити точність передбачення для 2 найкращих моделей Gradient Boosting (Accurancy – 0,99) Random Forest (Accurancy – 0,967).

Метрика	Gradient Boosting	Random Forest
Accuracy	0.9919	0.9675
F1-score	0.9655	0.8571
Precision	1.0000	0.9231
Recall	0.9333	0.8000
ROC AUC	0.9981	0.9951
PR AUC	0.9886	0.9654

Рисунок 3.11 – Точність найкращих моделей після їх тюнінгу

Також була побудована діаграма важливості ознак (рис. 3 12).

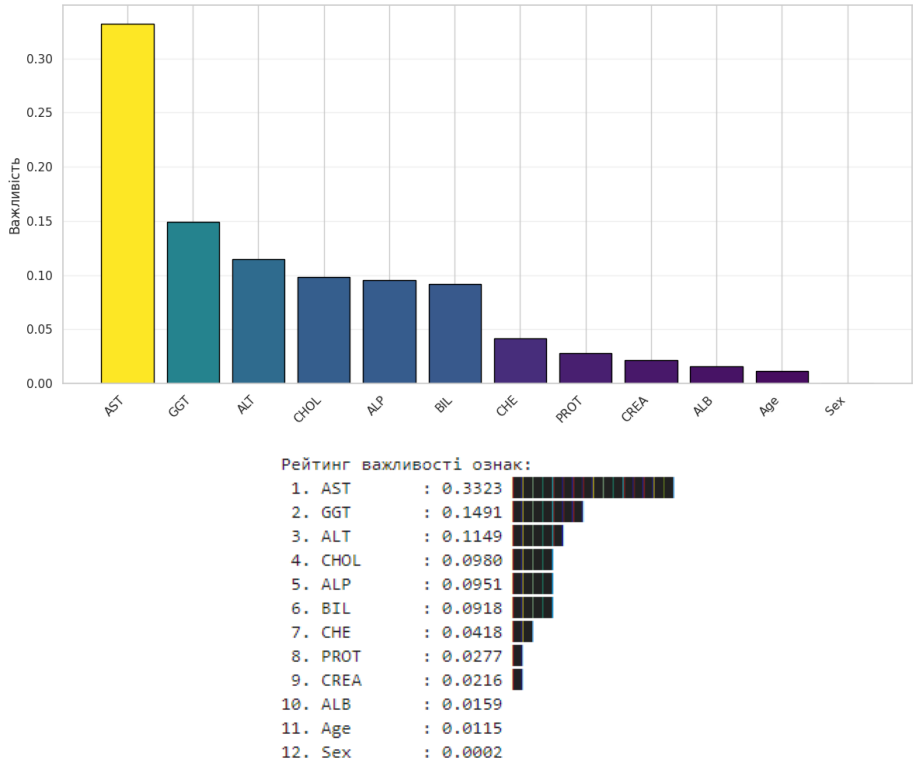

Рисунок 3.12 – Діаграма важливості ознак

Діаграма важливості ознак (рисунок 3.12) є стовпчастою діаграмою, що візуалізує відносну значущість різних вхідних ознак (змінних) для певної моделі машинного навчання, розробленого для передбачення стану захворювання печінки, враховуючи, що більшість ознак є біохімічними маркерами крові.

Діаграма чітко показує, як нерівномірно розподіляється важливість ознак. Значення по вертикальній осі (Y) представляє оцінку важливості (у цьому випадку, ймовірно, нормалізовану від 0 до 1, де сума всіх значень дорівнює 1). Горизонтальна вісь (X) позначає назви ознак.

Ппереважна більшість прогностичної сили зосереджена лише у перших трьох показниках.

1. Домінуючий показник: AST посідає перше місце з оцінкою важливості 0.33231. Це означає, що активність аспартатамінотрансферази (AST) є найбільш критичним чинником для передбачення стану хворих на гепатит у цій моделі, значно перевершуючи всі інші параметри.

2. Показники другої групи: Наступні два показники, GGT (0.1491) та ALT (0.1149), також мають високу важливість, але їхні значення сумарно лише трохи перевищують половину важливості показника AST1. Активність гамма-глутамілтрансферази (GGT) та аланінамінотрансферази (ALT) є другим та третім найважливішими біохімічними маркерами відповідно.

Сумарна важливість цих трьох показників (AST, GGT, ALT) становить приблизно 0.5973, що свідчить про те, що вони забезпечують майже 60% загальної прогностичної сили моделі.

Показники середньої важливості

Показники з 4-го по 6-те місце демонструють помірну, але схожу важливість.

4. CHOL (0.0980)1.

5. ALP (0.0951)1.

6. BIL (0.0918)1.

Важливо відзначити, що значення важливості показника BIL (білірубін) є останнім показником, який має оцінку вище 0.091.

Показники низької важливості

Після шостого місця спостерігається різке падіння рівня важливості ознак.

7. CHE (0.0418)1.

8. PROT (0.0277)1.

9. CREA (0.0216)1.

10. ALB (0.0159)1.

Ці показники (з 7-го по 10-те) мають дуже низький індивідуальний внесок у передбачення. Наприклад, показник CHE має важливість, яка приблизно у 8 разів менша, ніж показник AST.

Незначущі показники

Наприкінці рейтингу знаходяться демографічні дані та показники, що практично не впливають на передбачення стану хворих у цій моделі1.

11. Age (Вік): 0.01151.

12. Sex (Стать): 0.00021.

Важливість Sex (Стать) є мізерно малою (0.0002), що вказує на те, що цей параметр майже не використовується моделлю для розрізнення або передбачення стану пацієнтів1. Вік також демонструє дуже низьку прогностичну силу.

Аналіз діаграми важливості ознак показує, що для прогнозування стану хворих на гепатит модель майже повністю покладається на ключові печінкові ферменти (AST, GGT, ALT). Інші біохімічні показники (CHOL, ALP, BIL) мають вторинне значення, а демографічні дані (Age, Sex) та деякі інші лабораторні параметри (PROT, ALB, CREA) є найменш значущими для передбачення.

Ця структура рейтингу свідчить, що будь-яка відсутність або неточність даних щодо AST матиме найбільший негативний вплив на точність передбачення моделі.

3.3 Висновки

У результаті досліджень розроблено інформаційну технологію для класифікації медичних даних. Дані розбито на тренувальний (80%) і тестовий (20%) набори з фіксованим random_state для відтворюваності. Найвищу точність

показали градієнтний бустинг (93.50% на тренуванні, 94% на тесті) та випадковий ліс (92.68%), оптимізовані через пошук по сітці. Логістична регресія та метод опорних векторів досягли 90.24%. Матриці плутанини підтвердили кращу узагальнюючу здатність бустингу та лісу. Результати вказують на перспективність бустингу для медичної діагностики та потребу в подальшому вдосконаленні.

Надалі було застосовано трифазний підхід (розвідувальний аналіз, Optuna - широкий пошук, GridSearchCV - уточнення**)** до оптимізації моделей. Застосований підхід для тюнінгу моделей дозволив покращити точність передбачення для 2 найкращих моделей Gradient Boosting (Accurancy – 0,99) Random Forest (Accurancy – 0,967).

4 ЕКОНОМІЧНА ЧАСТИНА

Виконання науково-дослідної роботи завжди передбачає отримання певних результатів і вимагає відповідних витрат. Результати виконаної роботи завжди дають нам нові знання, які в подальшому можуть бути використані для удосконалення та/або розробки (побудови) нових, більш продуктивних зразків техніки, процесів та програмного забезпечення.

Дослідження на тему «Інформаційна технологія аналізу та передбачення стану хворих на гепатит» може бути віднесено до пошукових наукових досліджень і спрямоване на вирішення наукових проблем, пов’язаних з практичним застосуванням. Основою таких досліджень є науковий ефект, який виражається в отриманні наукових результатів, які збільшують обсяг знань про природу, техніку та суспільство, які розвивають теоретичну базу в тому чи іншому науковому напрямку, що дозволяє виявити нові закономірності, які можуть використовуватися на практиці.

4.1 Оцінювання наукового ефекту

Основними ознаками наукового ефекту науково-дослідної роботи є новизна роботи, рівень її теоретичного опрацювання, перспективність, рівень розповсюдження результатів, можливість реалізації. Науковий ефект НДР на тему «Інформаційна технологія аналізу та передбачення стану хворих на гепатит» можна охарактеризувати двома показниками: ступенем наукової новизни та рівнем теоретичного опрацювання.

