

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ДІАГНОСТИКИ ГЛАУКОМИ RESEARCH AND SOFTWARE IMPLEMENTATION
OF GLAUCOMA DIAGNOSTIC METHODS

Виконав(ла): студент(ка) 2 курсу, групи КНТ-114 м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ПЕТУНІН О.В.

(ПРИЗВИЩЕ та ініціали)

Керівник ТАБУНЩИК Г.В.

(ПРИЗВИЩЕ та ініціали)

Рецензент ШИТІКОВА О.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ПЕТУНІНА Олександра Володимировича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів діагностики
галуками Research and Software Implementation of Glaucoma Diagnostic Methods

керівник проєкту (роботи) к.т.н., професор, ТАБУНЩИК Галина Володимирівна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 30 ” вересня 2025 року № 447

2. Строк подання студентом проєкту (роботи) 1 грудня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір та обґрунтування системи, що
проєктується. 3. Основні рішення щодо реалізації системи, яка проєктується. 4.
Керівництво програміста. 5. Керівництво оператора. 6. Тестування і результати.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-6 Основна частина	ТАБУНЩИК Г.В., к.т.н., професор		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання “ 30 ” вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи. Аналіз предметної області.	1 тиждень	
2	Розробка та затвердження технічного завдання.	2-3 тижні	Технічне завдання
3	Проектування структури системи. Розробка архітектури програми.	4-5 тижні	Структурна схема
4	Розробка схеми функціонування системи та модулів програми.	6 тиждень	Функціональна схема. Модулі
5	Створення програмного коду системи.	7-8 тижні	Текст програми
6	Тестування та відлагодження програми.	9 тиждень	Працездатна система
7	Розробка розділів пояснювальної записки.	10 тиждень	ПЗ
8	Розробка слайдів презентації. Нормоконтроль та рецензування.	11 тиждень	Слайди презентації
9	Захист роботи.	12 тиждень	

Студент(ка)

(підпис) Олександр ПЕТУНІН
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Галина ТАБУНЩИК
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
103 с., 3 табл., 28 рис., 2 дод., 35 джерел.

ГЛАУКОМА, ДІАГНОЗ, ЗОБРАЖЕННЯ ОЧНОГО ДНА, МЕТОД
НАЙБЛИЖЧИХ СУСІДІВ, НАЇВНИЙ БАЙЄС, ОСОБЛИВОСТІ, МАШИННЕ
НАВЧАННЯ, МЕТРИКИ ОЦІНЮВАННЯ, КЛАСИФІКАЦІЯ, ВИБІРКА.

Об'єкт дослідження – процес розроблення програмного забезпечення для
автоматизованого медичного діагностування глаукоми.

Предмет дослідження – алгоритми та програмні компоненти обробки і
класифікації зображень у складі програмного забезпечення для визначення
наявності глаукоми.

Мета роботи – підвищення точності комп'ютерної діагностики глаукоми
за зображеннями очного дна шляхом застосування та порівняння алгоритмів та
ансамблів алгоритмів машинного навчання.

Матеріали, методи та технічні засоби: алгоритми машинного навчання,
структурне та об'єктно-орієнтоване програмування, мова програмування Python,
персональний комп'ютер з процесором Intel Core 2 Duo під управлінням
операційної системи Microsoft Windows 10.

Результати. Створено програмний Windows-застосунок, який при наданні
зображень очного дна, тренує моделі машинного навчання та діагностує
глаукому

Висновки. У результаті виконання роботи досягнуто підвищення точності
класифікації зображень очного дна у програмному забезпеченні для
автоматизованого діагностування глаукоми.

Галузь використання – діагностування хворих в медичних закладах
лікарями-офтальмологами.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 103 pages, 3 tables, 28 figures, 2 appendix, 35 sources.

GLAUCOMA, DIAGNOSIS, FUNDUS IMAGE, NEAREST NEIGHBOR METHOD, Naïve Bayes, FEATURES, MACHINE LEARNING, EVALUATION METRICS, CLASSIFICATION, SAMPLING.

The object of the study is the process of developing software for automated medical diagnosis of glaucoma.

The subject of the study is algorithms and software components for processing and classifying images in the software for determining the presence of glaucoma.

The purpose of the work is to increase the accuracy of computer diagnostics of glaucoma using fundus images by applying and comparing algorithms and ensembles of machine learning algorithms.

Materials, methods and technical means: machine learning algorithms, structured and object-oriented programming, Python programming language, personal computer with Intel Core 2 Duo processor running Microsoft Windows 10 operating system.

Results. A Windows software application was created that, when providing fundus images, trains machine learning models and diagnoses glaucoma

Conclusions. As a result of the work, an increase in the accuracy of fundus image classification in software for automated glaucoma diagnosis was achieved.

Field of use – diagnosis of patients in medical institutions by ophthalmologists.