Значення показників ступеня новизни і рівня теоретичного опрацювання науково-дослідної роботи в балах наведені в таблицях 4.1 та 4.2.

Таблиця 4.1 – Показники ступеня новизни науково-дослідної роботи виставлені експертами

Ступінь новизни	Характеристика ступеня новизни	Значення ступеня новизни, бали		
		Експерти (ПІБ, посада)		
		1	2	3
Принципово нова	Робота якісно нова за постановкою задачі і ґрунтується на застосуванні оригінальних методів дослідження. Результати дослідження відкривають новий напрям в даній галузі науки і техніки. Отримані принципово нові факти, закономірності; розроблена нова теорія. Створено принципово новий пристрій, спосіб, метод	0	0	0
Нова	Отримана нова інформація, яка суттєво зменшує невизначеність наявних значень (по-новому або вперше пояснені відомі факти, закономірності, впроваджені нові поняття, розкрита структура змісту). Проведено суттєве вдосконалення, доповнення і уточнення раніше досягнутих результатів	58	60	55
Відносно нова	Робота має елементи новизни в постановці задачі і методах дослідження. Результати дослідження систематизують і узагальнюють наявну інформацію, визначають шляхи подальших досліджень; вперше знайдено зв’язок (або знайдено новий зв’язок) між явищами. В принципі відомі положення розповсюджені на велику кількість об’єктів, в результаті чого знайдено ефективне рішення. Розроблені більш прості способи для досягнення відомих результатів. Проведена часткова раціональна модифікація (з ознаками новизни)	0	0	0
Традиційна	Робота виконана за традиційною методикою. Результати дослідження мають інформаційний характер. Підтверджені або поставлені під сумнів відомі факти та твердження, які потребують перевірки. Знайдено новий варіант рішення, який не дає суттєвих переваг в порівнянні з існуючим	0	0	0
Не нова	Отримано результат, який раніше зафіксований в інформаційному полі, та не був відомий авторам	0	0	0
Середнє значення балів експертів		57,7		

Згідно отриманого середнього значення балів експертів ступінь новизни характеризується як нова, тобто отримана нова інформація, яка суттєво зменшує невизначеність наявних знань (по-новому або вперше пояснені відомі факти, закономірності, впроваджені нові поняття, розкрита структура змісту) та

проведено суттєве вдосконалення, доповнення і уточнення раніше досягнутих результатів.

Таблиця 4.2 – Показники рівня теоретичного опрацювання науково-дослідної роботи виставлені експертами

Характеристика рівня теоретичного опрацювання	Значення показника рівня теоретичного опрацювання, бали		
	Експерт (ПІБ, посада)		
	1	2	3
Відкриття закону, розробка теорії	0	0	0
Глибоке опрацювання проблеми: багатоаспектний аналіз зв'язків, взаємозалежності між фактами з наявністю пояснень, наукової систематизації з побудовою евристичної моделі або комплексного прогнозу	69	80	67
Розробка способу (алгоритму, програми), пристрою, отримання нової речовини	0	0	0
Елементарний аналіз зв'язків між фактами та наявною гіпотезою, класифікація, практичні рекомендації для окремого випадку тощо	0	0	0
Опис окремих елементарних фактів, викладення досвіду, результатів спостережень, вимірювань тощо	0	0	0
Середнє значення балів експертів	72,0		

Згідно отриманого середнього значення балів експертів рівень теоретичного опрацювання науково-дослідної роботи характеризується як глибоке опрацювання проблеми: багатоаспектний аналіз зв'язків, взаємозалежності між фактами з наявністю пояснень, наукової систематизації з побудовою евристичної моделі або комплексного прогнозу.

Показник, який характеризує рівень наукового ефекту, визначаємо за формулою [19]:

$$E_{нау} = 0,6 \cdot k_{нов} + 0,4 \cdot k_{теор}, \quad (4.1)$$

де $k_{нов}, k_{теор}$ - показники ступеня новизни та рівня теоретичного опрацювання науково-дослідної роботи, $k_{нов} = 57,7, k_{теор} = 72,0$ балів;

0,6 та 0,4 – питома вага (значимість) показників ступеня новизни та рівня теоретичного опрацювання науково-дослідної роботи.

$$E_{нау} = 0,6 \cdot k_{нов} + 0,4 \cdot k_{теор}$$=0,6·57,7 + 0,4·72,00 = 63,40 балів.

Визначення характеристики показника $E_{нау}$ проводиться на основі висновків експертів виходячи з граничних значень, які наведені в таблиці 4.3.

Таблиця 4.3 – Граничні значення показника наукового ефекту

Досягнутий рівень показника	Кількість балів
Високий	70…100
Середній	50…69
Достатній	15…49
Низький (помилкові дослідження)	1…14

Відповідно до визначеного рівня наукового ефекту проведеної науково-дослідної роботи на тему «Інформаційна технологія аналізу та передбачення стану хворих на гепатит», даний рівень становить 63,40 балів і відповідає статусу - середній рівень. Тобто у даному випадку можна вести мову про потенційну фактичну ефективність науково-дослідної роботи.

4.2 Розрахунок витрат на здійснення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Інформаційна технологія аналізу та передбачення стану хворих на гепатит», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та

іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [19-20]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.2)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=21 дні.

$$З_о = 24300{,}00 \cdot 21 / 21 = 24300{,}00 \text{ (грн)}.$$

Проведені розрахунки зведемо до таблиці 4.4.

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник науково-дослідної роботи з дослідження інформаційних технологій	24300,00	1157,14	21	24300,00
Науковий співробітник	24100,00	1147,62	7	8033,33
Інженер-аналітик (системний аналіз)	23100,00	1100,00	21	23100,00
Консультант (лікар-терапевт вищої категорії)	22000,00	1047,62	7	7333,33
Лаборант	8800,00	419,05	14	5866,67
Всього				68633,33

Основна заробітна плата робітників.

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Інформаційна технологія аналізу та передбачення стану хворих на гепатит» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \tag{4.3}$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \tag{4.4}$$

де $М_М$ – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=8000,00 грн;

$К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (таблиця 4.5) [19];

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$Т_р$ – середнє число робочих днів в місяці, приблизно $Т_р$ = 21 дн;

$t_{зм}$ – тривалість зміни, год.

$С_1$ = 8000,00 · 1,10 · 1,15 / (21 · 8) = 60,24 (грн).

$З_{р1}$ = 60,24 · 8,00 = 481,90 (грн).

Таблиця 4.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Підготовка робочого місця інженера-аналітика	8,00	2	1,10	60,24	481,90
Інсталяція програмного забезпечення моделювання смптомів захворювання	5,52	3	1,35	73,93	408,09
Введення програмних блоків моделювання процедурних шумів	18,10	4	1,50	82,14	1486,79
Налагодження програмних блоків математичної моделі	6,45	5	1,70	93,10	600,46
Формування (введення) бази даних дослідження моделі станів хворих на гепатит	32,00	3	1,35	73,93	2365,71
Тестування взаємодії досліджуваних моделей	9,50	5	1,70	93,10	884,40
Контроль циклічного експерименту	22,00	3	1,35	73,93	1626,43
Технічна підтримка експерименту	11,00	3	1,35	73,93	813,21
Всього					8667,00

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.5)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 10%.

$З_{дод}$ = (68633,33 + 8667,00) · 10 / 100% = 7730,03 (грн).

4.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.6)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (68633,33 + 8667,00 + 7730,03) · 22 / 100% = 18706,68 (грн).

4.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Інформаційна технологія аналізу та передбачення стану хворих на гепатит».

Витрати на матеріали на даному етапі проведення досліджень в основному пов’язані з використанням моделей елементів та моделювання роботи і досліджень за допомогою комп’ютерної техніки та створення експериментальних математичних моделей або програмного забезпечення, тому дані витрати формуються на основі витратних матеріалів характерних для офісних робіт.

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{в\,j}, \quad (4.7)$$

де $Н_j$ – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*);

B_j – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

M_1 = 2,000 · 174,00 · 1,05 - 0 · 0 = 365,40 (грн).

Проведені розрахунки зведемо до таблиці 4.6.

Таблиця 4.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 од, грн	Норма витрат, од.	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Багатофункціональний білий офісний папір OFFICE-500 A4	174,00	2	0	0	365,40
Папір для записів OFFICE 70 A5-250	117,00	3	0	0	368,55
Органайзер офісний OFFICE 100	236,00	4	0	0	991,20
Набір офісний DATUM 300	214,00	3	0	0	674,10
Картридж для принтера HP-5500	1256,00	1	0	0	1318,80
Інші матеріали	250,00	1	0	0	262,50
Flesh-пам'ять 64 GB	201,00	2	0	0	422,10
Тека для паперів	102,00	1	0	0	107,10
Всього					4509,75

4.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Інформаційна технологія аналізу та передбачення стану хворих на гепатит», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (4.8)$$

де $Н_j$ – кількість комплектуючих j-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих *j*-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*).

$К_в = 1 \cdot 599{,}00 \cdot 1{,}07 = 640{,}93$ (грн).

Проведені розрахунки зведемо до таблиці 4.7.

Таблиця 4.7 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Концентратор Defender SEPTIMA SLIM (83505)	1	599,00	640,93
Кабель для передачі даних USB to COM 1.0m Patron (CAB-PN-USB-COM)	1	499,00	533,93
Жорсткий диск 2.5" 2TB Seagate (STGD2000200)	1	3499,00	3743,93
Всього			4918,79

4.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (4.9)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

$К_i$ – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($К_i$ = *1,10…1,12);*

k – кількість найменувань устаткування.

$B_{спец}$ = 3699,00 · 1 · 1,08 = 3994,92 (грн).

Отримані результати зведемо до таблиці 4.8:

Таблиця 4.8 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Маршрутизатор TP-LINK Archer AX55	1	3699,00	3994,92
Моноблок HP 205 G8 Starry White (6D455EA)	1	41599,00	44926,92
Серверне обладнання обробки (Комп'ютер Artline Gaming X66 v22 (X66v22) AMD Ryzen 5 5600X / RAM 16ГБ / HDD 2ТБ + SSD 480ГБ / nVidia GeForce RTX 3060 Ti 8ГБ)	1	38499,00	41578,92
Всього			90500,76

4.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.10)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.і}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

$K_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_і$ = *1,10…1,12);*

k – кількість найменувань програмних засобів.

$B_{прг}$ = 7410,00 · 1 · 1,1 = 8151,00 (грн).

Отримані результати зведемо до таблиці 4.9:

Таблиця 4.9 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Прикладне програмне забезпечення розробки системи аналізу	1	7410,00	8151,00
Платформа Kaggle	1	4299,00	4728,90
Доступ до мережі Internet (високошвидкісний) грн/місяць	1	350,00	385,00
Hepatitis C Prediction Dataset	1	2599,00	2858,90
Всього			16123,80

4.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (4.11)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл}$ = (26950,00 · 1) / (4 · 12) = 561,46 (грн).

Проведені розрахунки зведемо до таблиці 4.10.

Таблиця 4.10 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Мережеве сховище Synology 4BAY DS923+	26950,00	4	1	561,46
Дослідницька лабораторія	430000,00	30	1	1194,44
Місце аналітика спеціалізоване	8150,00	5	1	135,83
Офісна оргтехніка Samsung TM	10200,00	4	1	212,50
Пристрої виведення інформації	6699,00	5	1	111,65
Програмне забезпечення Microsoft Windows, Offise 2021	9280,00	2	1	386,67
Програмно-обчислювальний комплекс розробки системи аналізу даних ARTLINE X57WHITE v41) (X57WHITEv41) Intel Core i5-12400F / RAM 16ГБ / HDD 2ТБ	39999,00	2	1	1666,63

+ SSD 480ГБ / nVidia GeForce RTX 3060 Ti 8 ГБ				
Всього				4269,18

4.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (4.12)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 12,56 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

B_e = 0,08 · 160,0 · 12,56 · 0,95 / 0,97 = 160,77 (грн).

Проведені розрахунки зведемо до таблиці 4.11.

Таблиця 4.11 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Мережеве сховище Synology 4BAY DS923+	0,08	160,0	160,77
Офісна оргтехніка Samsung TM	0,52	2,2	14,37
Пристрої виведення інформації	0,30	3,2	12,06
Програмно-обчислювальний комплекс розробки системи аналізу даних ARTLINE Gaming X57WHITE v41) (X57WHITEv41) Intel Core i5-12400F / RAM 16ГБ / HDD 2ТБ + SSD 480ГБ / nVidia GeForce RTX 3060 Ti 8 ГБ	0,36	160,0	723,46

Маршрутизатор TP-LINK Archer AX55	0,02	160,0	40,19
Моноблок HP 205 G8 Starry White (6D455EA)	0,20	160,0	401,92
Серверне обладнання обробки (Комп'ютер Artline Gaming X66 v22 (X66v22) AMD Ryzen 5 5600X / RAM 16ГБ / HDD 2ТБ + SSD 480ГБ / nVidia GeForce RTX 3060 Ti 8ГБ)	0,40	160,0	803,84
Всього			2156,60

4.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Інформаційна технологія аналізу та передбачення стану хворих на гепатит» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{H_{св}}{100\%}, \quad (4.13)$$

де $H_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св} = 21\%$.

$B_{св}$ = (68633,33 + 8667,00) · 21 / 100% = 16233,07 (грн).

4.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (4.14)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 35%.

$B_{сп}$ = (68633,33 + 8667,00) · 35 / 100% = 27055,12 (грн).

4.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.15)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 60%.

$І_в$ = (68633,33 + 8667,00) · 60 / 100% = 46380,20 (грн).

4.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків;

витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.16)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 107%.

$B_{нзв}$ = (68633,33 + 8667,00) · 107 / 100% = 82711,36 (грн).

Витрати на проведення науково-дослідної роботи на тему «Інформаційна технологія аналізу та передбачення стану хворих на гепатит» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + I_в + B_{нзв}. \quad (4.17)$$

$B_{заг}$= 68633,33 +8667,00 +7730,03 +18706,68 +4509,75 +4918,79 +90500,76 + 16123,80 + 4269,18 +2156,60 +16233,07 +27055,12 +46380,20 +82711,36 = 398595,68 (грн).

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (4.18)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

ЗВ = 398595,68 / 0,9 = 442884,09 (грн).

4.3 Оцінювання важливості та наукової значимості науково-дослідної роботи

Оцінювання та доведення ефективності виконання науково-дослідної роботи фундаментального чи пошукового характеру є достатньо складним процесом і часто базується на експертних оцінках, тому має вірогідний характер.

Для обґрунтування доцільності виконання науково-дослідної роботи на тему «Інформаційна технологія аналізу та передбачення стану хворих на гепатит» використовується спеціальний комплексний показник, що враховує важливість, результативність роботи, можливість впровадження її результатів у виробництво, величину витрат на роботу.

Комплексний показник K_P рівня науково-дослідної роботи може бути розрахований за формулою:

$$K_P = \frac{I^n \cdot T_C \cdot R}{B \cdot t}, \tag{4.19}$$

де I – коефіцієнт важливості роботи. Приймемо $I = 4$;

n – коефіцієнт використання результатів роботи; $n = 0$, коли результати роботи не будуть використовуватись; $n = 1$, коли результати роботи будуть використовуватись частково; $n = 2$, коли результати роботи будуть використовуватись в дослідно-конструкторських розробках; $n = 3$, коли результати можуть використовуватись навіть без проведення дослідно-конструкторських розробок. Приймемо n=3;

T_C – коефіцієнт складності роботи. Приймемо $T_C = 2$;

R – коефіцієнт результативності роботи; якщо результати роботи плануються вище відомих, то $R = 4$; якщо результати роботи відповідають відомому рівню, то $R = 3$; якщо нижче відомих результатів, то $R = 1$. Приймемо $R = 3$;

B – вартість науково-дослідної роботи, тис. грн. Приймемо B =442884,09 грн;

t – час проведення дослідження. Приймемо t = 0,08 років, (1 міс.).

Визначення показників *I, n, T_C, R, B, t* здійснюється експертним шляхом або на основі нормативів [Козловський, Лесько, Кавецький].

$$K_P = \frac{I^n \cdot T_C \cdot R}{B \cdot t} = 4^3 \cdot 2 \cdot 3 \,/\, 442{,}9 \cdot 0{,}08 = 10{,}40.$$

Якщо $K_P > 1$, то науково-дослідну роботу на тему «Інформаційна технологія аналізу та передбачення стану хворих на гепатит» можна вважати ефективною з високим науковим, технічним і економічним рівнем.