ЗМІСТ

	С.
<u>Перелік скорочень та умовних познач</u>	<u>8</u>
<u>Вступ</u>	<u>9</u>
<u>1 Аналіз предметної області</u>	<u>11</u>
<u>1.1 Огляд існуючих методів вирішення завдання</u>	<u>11</u>
<u>1.2 Огляд існуючого програмного забезпечення, що вирішує аналогічні завдання</u>	<u>12</u>
<u>1.2.1 Огляд програми glaucoscan.ai</u>	<u>12</u>
<u>1.2.2 Огляд програми ihealthscreen ipredict</u>	<u>13</u>
<u>1.2.3 Огляд програми altris oct</u>	<u>14</u>
<u>1.2.4 Порівняльна характеристика наявних систем</u>	<u>15</u>
<u>1.3 Постановка завдання роботи</u>	<u>15</u>
<u>1.4 Висновки за розділом</u>	<u>16</u>
<u>2 Вибір та обґрунтування структури системи, яка проектується</u>	<u>17</u>
<u>2.1 Вибір інструментарію розробки програмного забезпечення</u>	<u>17</u>
<u>2.1.1 Огляд особливостей мови програмування</u>	<u>19</u>
<u>2.1.2 Огляд особливостей обраного програмного середовища</u>	<u>22</u>
<u>2.2 Структурна схема програми</u>	<u>22</u>
<u>2.3 Опис модулів розроблюваної системи</u>	<u>23</u>
<u>2.4 Висновки за розділом 2</u>	<u>26</u>
<u>3 Основні рішення щодо реалізацій компонентів системи</u>	<u>28</u>
<u>3.1 Алгоритм функціонування розробленої програми</u>	<u>28</u>
<u>3.1.1 Метод n-найближчих сусідів</u>	<u>28</u>
<u>3.1.2 Наївний класифікатор байеса</u>	<u>30</u>
<u>3.1.3 Метод випадового лісу</u>	<u>31</u>
<u>3.1.4 Кластерний аналіз</u>	<u>32</u>
<u>3.1.5 Метод опорних векторів</u>	<u>34</u>
<u>3.1.6 Ансамблі методів машинного навчання</u>	<u>39</u>

	7
3.2 Огляд особливостей метрик інтелектуального аналізу	40
3.3 Висновки за розділом 3	42
4 Керівництво програміста	43
4.1 Призначення й умови застосування програми	43
4.2 Характеристики програми	44
4.3 Звертання до програми	44
4.4 Початкові та вихідні дані	44
4.5 Повідомлення програмісту	44
4.6 Висновок	45
5 Керівництво оператора	46
5.1 Призначення програми	46
5.2 Умови виконання програми	46
5.3 Виконання програми	46
5.4 Повідомлення оператору	49
5.5 Висновок	49
6 Тестування і результати	50
Висновки	56
Перелік джерел і посилання	58
Додаток А Текст програми	62
Додаток Б Слайди презентації	95

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;
BPN – backpropagation network;
CAD – Computer Automated Diagnostics;
CSLO – confocal scanning laser ophthalmoscopy;
DWT – Discrete wavelet transform;
HRT – Heidelberg retina tomography;
GLCM – gray-level co-occurrence matrix;
IDE – Integrated Development Environment;
KNN – K-nearest neighbours;
NB – naive Bayes;
PCA – Principal component analysis;
PCT – Principal Component Analysis;
PNN – probabilistic neural network;
ROI – region of interest;
SFFS – sequential forward floating selection;
OCT – optical coherence tomography;
VS – Visual Studio;
ВОТ – внутрішньо-очний тиск;
ПЗ – програмне забезпечення;
ПП – програмний продукт;
САПР- Системи автоматизованого прогнозування.

ВСТУП

Глаукома посідає друге місце серед найпоширеніших захворювань очей у світі за приблизно 60 мільйонів випадків, зареєстрованих у всьому світі в 2010 році, до 2040 року кількість пацієнтів, за прогнозами, зросте приблизно до 111,8 мільйонів [1]. Глаукома - захворювання очей, при якому тиск всередині ока підвищується, поле зору звужується, зоровий нерв починається до атрофії і зниження зору. Тривале підвищення внутрішньоочного тиску (ВОТ) старить нервові волокна в оці та зоровий нерв, що призводить до звуження поля зору та погіршення зору [2].

Важливим завданням у сфері біомедичної інженерії є виявлення фізіологічних змін, що відбуваються всередині людського тіла, що є важкою справою. На даний момент ці нерівності оцінюються вручну, що є дуже складним, трудомістким і виснажливим через багато складнощів, пов'язаних із методами їх виявлення. З метою виявлення захворювань на ранній стадії використання комп'ютерної діагностики набуло підвищеної уваги внаслідок потреби в системі виявлення захворювань [3]. Основна мета цієї запропонованої роботи полягає в тому, щоб побудувати систему автоматизованого проектування (CAD), яка допоможе в ранньому виявленні глаукоми, а також у скринінгу та лікуванні захворювання.

Камера очного дна – це найдоступніший спосіб аналізу зображень, який відповідає фінансовим можливостям більшості медичних закладів [4]. Вилучення особливостей структури сегментованого диска зорового нерва та сегментованої чашки зорового нерва є одним з перших методів, що використовуються для діагностики глаукоми та визначення її тяжкості [5].

Сліпота, яка виникає внаслідок глаукоми, є незворотною, але вчасне лікування можуть вберегти від неї. Тому, раннє виявлення та діагностика можуть зменшити непоправні побічні ефекти глаукоми.

Актуальність роботи полягає у збільшенні населенні, а значить і кількості людей з можливою глаукомою. При лікуванні цієї хвороби необхідно її швидке

виявлення, в чому звичайна людина може не впоратися. Тому наша розробка призначена саме для створення застосунку для швидкого автоматичного діагнозу глаукоми по цифровим зображенням очей пацієнтів.

Програма буде виконувати наступні функції:

- обробляти зображення, виводячи з них особливості;
- вирахування даних особливостей із зображень;
- запис особливостей та даних у файли;
- визначення класу діагностованих зображень;
- виведення класів оброблених зображень.

Всі необхідні дані будуть завантажені з відкритих медичних баз даних, розташованих в інтернеті.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих методів вирішення завдання

Глаукома – невиліковна хвороба, яка спричиняє погіршення зору, і є другою за частотою причиною сліпоти у світі. Щоб виявити особливості глаукоми, експерти використовують кілька методів візуалізації, включаючи конфокально скануючу лазерну офтальмоскопію (КСЛО), гейдельберзьку томографію сітківки (ГВТ), оптичну когерентну томографію (ОКТ) і візуалізацію очного дна [6]. Для діагностування хвороби аналізують наступні особливості структури сітківки: головка зорового нерва (ГЗВ) – яскрава, округла об'ємність; чашка – менша округла ділянка усередині ГЗВ; перипапільна атрофія – стоншення або відмирання нервів у вигляді півмісяця навколо диска ГЗВ та шару нервових волокон сітківки, що можна розрізнити за білою смігастою формою [7]. Для виявлення ознак глаукоми можна застосувати два види методик. Ці методи базуються на структурах, які вимагають сегментації, або на основі зображень, які потребують виділення ознак. [8]

Попередні дослідження, пов'язані з виявленням глаукоми, були зосереджені на особливостях на зображеннях очного дна, оскільки ГЗВ є основним місцем, ураженим глаукомою [9]. Перед виділенням ознак на зображенні очного дна, необхідно спочатку провести локалізацію та сегментацію ГЗВ. Локалізація виконується для формування підзображення, яке містить цікаву область у розділі ГЗВ. Потім це підзображення називається зображенням області інтересу (ЗОІ), і воно містить весь ГЗВ. Мета сегментації — відрізнити ГЗВ від фону. Деякі з найпопулярніших методів включають процеси локалізації та сегментації, такі як порогове визначення та активний контур. Були розроблені додаткові методи, такі як суперпиксель і нечіткі с-середні. Ці методи часто неправильно класифікують область ГЗВ, тому для подолання цього застосовуються морфологічні операції [10].

На основі ГЗВ розроблено кілька методів виділення ознак текстури. Методи вейвлетів і спектрів вищого порядку (СВП) є найпопулярнішими

методами виділення ознак. Аналіз основних компонентів (АОК) використовувався для зменшення ознак, отриманих за допомогою DTW, а ймовірнісна нейронна мережа (ЙНМ) використовувалася як класифікатор [11].

Щоб уникнути неприємних і незворотних наслідків глаукоми, необхідно відвідувати очних лікарів профілактично та перевірити зір. Для точної діагностики глаукоми необхідні різні тести. Через первинну глаукому протікає безсимптомно, його важко виявити на ранній стадії. Уже була описана мінливість ширини обода, поля оптичного диска і обода, запас і складність процесу ручної роботи для пошуку потоншення обода. [12].

Тому інструменти комп'ютерної діагностики (CAD) останнім часом стають більш необхідними об'єктами дослідників, які вже працюють над розробкою засобів САПР для використання офтальмологами при діабетичній ретинопатії, змінах кровоносних судин, короткозорості, ідентифікація гіпертомії. Інструменти САПР використовують цифрові зображення очного дна, зроблені за допомогою камери очного дна, і допомогу для визначення здоров'я сітківки за допомогою різних обчислювальних алгоритмів.

1.2 Огляд існуючого програмного забезпечення, що вирішує аналогічні завдання

1.2.1 Огляд програми Glaucoscan.ai

Glaucoscan.ai — це AI-рішення, яке дозволяє виявляти глаукому за допомогою фотографій сітківки, зроблених смартфоном із спеціальною лінзою, роздрукованою на 3-д принтері розробниками. Програма дає миттєвий аналіз зображення, оцінюючи ризик глаукоми, і задумана як інструмент скринінгу, особливо в низькодоступних регіонах.

Плюси:

- дозволяє проводити скринінг глаукоми за допомогою смартфона та спеціальних лінз, що можна роздрукувати на 3-д принтері;
- для використання потрібне лише доступ до інтернету;

- після завантаження фото скану штучний інтелект дає оцінку в реальному часі або дуже швидко;
- розробники опублікували відкритий код програми у своєму репозиторії гітхабу для можливої модифікації під потреби користувачів;
- згідно заяв розробників, програма має 92% точності, 91% чутливості та 94% специфічності.

Недоліки:

- для використання з телефону треба отримати або роздрукувати на 3-д принтері спеціальну лінзу;
- модель призначена для попереднього скринінгу, вирішальний діагноз потребує залучення професіонального лікаря;
- для діагностування модель потребує зображення високої якості, що відхиляє використання бюджетних камер.

1.2.2 Огляд програми iHealthScreen iPredict

iHealthScreen iPredict — набір програм для телемедичного програмування, які аналізують кольорові знімки очного дна, засновані на методах глибокого навчання. Він автоматично перевіряє якість зображень, вираховує ознаки глаукоми, сухого ока та вікової макулодистрофії, повертає звіт користувачу та за необхідністю пропонує звернутися до офтальмолога.