4.4 Висновки

Витрати на проведення науково-дослідної роботи на тему «Інформаційна технологія аналізу та передбачення стану хворих на гепатит» складають 442884,09 грн. Відповідно до проведеного аналізу та розрахунків рівень наукового ефекту проведеної науково-дослідної роботи на тему «Інформаційна технологія аналізу та передбачення стану хворих на гепатит» є середній, а дослідження актуальними, рівень доцільності виконання науково-дослідної роботи $K_P > 1$, що свідчить про потенційну ефективність з високим науковим, технічним і економічним рівнем.

ВИСНОВКИ

Проведено аналіз предметної області, обґрунтовано актуальність дослідження та реалізовано інформаційну технологію для передбачення медичних даних. У першому розділі розглянуто об’єкт дослідження, визначено його актуальність та проаналізовано методи моделювання, зокрема Random Forest, Support Vector Machine, Logistic Regression і Gradient Boosting. Здійснено вибір сучасних інструментів, таких як Python і платформа Kaggle, який забезпечив ефективне розв’язання задачі.

Здійснено підготовку середовища Kaggle, імпорт бібліотек, попередню обробку датасету та розвідувальний аналіз. Аналіз сирих даних виявив закономірності, зокрема значний вплив параметра AST на виникнення гепатиту, що дозволило ідентифікувати ключові фактори ризику та закласти основу для моделювання.

Побудовано моделі передбачення з використанням асамблевого підходу, який підвищив точність до 0.95. Найкращий результат продемонстрував Random Forest, що домінував у голосуванні асамблевого моделювання. Gradient Boosting також показав високу ефективність (93.50% на тренуванні та 94% на тесті), підтвердивши надійність ансамблевих методів.

Надалі було застосовано трифазний підхід (розвідувальний аналіз, Optuna - широкий пошук, GridSearchCV - уточнення**)** до оптимізації моделей. Застосований підхід для тюнінгу моделей дозволив покращити точність передбачення для 2 найкращих моделей Gradient Boosting (Accurancy – 0,99) Random Forest (Accurancy – 0,967). Отримані результати свідчать про успішне виконання поставлених цілей, а також відкривають перспективи для практичного застосування моделей у медичній діагностиці з можливістю подальшого вдосконалення.

Відповідно до проведеного аналізу та розрахунків рівень наукового ефекту проведеної науково-дослідної роботи на тему «Інформаційна технологія аналізу та передбачення стану хворих на гепатит» є середній, а дослідження актуальними, рівень доцільності виконання науково-дослідної роботи $K_P > 1$, що

свідчить про потенційну ефективність з високим науковим, технічним і економічним рівнем.

За результатами даного дослідження подано тези на конференцію «LV Всеукраїнську науково-технічну конференцію підрозділів Вінницького національного технічного університету (ВНТКП ВНТУ).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Поліщук Д.М., Крижановський Є.М., Штельмах І.М., «Інформаційна технологія аналізу та передбачення стану хворих на гепатит», «LV Всеукраїнська науково-технічна конференція підрозділів Вінницького національного технічного університету (ВНТКП ВНТУ)» (Вінниця, 2025-2026 рр.). URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2026/paper/view/25993/21471 (дата звернення: 15.11.2025).

2. Hepatitis C - Symptoms and causes - Mayo Clinic URL:URL: https://www.mayoclinic.org/diseases-conditions/hepatitis-c/symptoms-causes/syc-20354278 (дата звернення: 13.10.2025).

3. Hepatitis C - FAQs, Statistics, Data, & Guidelines | CDC: URL: https://www.cdc.gov/hepatitis/hcv/index.htm (дата звернення: 15.11.2025)

4. Гепатит С, гострий - MSD Manual Professional Edition: URL: https://www.msdmanuals.com/uk/professional/hepatic-and-biliary-disorders/hepatitis/hepatitis-c-acute (дата звернення: 10.11.2025).

5. Machine Learning - Машинне навчання: URL: https://www.it.ua/knowledge-base/technology-innovation/machine-learning (дата звернення: 12.10.2025)

6. Verhun V. R. Характеристика методів розв'язання задачі класифікації в інтелектуальному аналізі даних навчальних програм. Scientific Bulletin of UNFU. 2019. T. 29, № 6. C. 136–139.

7. Задачі Data Mining та їх класифікація. Інформація та знання. URL: http://surl.li/celjv (дата звернення: 11.11.2025)

8. Wikipedia. URL: https://uk.wikipedia.org/ (дата звернення: 15.11.2025)

9. Programming: Pick up Python. Vol. 518, no. 7537. C. 125–126. Nature. 2015.

10. Kaggle. URL: https://www.kaggle.com/ (дата звернення: 5.12.2025)

11. Бібліотека NumPy. 2025. URL: https://devzone.org.ua/post/chomu-vamslid-vykorystovuvaty-numpy. (Дата звернення: 10.10.2025).

12. Python. 2025. URL: https://hyperhost.ua/info/uk/shho-take-python-inavishho-vin-potriben. (Дата звернення: 10.10.2025).

13. Hu, J., Szymczak, S. A review on longitudinal data analysis with random forest. Briefings in Bioinformatics. 2023, vol. 24, issue 2, article ID bbad002. DOI: 10.1093/bib/bbad002.

14. Haddouchi, M., Berrado, A. A survey and taxonomy of methods interpreting random forest models. arXiv preprint, 2024. arXiv: 2407.12759. DOI: 10.48550/arXiv.2407.12759. URL: https://arxiv.org/abs/2407.12759.

15. Sun, Z., Wang, G., Li, P., Wang, H., Zhang, M., Liang, X. An improved random forest based on the classification accuracy and correlation measurement of decision trees. Expert Systems with Applications. 2024, vol. 237, article ID 121549. DOI: 10.1016/j.eswa.2023.121549.

16. Петрина, В. В., Дорошенко, А. В. Ефективність застосування методів класифікації для задач інтелектуального аналізу великих даних. Науковий вісник НЛТУ України. 2024, т. 34, № 5, с. 119–128. DOI: 10.36930/40340516. URL: https://nv.nltu.edu.ua/index.php/journal/article/view/2639.

17. Мокін, В. Б., Дратований, М. В. Наука про дані: машинне навчання та інтелектуальний аналіз даних : електрон. навч. посіб. Вінниця : ВНТУ, 2024. 263 с. URL: https://pdf.lib.vntu.edu.ua/books/2024/Mokin_2024_263.pdf.

18. Гавриленко, С. Ю. Машинне навчання : навч.-метод. матеріали (тема «Ансамблеві методи, Random Forest»). Харків : [КПІ Харків], 2024.

19. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. 42 с.

20. Кавецький В. В., Козловський В.О., Причепа І. В. Економічне обґрунтування інноваційних рішень: практикум. Вінниця : ВНТУ, 2016. 113 с.

21. Hepatitis C - WHO | World Health Organization. URL: https://www.who.int/news-room/fact-sheets/detail/hepatitis-c (дата звернення: 15.11.2025)

Додаток А

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«___» _____ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на магістерську кваліфікаційну роботу
«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПЕРЕДБАЧЕННЯ СТАНУ ХВОРИХ НА ГЕПАТИТ»
08-34.МКР.012.02.000.ТЗ

Керівник: к.т.н., ас. каф. САІТ
_____ Ігор ШТЕЛЬМАХ
«___» _____ 2025 р.

Розробив: студент гр. 2ІСТ-24м
_____ Денис ПОЛІЩУК
«___» _____ 2025 р.

Вінниця 2025

1. Підстава для проведення робіт

Підставою для виконання роботи є наказ № __ по ВНТУ від «__» _____ 2025 р., та індивідуальне завдання на МКР, затверджене протоколом № __ засідання кафедри САІТ від «__» _____ 2025 р.

2. Джерела розробки:

- Гепатит С, гострий - MSD Manual Professional Edition [Електронний ресурс]: URL: https://www.msdmanuals.com/uk/professional/hepatic-and-biliary-disorders/hepatitis/hepatitis-c-acute
- Machine Learning - Машинне навчання [Електронний ресурс]: URL: https://www.it.ua/knowledge-base/technology-innovation/machine-learning

3. Мета і призначення роботи:

Метою роботи є підвищення точності визначення типу гепатиту у пацієнтів шляхом застосування інформаційних технологій для обробки та аналізу їх медичних даних.

4. Вихідні дані для проведення робіт:

- Набір даних біомедичних даних для передбачення стану хворих на гепатит. Kaggle Dataset «Hepatitis C Prediction Dataset» https://www.kaggle.com/datasets/fedesoriano/hepatitis-c-dataset;

5.Методи дослідження:

- методи статистичного аналізу даних;
- методи машинного навчання;
- методи кореляційного аналізу даних;
- методи тюнінгу моделей.

6. Етапи роботи і терміни їх виконання:

a) Аналіз предметної області ..... ____ – ____
b) Вибір оптимальних інформаційних технологій ..... ____ – ____
c) Вибір мови програмування та середовища розробки ..... ____ – ____
d) Розробка інформаційної технології ..... ____ – ____
e) Експериментальна перевірка роботи програми..... ____ – ____
f) Економічна частина..... ____ – ____
g) Оформлення матеріалів до захисту МКР ..... ____ – ____

7. Очікувані результати та порядок реалізації:

Розроблена та апробована на достатній кількості даних інформаційна технологія аналізу та передбачення стану хворих на гепатит.

8. Вимоги до розробленої документації

Текстова та ілюстративна частини роботи оформлені у відповідності до вимог «Методичних вказівок до виконання магістерських кваліфікаційних робіт для студентів спеціальності 126 «Інформаційні системи та технології» (освітня програма «Інформаційні технології аналізу даних та зображень»)».

9. Порядок приймання роботи
Публічний захист..... «___» _____ 2025 р.
Початок розробки ..... «___» _____ 2025 р.
Граничні терміни виконання МКР..... «___» _____ 2025 р.

Розробив студент групи 2ІСТ-24м _____ Денис ПОЛІЩУК

Додаток Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: « Інформаційна технологія аналізу та передбачення стану хворих на гепатит»

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра САІТ, ФІІТА, гр. 2ІСТ-24м

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 1.74 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне):

- ■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту
- □ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- □ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Віталій МОКІН, зав. каф. САІТ _____ (підпис)

Сергій ЖУКОВ, доц. каф. САІТ _____ (підпис)

Особа, відповідальна за перевірку _____ (підпис) Сергій ЖУКОВ

З висновком експертної комісії ознайомлений(-на)

Керівник _____ (підпис) Ігор ШТЕЛЬМАХ, к.т.н., ас. каф. САІТ

Здобувач _____ (підпис) Денис ПОЛІЩУК

Додаток В

Фрагмент лістингу програмни

```
import kagglehub
fedesoriano_hepatitis_c_dataset_path = kagglehub.dataset_download('fedesoriano/hepatitis-c-dataset')
print('Data source import complete.')

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import style
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OrdinalEncoder
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

style.use("fivethirtyeight")
import warnings
warnings.filterwarnings('ignore')

df=pd.read_csv('/kaggle/input/hepatitis-c-dataset/HepatitisCdata.csv')

df
```

```
print(df.isnull().sum())

df['ALB'].fillna(df['ALB'].mean(), inplace=True)
df['ALP'].fillna(df['ALP'].mean(), inplace=True)
df['CHOL'].fillna(df['CHOL'].mean(), inplace=True)
df['PROT'].fillna(df['PROT'].mean(), inplace=True)
df['ALT'].fillna(df['ALT'].mean(), inplace=True)
df = df.drop('Unnamed: 0', axis=1)

print(df.isnull().sum())


df['Category'] = df['Category'].replace({'0=Blood Donor': 0, '0s=suspect Blood Donor': 0, '1=Hepatitis': 1, '2=Fibrosis': 1, '3=Cirrhosis': 1})

df['Sex'] = df['Sex'].replace({'m': 0, 'f': 1})

df

# Value counts of categorical variables
print(df['Category'].value_counts())
print(df['Sex'].value_counts())

col=['Category', 'Age', 'Sex', 'ALB', 'ALP', 'ALT', 'AST', 'BIL', 'CHE',
     'CHOL', 'CREA', 'GGT', 'PROT']

import matplotlib.pyplot as plt

# create a list of the columns to plot
columns_to_plot = ['ALB', 'ALP', 'ALT', 'AST', 'BIL', 'CHE', 'CHOL', 'CREA', 'GGT', 'PROT']

# create a box plot for each column
plt.figure(figsize=(10,6))
plt.boxplot(df[columns_to_plot].values, labels=columns_to_plot, showfliers=True)
plt.xticks(rotation=45)
plt.show()
```

```
q_low = df[col].quantile(0.01)
q_hi  = df[col].quantile(0.99)

df_outliers = df[(df[col] < q_low) | (df[col] > q_hi)]
outlier_percentage = (df_outliers.sum() / len(df_outliers)) * 100
print(outlier_percentage)

from sklearn.preprocessing import RobustScaler

# Create a RobustScaler object
robust_scaler = RobustScaler()

# Define the columns to be scaled using RobustScaler
cols_to_scale = ['ALB', 'ALP', 'ALT', 'AST', 'BIL', 'CHE', 'CHOL', 'CREA', 'GGT', 'PROT']

# Scale the selected columns using RobustScaler
df[cols_to_scale] = robust_scaler.fit_transform(df[cols_to_scale])

q_low = df[col].quantile(0.01)
q_hi  = df[col].quantile(0.99)

df_outliers = df[(df[col] < q_low) | (df[col] > q_hi)]
outlier_percentage = (df_outliers.sum() / len(df_outliers)) * 100
print(outlier_percentage)

import matplotlib.pyplot as plt

plt.hist(df['Age'])
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()

fig, ax = plt.subplots(figsize=(8,8))
plt.pie(x=df["Sex"].value_counts(),
        colors=["blue","red"],
```

```
        labels=["Male","Female"],
        autopct="%1.2f%%",
        )
    plt.show()

    import seaborn as sns
    import matplotlib.pyplot as plt

    sns.set(style="whitegrid")
    correlation_matrix = df.corr()
    fig, ax = plt.subplots(figsize=(12, 10))
    plt.title("Correlation Matrix Heatmap", fontsize=16)
    sns.heatmap(correlation_matrix, annot=True, annot_kws={"size": 12}, cmap='coolwarm',
ax=ax)
    cbar = ax.collections[0].colorbar
    cbar.ax.tick_params(labelsize=12)
    cbar.set_label('Correlation Strength', rotation=270, fontsize=14, labelpad=15)
    plt.show()

    from sklearn.model_selection import train_test_split

    X = df.drop("Category", axis=1)
    y = df["Category"]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    X

    import warnings
    import joblib

    # define a list of models to train and their corresponding hyperparameters to tune
    models = [
       {
          "name": "Logistic Regression",
          "estimator": LogisticRegression(),
```

```
        "hyperparameters": {
            "penalty": ["l2"],
            "C": [0.01, 0.1, 1, 10],
            "max_iter": [500]
        }
    },
    {
        "name": "Random Forest",
        "estimator": RandomForestClassifier(),
        "hyperparameters": {
            "n_estimators": [100, 200, 300],
            "max_depth": [5, 10, 20, None]
        }
    },
    {
        "name": "Gradient Boosting",
        "estimator": GradientBoostingClassifier(),
        "hyperparameters": {
            "n_estimators": [100, 200, 300],
            "learning_rate": [0.01, 0.1, 1],
            "max_depth": [3, 5, 10]
        }
    },
    {
        "name": "Support Vector Machine",
        "estimator": SVC(),
        "hyperparameters": {
            "C": [0.01, 0.1, 1, 10],
            "kernel": ["linear", "rbf", "sigmoid"],
            "gamma": ["scale", "auto"]
        }
    }

]

# train and tune each model
```

```
accuracies = []
best_models = {}
for model in models:
    with warnings.catch_warnings():
        warnings.simplefilter("ignore")
        print(f"Training {model['name']}...")
        grid_search = GridSearchCV(
            estimator=model['estimator'],
            param_grid=model['hyperparameters'],
            scoring='accuracy',
            cv=5
        )
        grid_search.fit(X_train, y_train)

        # evaluate the model's performance
        best_model = grid_search.best_estimator_
        y_pred = best_model.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        conf_matrix = confusion_matrix(y_test, y_pred)

        accuracies.append((model['name'], accuracy))
        best_models[model['name']] = best_model

        print(f"Best parameters for {model['name']}: {grid_search.best_params_}")
        print(f"Accuracy for {model['name']}: {accuracy}")


# create the Logistic Regression model with the best hyperparameters
log_reg_model = LogisticRegression(
    C=10,
    max_iter=500,
    penalty='l2'
)

# create the Random Forest model with the best hyperparameters
rf_model = RandomForestClassifier(
```

```
    max_depth=10,
    n_estimators=300
)

# create the Gradient Boosting model with the best hyperparameters
gb_model = GradientBoostingClassifier(
    learning_rate=0.1,
    max_depth=3,
    n_estimators=100
)

# create the Support Vector Machine model with the best hyperparameters
svm_model = SVC(
    C=10,
    gamma='scale',
    kernel='linear'
)



# train the Logistic Regression model on the training data
log_reg_model.fit(X_train, y_train)

# make predictions on the testing data
y_pred_log_reg = log_reg_model.predict(X_test)

# train the Random Forest model on the training data
rf_model.fit(X_train, y_train)

# make predictions on the testing data
y_pred_rf = rf_model.predict(X_test)

# train the Gradient Boosting model on the training data
gb_model.fit(X_train, y_train)

# make predictions on the testing data
y_pred_gb = gb_model.predict(X_test)
```

```
# train the Support Vector Machine model on the training data
svm_model.fit(X_train, y_train)

# make predictions on the testing data
y_pred_svm = svm_model.predict(X_test)



from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# define the models and their names
models = {
    'Logistic Regression': log_reg_model,
    'Random Forest': rf_model,
    'Gradient Boosting': gb_model,
    'Support Vector Machine': svm_model
}

# create a subplot grid with 2 rows and 2 columns
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(20, 10))

# iterate over the models and plot the confusion matrix in the corresponding subplot
for i, (name, model) in enumerate(models.items()):
    row = i // 2
    col = i % 2
    disp = ConfusionMatrixDisplay.from_estimator(model, X_test, y_test, cmap='Blues',
ax=axs[row, col])
    disp.ax_.set_title(name)

plt.tight_layout()
plt.show()

import matplotlib.pyplot as plt

# Sort accuracies in descending order
```

```
accuracies.sort(key=lambda x: x[1], reverse=True)

# Extract model names and accuracies
names, values = zip(*accuracies)

# Set color scheme
colors = ['tab:blue', 'tab:orange', 'tab:green', 'tab:red', 'tab:purple']

# Create bar chart
fig, ax = plt.subplots(figsize=(20, 15))
ax.barh(range(len(names)), values, color=colors, edgecolor='black', height=0.6, alpha=0.8, capsize=5, tick_label='')

# Add labels to the bars
for i, (name, acc) in enumerate(zip(names, values)):
    ax.text(acc + 0.01, i, f"{acc:.2%}", ha='left', va='center', fontsize=16)
    ax.text(-0.01, i, name, ha='right', va='center', fontsize=16, rotation=90)

# Set axis labels and title
ax.set_xlabel('Accuracy')
ax.set_ylabel('Models')
ax.set_title('Model Comparison')


# Adjust layout and save plot
plt.subplots_adjust(left=0.3)
plt.tight_layout()
plt.savefig('model_comparison.png', dpi=300)
plt.show()


accuracy = gb_model.score(X_test, y_test)
print(f"Accuracy of gb_model: {accuracy:.2f}")

import matplotlib.pyplot as plt
```

```
# fit the model
gb_model.fit(X_train, y_train)

# get feature importances
importances = gb_model.feature_importances_

# get feature names
feature_names = X.columns

# sort feature importances in descending order
indices = np.argsort(importances)[::-1]

# plot feature importances
plt.figure(figsize=(10,5))
plt.title("Feature Importances")
plt.bar(range(len(indices)), importances[indices])
plt.xticks(range(len(indices)), feature_names[indices], rotation='vertical')
plt.show()
```