Плюси:

- заявлена точність досягає 94%;
- результати скринінгу можна отримати у вигляді звіту меншу ніж за 60 секунд, а повний тест проходиться за 5 хвилин;
- програму можна використовувати віддалено, маючи доступ до мережі.

Мінуси:

- призначена для скринінгу, а тому пацієнту необхідно самостійно звернутись до професіоналів при відповідній відповіді у звіті;
- потребує зображення великої якості;

– модель досягла специфічності у 83.33 відстока, а тому може пропустити деякі випадки захворювання.

1.2.3 Огляд програми Altris OCT Ai

Altris OCT in Glaucoma — програма, яка використовує машинне навчання для аналізу та класифікації глаукоми за зображеннями оптично когерентної томографії. Вона використовує шкалу ймовірності пошкодження оптичного диску для визначення захворювання.

Плюси:

- на відміну від багатьох інших розробок, використовує скани з кількох ОКТ-систем;
- призначена для роботи у складних випадках;
- призначена не тільки для виконання діагнозу глаукоми, а й багатьох інших хвороб ока;
- генерує звіти для освідомлення лікарів та пацієнтів;
- компанія-розробник дотримується вимог GDPR, тому обробка інформації відбувається відповідно до норм приватності.

Мінуси:

- ОКТ потребує наявності дорогого та складного обладнання, і їх якість суттєво вплине на результат роботи;
- обладнання для ОКТ потребує навчаних спеціалістів, що зменшує розповсюдженість та доступність цього методу;
- результати роботи потребують перевірки професійних лікарів;
- висока ціна підписки та впровадження системи у медичних закладах.

1.2.4 Порівняльна характеристики наявних систем

Ознайомившись з програмами для діагностики глаукоми, було створено їх порівняльну характеристику (табл. 1.1).

Таблиця 1.1 – Порівняльна характеристика наявних програмних засобів

Критерій порівняння	Системи автоматизованого діагностування глаукоми		
	GlaucoScan AI [13]	iHealth iPredict [14]	Altris OCT in Glaucoma [15]
Тип даних для аналізу	Знімки очного дна	Знімки очного дна	ОКТ-знімки
Призначення	Рання діагностика глаукоми за зображенням очного дна	Скринінг ризику глаукоми	Медичний аналіз структур ока за наявністю глаукоми
Точність моделі	Висока	Високу	Дуже висока
Цільова аудиторія	Лікарі-офтальмологи увіддалених місцях	Лікарі-офтальмологи	Професійні медичні заклади
Формат результатів	Шанс наявності глаукоми	Діагноз глаукоми та можлива порада звернутись до лікарів	Аналіз ОКТ
Потрібне обладнання	Телефон, спеціальна лінза	Камера для зйомки очного дна	ОСТ-сканер

Зважаючи усі за характеристики програм найкращою була обрана altris ai , оскільки вона поєднує медичну точність із практичністю використання. Система підтримує роботу як через вебзастосунок, так і через інтеграцію з клінічними PACS-системами, що дозволяє лікарям працювати незалежно від конкретного обладнання та мати доступ до результатів у будь-який момент.

1.3 Постановка завдання роботи

ПП повинен забезпечувати виконання нижче перерахованих функцій:

- введення зображень;
- діагностування глаукоми у зображених очах;
- виведення результатів у графічному форматі;
- виведення попереджень при помилках.

1.4 Висновки за розділом 1

В першому розділі було проаналізовано предметну область, її історію та вже існуючі програмні аналоги проєкту.

У ході аналізу було виявлено переваги та недоліки аналогів, функції, що повинен задовольняти проєкт.

Актуальність була доведена збільшеною кількістю пацієнтів та необхідністю їх швидко діагностувати.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЄКТУЄТЬСЯ

2.1 Вибір інструментарію розробки програмного забезпечення

Робота над проєктом почалась з вибору середовища розробки, де серед важливих компонентів є простота роботи, зручність інтерфейсу, наявність додатків. До розглянутих систем входили Visual Studio, Visual Studio Code та Eclipse.

Eclipse IDE - це інтегроване середовище розробки, що надає фундаментальний робочий простір і підтримку плагінів для задоволення потреб користувача. Основною мовою розробки, яку підтримує Eclipse є Java, а з встановленими плагінами, надає доступ до таких мов, як C, C++, C#, Clojure, Haskell, JavaScript, Perl, PHP, Python, R, Ruby, Rust. Eclipse IDE є безкоштовним і підтримується на таких операційних системах, як Windows, macOS та Linux. До головних переваг цього середовища зазвичай відносять велику кількість плагінів, швидкості та простоті у використанні, що робить добрим вибором для розробки у різних напрямках [16].

Visual Studio Code — це редактор вихідного коду, створений для роботи з такими мовами як C, C#, C++, Java, JavaScript, Python. Серед базових функцій є налагодження, кольорове підсвічування синтаксису, згортання коду, рефакторинг, доступ до Git, зіставлення дужок, спеціальні фрагменти, налаштування теми, комбінації клавіш, та завантаження розширень, для покращення функціональності і розширення списку доступних мов, підтримка Azure DevOps, можливість працювати з одним або кількома каталогами водночас, які можна встановити як робочі області для майбутнього використання. Це робить його універсальним редактором для проєктів різного напрямку [17].