Додаток Г

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПЕРЕДБАЧЕННЯ СТАНУ ХВОРИХ НА ГЕПАТИТ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2025 р.

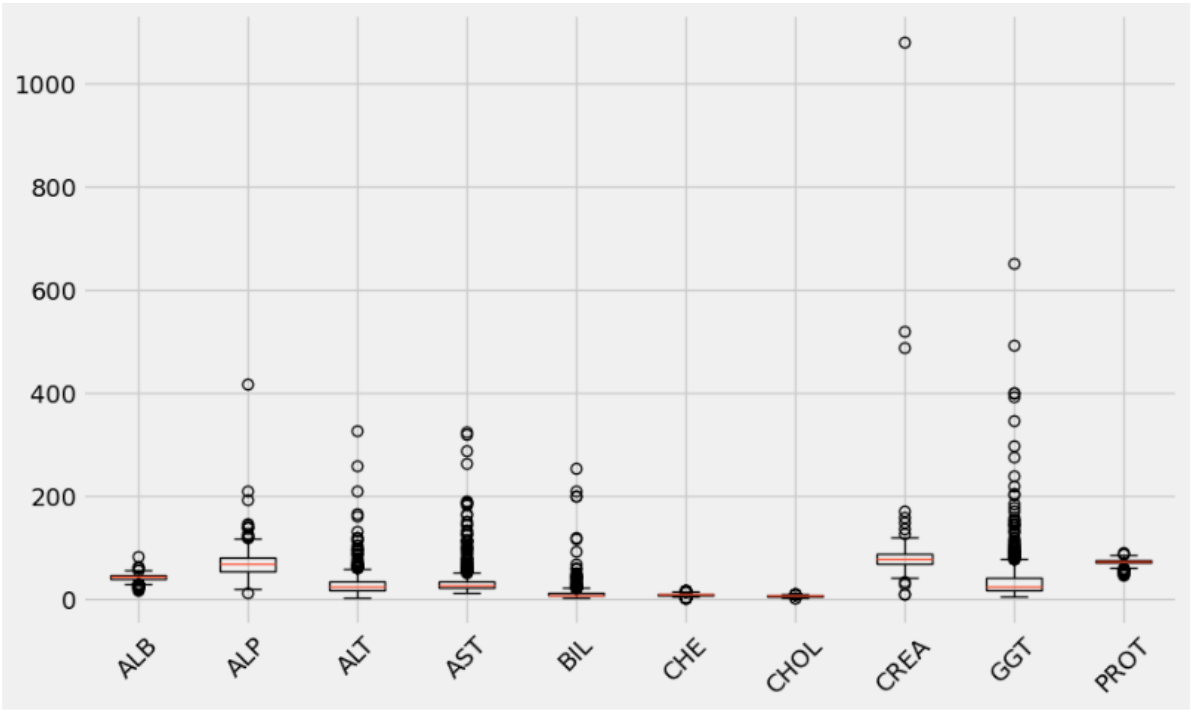