Visual Studio — це набір продуктів Microsoft, який містить інтегроване середовище розробки програмного забезпечення та різні інструменти полегшення роботи. Visual Studio призначене для створення ігор, програм із

графічним інтерфейсом, консольних додатків, сайтів, служб Windows,.NET Framework, .NET Core, додатків для iOS та Android. Редактор вихідного коду має технологію IntelliSense, прості можливості рефакторингу коду, підтримку роботи налагоджувача як на рівні вихідного коду, так і на рівні машини. Серед інтегрованих інструментів для непрямої роботи з кодом є редактор форм для графічного інтерфейсу програми, редактор, дизайнер класів і дизайнер схем бази даних. Для розширення функціоналу середовища, можна встановити плагіни для додавання підтримки систем контролю версій вихідного коду, нових наборів інструментів, або інструменти орієнтовані на інші [18].

На таблиці 2.1 порівнюються розглянуті середовища розробки. Для розробки цього проєкту було обрано Visual Studio Code, через простоту роботи, підтримку розширень, швидкість компіляції.

Таблиця 2.1 – Порівняння середовищ розробки

Критерій порівняння	Середовища розробки		
	Visual studio code	Visual Studio	Eclipse
Розмір та функціональність	Важкий, багатофункціональний, призначений для серйозної розробки	Простий та швидкий	Простий, швидкий
Підтримувані мови	.NET (C#, VB.NET, F#), C++, Python, JavaScript	Js, Python, PHP, Java, C/C++/C#	Java, C/C++, PHP, Python, JavaScript тощо
Інтерфейс користувача	Комплексний і багатофункціональний	Простий та зрозумілий	Простий та зрозумілий
Вбудовані інструменти	Розширене налагодження, тестування, профілювання	Базове налагодження, рефакторинг, тестування	Розширене налагодження, тестування, профілювання
Інтеграція з системами контролю версій	Вбудовані Git, GitHub, Azure DevOps	Вбудований Git, розширюваний іншими системами	Вбудовані Git, SVN, CVS та інші системи
Спільнота та підтримка	Сильна, офіційна та спільнотна підтримка	Сильна, офіційна та спільнотна підтримка	Сильна, офіційна та спільнотна підтримка
Популярні випадки використання	Корпоративні додатки, комплексні системи, великі проєкти	Веб-розробка, скрипти, легка розробка	Java-додатки, великі корпоративні проєкти

2.1.1 Огляд особливостей мови програмування

Python — це високорівнева універсальна мова програмування відома своєю простотою, універсальністю та легкістю, призначена як для новачків, так і для професіоналів. Стандартна бібліотека модулів покриває усі потреби користувачів та доступна для використання без додаткової інсталяції. Python часто використовується для розробки, аналізу даних, штучного інтелекту, машинного навчання, наукових обчислень і автоматизації. Python має величезну спільноту розробників, і додачу є численні сторонні модулі та бібліотеки, доступні для використання. Спільнота Python дуже активна, і мова постійно розвивається новими функціями та особливостями. Серед популярних бібліотеки та фреймворки є Pandas, NumPy, SciPy, Matplotlib, Scikit-learn, TensorFlow, Keras, PyTorch, Django та Flask. Вони надають можливість виконувати завдання машинного навчання, аналізу даних розробки, веб-розробки математичних розрахунків та інших. Python є інтерпретованою мовою, що означає, що код не компілюється в машинний код перед його виконанням, оскільки інтерпретатор перетворює його у машинні інструкції під час виконання. Це спрощує процес розробки та тестування, оскільки відсутня необхідність у складних процедурах збирання проєктів. Python також є підтримує роботу на кількох операційних системах, включаючи Windows, macOS і Linux. Код, написаний на Python, можна запускати без змін на будь-якій платформі за наявності встановленого інтерпретатора. Завдяки описаним особливостям Python широко застосовується у науковій діяльності та дослідницьких проєктах і залишається однією з найпопулярніших мов програмування для програмістів різного рівня [19].

NumPy — це бібліотека, яка призначена для реалізації багатьох математичних функцій і структур у мові Python. До їх числа входять багатовимірні масиви, статистичні операції, функції лінійної алгебри, перетворення Фур'є, генерацію випадкових чисел. Важливою особливістю бібліотеки є те, що вона написана на мові C. Це робить її функції набагато швидшими ніж їх аналоги написані на Python.

Популярність цієї бібліотеки посилюється її сумісністю з багатьма іншими математичними бібліотеками, такимим як Pandas, SciPy, Matplotlib, ScyKit-learn, Openscv. Популярна бібліотека SciPy є розширенням NumPy і потребує її попереднє встановлення для своєї роботи. Вона використовує масиви NumPy проведення складних математичних розрахунків. Pandas — бібліотека з новими структурами, що сумісні з функціями NumPy. Matplotlib — бібліотека графічного виведення даних у графіках, використовується для зображення результатів роботи різних функцій і алгоритмів. SciKyt-learn призначена для реалізації алгоритмів машинного навчання. NumPy сумісний з ними і багатьма іншими, дозволяючи виконувати найрізноманітніші послідовності математичних операцій над даними для рішення великої кількості наукових проблем у дослідженнях.

OpenCV (Open Source Computer Vision Library) – це бібліотека програмного забезпечення з відкритим кодом для комп'ютерного зору та оброки зображень і відео. OpenCV була створена для забезпечення спільної інфраструктури для програм комп'ютерного зору та пришвидшення використання машинного сприйняття в комерційних продуктах.

Вона написана з можливістю для використання функцій та структур з NumPy і бібліотек написаних на ній. Зображення програмно представлені, як багатовимірні масиви пікселів, які самі є тривіірними масивами, що містять значення кольорового каналу по кожній комірці. Це дозволяє виконувати математичні операції над самими зображеннями. Це можна використати для збільшення яскравості, пошуку пікселів певних кольорів та відтінків, змішування зображень та накладання написів понад них.

Бібліотека містить багато найсучасніших оптимізованих алгоритмів комп'ютерного зору та машинного навчання. Приклади їх використання включають: вилучення 3D-моделей об'єктів, виявлення та розпізнавання облич, відстеження рухів камери, ідентифікації об'єктів, класифікації дій людини у відео, відстеження рухомих об'єктів, пошуку схожих зображень з бази даних зображень, створення 3D-хмар точок зі стереокамер, видалення ефекту червоних

очей із зображень, зроблених за допомогою спалаху, відстеження рухів очей, розпізнавання пейзажів та встановлення маркерів для накладання на них доповненої реальності тощо [20].

SKlearn — це бібліотека призначена для імплементації машинного навчання з використанням мови Python. Він надає доступ до величезної кількості готових алгоритмів машинного навчання, методів обробки даних та функцій для вирішення найпоширеніших завдань у науковій діяльності. Ці алгоритми підходять для вирішення задач кластеризації, регресії, класифікації, детекції аномалій. Ця бібліотека широко використовує інші бібліотеки, такі як, Numpy, Pandas, Matplotlib. Вона ідеально підходить якщо, немає ресурсів та часу для реалізації більш складних моделей, але є необхідність їх використання [21].

Tkinter — це кросплатформна бібліотека, призначена для створення графічних інтерфейсів додатків. Він підходить для створення простих програм із графічним інтерфейсом користувача, таких як калькулятори, аудіо-програвачі та програми оброблення зображень. Серед асортименту елементів є кнопки, слайдери, надписи, зображення, Водночас в нього відсутня підтримка деяких більш складних елементів, таких як анімація, відтворення відео або 3D-графіка.

Загалом Numpy, OpenCV, SKlearn і Tkinter є підходящими бібліотеками та фреймворками для розробки обраного проекту. Numpy допоможе у реалізації певних алгоритмів та виконанні багатьох математичних функцій, необхідних для роботи, OpenCV незамінний при роботі з зображеннями та відео, дозволить знаходити та вирізати місце інтересу із зображень очного дна та подальшого виділення ознак, TkInter полегшить завдання зі створення програмного інтерфейсу для взаємодії з моделями та виведення результатів їх роботи, а Sklearn надасть доступ до більш складних алгоритмів.

2.1.2 Огляд особливостей обраного середовища розробки

Таблиця 2.2 – Порівняння середовищ розробки

Критерій порівняння	Середовища розробки		
	Visual Studio	Visual Studio Code	Eclipse
Простота використання	++	+++	+
Швидкість компіляції	+	+++	++
Наявність візуального конструктора	++	+	+
Підтримка репозиторіїв	+++	++	+
Розширення	++	+++	++

2.2 Структурна схема програми

Розроблюваний ПП повинен обов'язково містити такі модулі:

- файли формату Json;
- файли формату Pkl;
- алгоритмів;
- ансамблів;
- графічний;
- модуль зв'язку всіх елементів ПП.

Зв'язок між модулями можна побачити на рис. 2.1.