Рисунок Г.1 – Графік розподілу значень

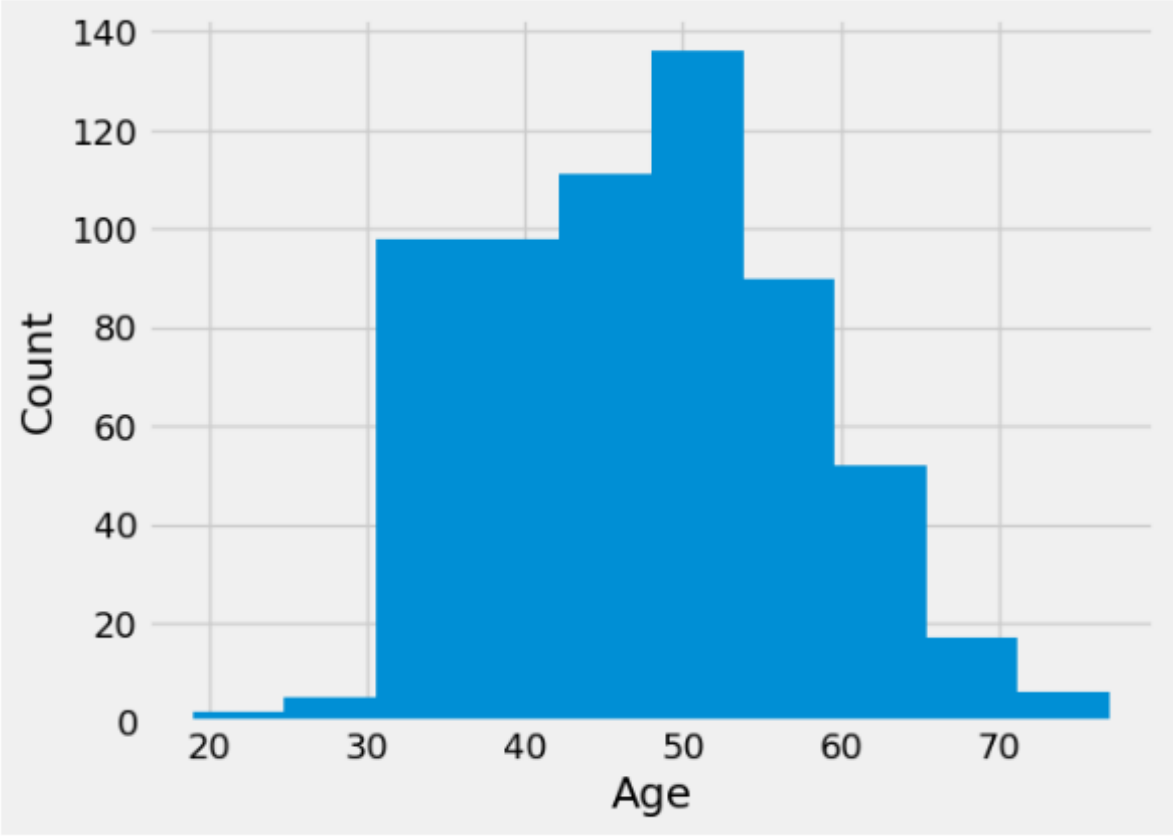

Рисунок Г.2 – Розподіл даних за віком

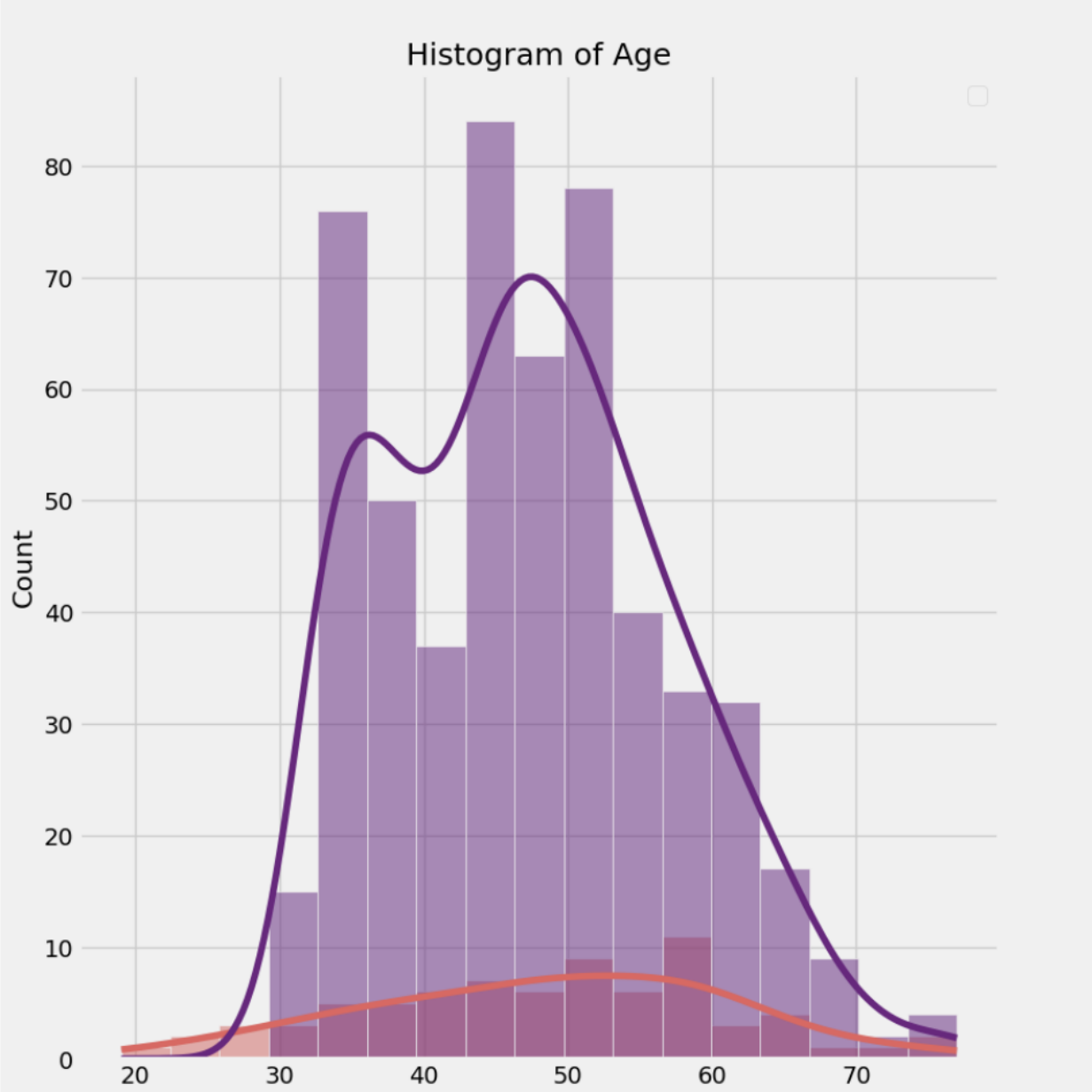

Рисунок Г.3 – Гістограма розподілу за віком

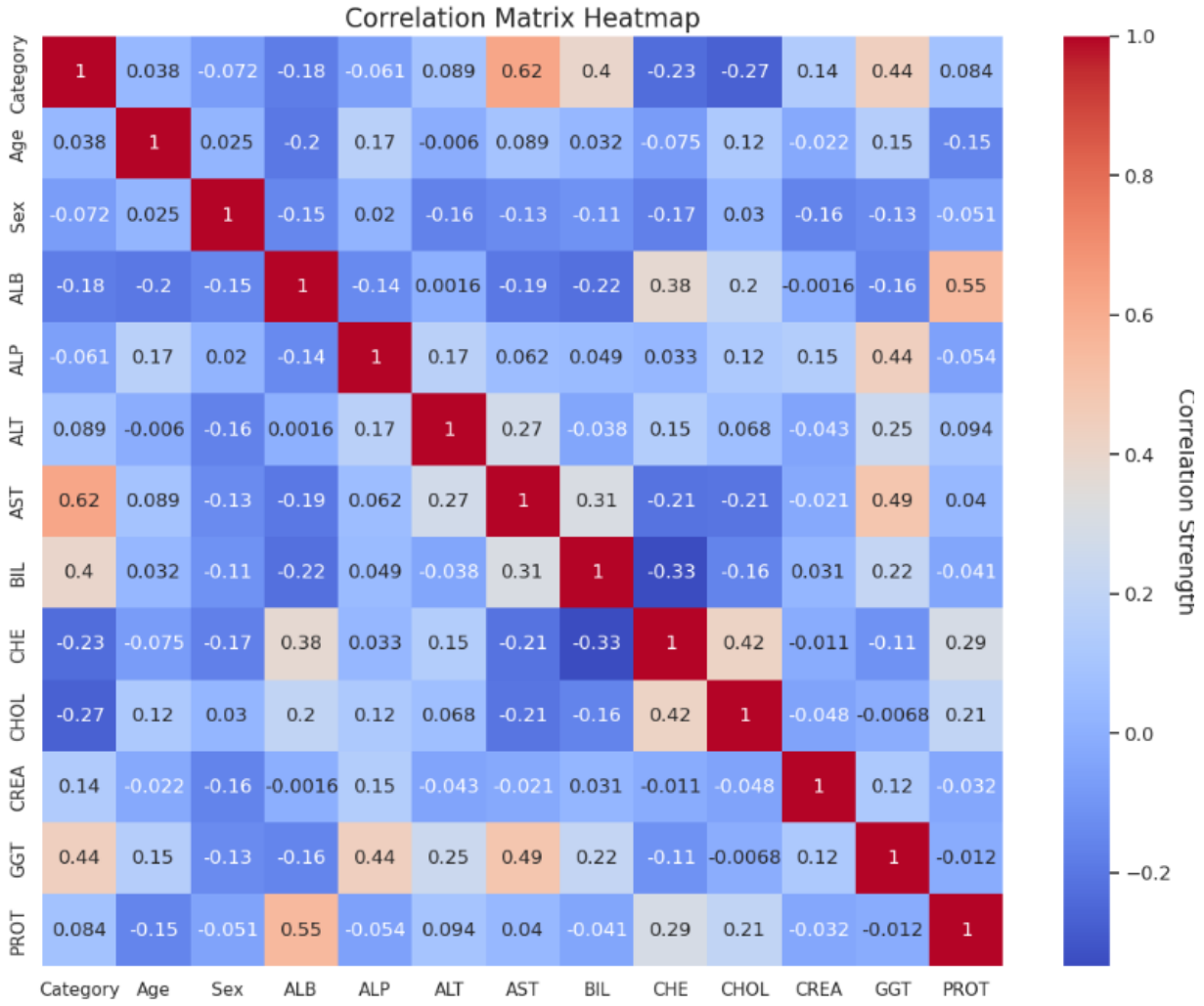

Рисунок Г.4 – Кореляційна матриця

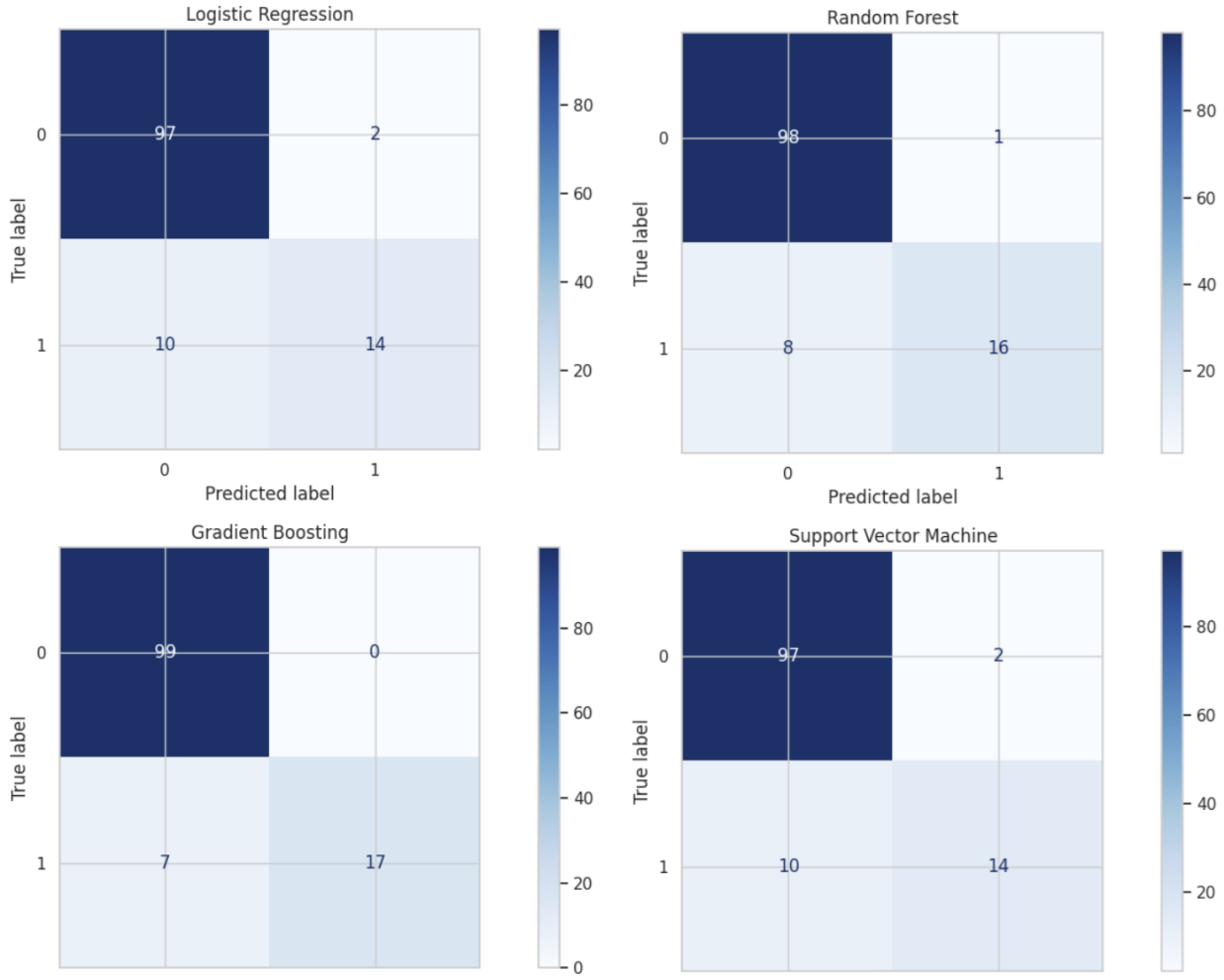

Рисунок Г.5 – Матриці плутанини

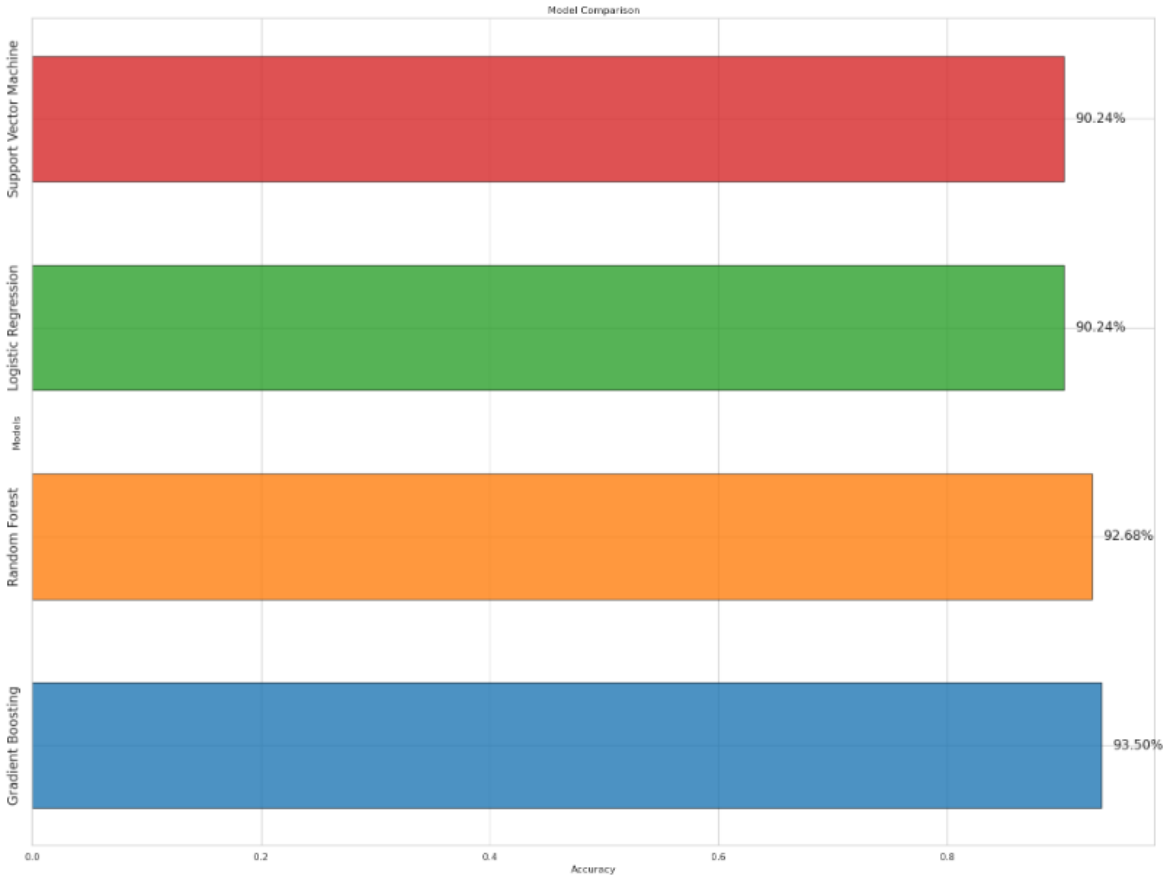

Рисунок Г.6 – Результати роботи моделей