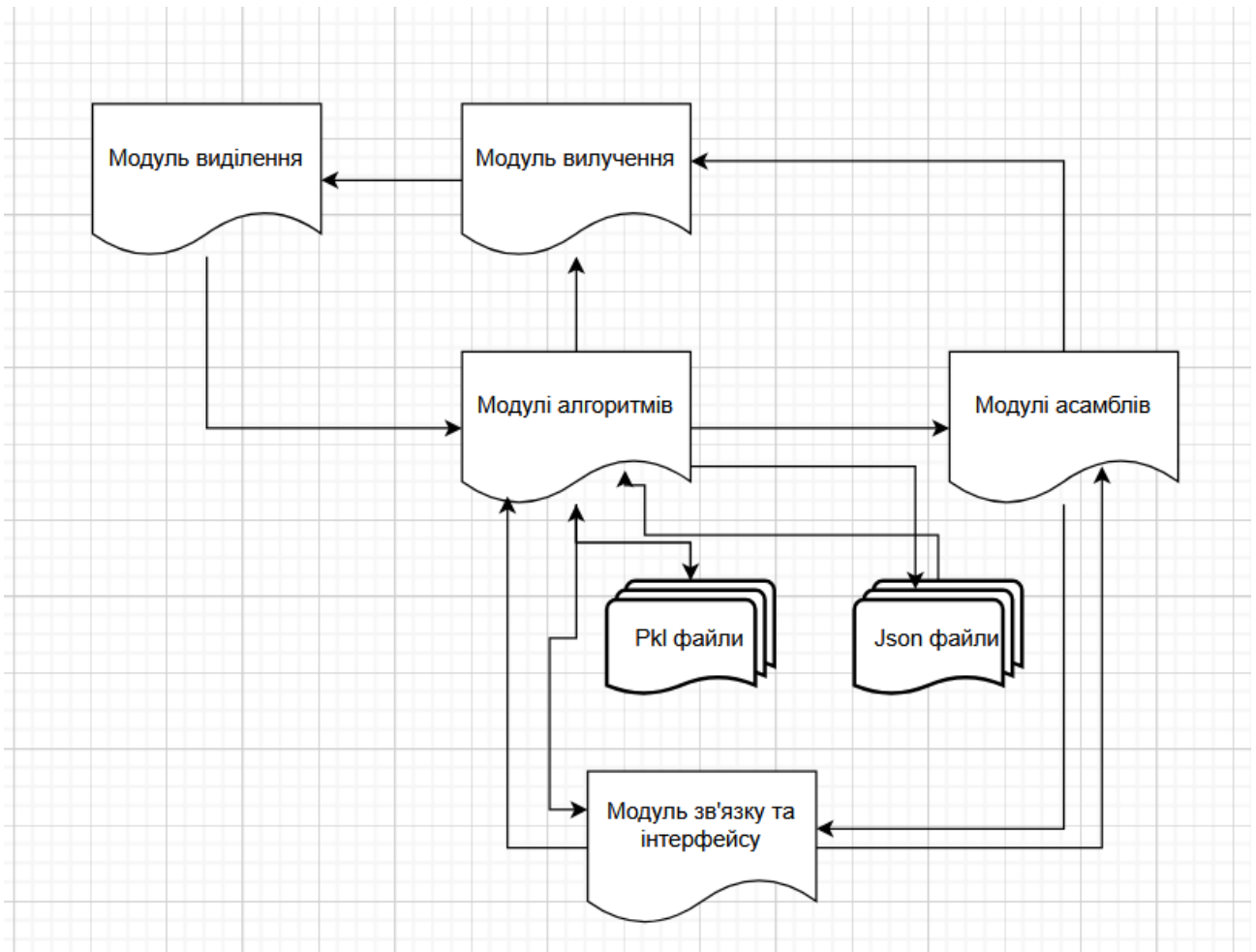


Рисунок 2.1 – Структурна схема програми

2.3 Опис модулів розроблюваної системи

Розроблюваний програмний продукт використовує дані зображень для виконання діагностування ока. До цих даних належать як виділенні особливості зображень, так і навчені на їх основі моделі. Ці дані можуть бути збережені в різних типах сховищ:

- текстові файли (CSV, txt, json, xml,...);
- бінарні файли даних;
- не реляційні бази даних (NoSQL сховища);
- реляційні бази даних (MySQL, MS SQL Server, Firebird, ...).

Для збереження вилучених ознак необхідні файли, що надають можливість легко і швидко працювати із складними і великими структурами даних, для додавання і вилучення інформації з них. Причому самі файли мають займати якомога менший об'єм пам'яті. До цих вимог підходить формат файлів Json. Серед його перваг є простота взаємодії, легка читабельність збережених даних, гнучкість підтримуваних структур.

Окрім цього необхідне середовище для зберігання створених моделей. Json не може зберігати більш складні елементи і обмежений простими типами даних. Для рішення цієї проблеми було використано формат Pkl, що здатний серіалізувати об'єкт моделі машинного навчання.

Модуль зв'язку та інтерфейсу – це основний модуль, що дає доступ користувачу до програми та об'єднує усі файли в один. Він включає в себе керування системами навчання та графічний вивід результатів роботи

Модуль вилучення вирізання зони інтересу – модуль призначений для виділення оптичного диску із зображення та його передачі модулю виділення. Він містить функцію для вирізання найсвітлішої ділянки зображення. Усі взаємодії із зображеннями виконуються за допомогою бібліотеки OpenCV. Зображення покращуються у якості вбудованими, після чого воно переводиться у формат HSV(тон, насиченість, значення кольору), з якого окремо виділяється канал значення кольорів [22]. У даному каналі видалюються усі пікселі, що мають значення менше за 97 відсотків від значення найсвітлішого пікселя зображення. Сформований канал переводиться у бінарний стан (чорно-білий) в якому визначаються усі контури з яких обирається найбільший [23], [24]. На оригінальне зображення накладається знайдений контур, після чого виділена частина вирізається. Зміни зображення відповідно до етапів показано на рис. 2.2 – 2.4.

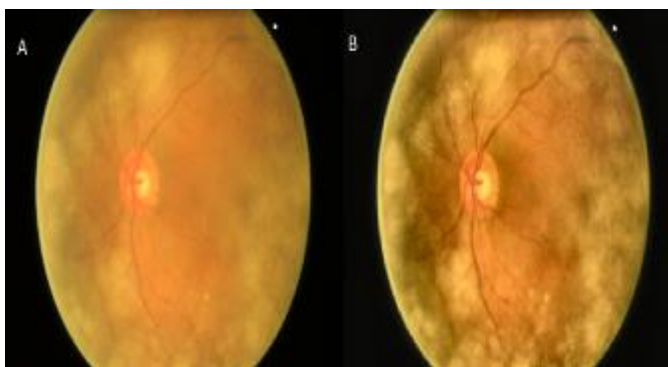


Рисунок 2.2 – А) Оригінальне зображення; В) покращена копія

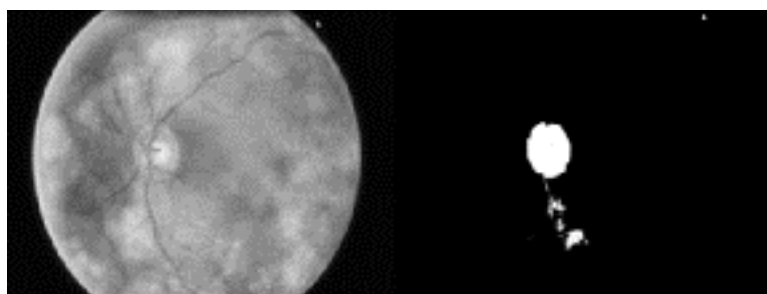


Рисунок 2.3 – А) Канал кольорів зображення; В) Бінарне зображення

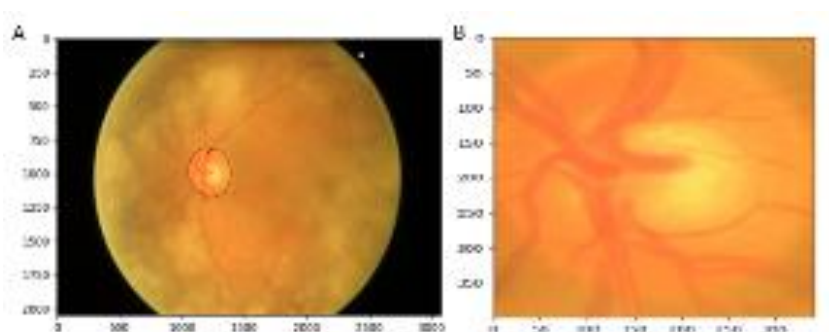


Рисунок 2.4 – А) Виділений контур на зображенні; В) Зона інтересу зображення

З вирізаної частини(зони інтересу) відбувається отримання особливостей зображення. Особливостями обраними для класифікацій зображень є середнє значення, стандартне відхилення, третій момент, симетричність. Середнє значення визначається суммацією значень усіх пікселів та її поділом на значення усіх пікселів, з нього виводяться усі інші показники. Ці значення надходять системам у фазу тренування та тестування.

Модуль виділення призначений для вилучення особливостей із зображень оптичного диску. Він виділяє наступні особливості:

- середнє значення;
- середньоквадратичне;
- стандартне відхилення;
- метод випадкового лісу;
- нахил;
- асиметрія.

Модулі алгоритмів приймають значення від модуля виділення та використовують їх для тренування та діагностування систе у залежності від команди модуля зв'язку. Серед розроблених методів є:

- метод n-ближчих сусідів;
- метод наївного байеса;
- метод K-Means (центроїдний кластерний аналіз);
- метод випадкового лісу;
- метод опорних векторів.

Модулі ансамблів використовують комбінації модулів алгоритмів для більшої точності діагнозу. Перед використанням ансамблів, інші моделі порівнюють свої результати с кластерами k-середнього та визначають до якого класу належить кожен з них. Серед реалізованих ансамблів є:

- просте голосування;
- метод опорних векторів.

2.4 Висновки за розділом 2

У другому розділі було розглянуто різноманітні середовища розробки та мови програмування, обрано один для зберігання даних ПП, проведено аналіз сильних і слабких сторін цих середовищ, побудовано структурну схему проекту та надано опис модулів проекту і зв'язків між ними.

У ході аналізу було виявлено сильні та слабкі сторони розглянутих середовищ, візуалізовано структурну схему та описано модулі проекту були і зв'язки між ними.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Алгоритм функціонування розробленої програми

Завданням даної роботи є програмна реалізація методів автоматизованого діагностування глаукоми на основі аналізу медичних зображень ока.

Для досягнення поставленої мети необхідно створити модуль попередньої обробки зображень, модуль виділення зони інтересу та модуль обчислення інформативних ознак, на основі яких здійснюється діагностика. Окрім цього, потрібно реалізувати та дослідити декілька методів класифікації для визначення приналежності зображення до класу «глаукома» або «норма».

Для виконання задачі у роботі використано такі алгоритми класифікації, як метод k найближчих сусідів (k -NN), наївний Байєс, Random Forest та ансамблеві методи. Використання декількох різних моделей дозволяє порівняти їхню точність, швидкість та стійкість до шумів, а також обрати найбільш ефективний метод для автоматизованої діагностики.

Високоточні алгоритми, такі як Random Forest, забезпечують достовірність класифікації, тоді як більш швидкі — наприклад, наївний Байєс — дозволяють отримати результат у найкоротший час. Детальний аналіз роботи кожного алгоритму наведено у наступних розділах.

3.1.1 Метод n -найближчих сусудів

k -NN — це тип класифікації, де функція апроксимується лише локально, а всі обчислення відкладено до оцінки функції.

Навчальні приклади в алгоритмі k -NN подаються у вигляді векторів у багатовимірному просторі ознак, кожному з яких відповідає певна мітка класу. Фаза навчання зводиться лише до збереження векторів ознак разом із відповідними мітками класів навчальної вибірки.

На етапі класифікації k є заданою користувачем константою, а немаркований вектор відноситься до того класу, який найчастіше зустрічається серед k найближчих до нього навчальних прикладів.

Часто використовуваною метрикою відстані для неперервних змінних є евклідова відстань. Для неперервних змінних зазвичай використовується евклідова метрика відстані, тоді як для дискретних ознак, можуть застосовуватися інші міри, зокрема метрика перекриття або відстань Геммінга. При аналізі мікромасивів даних експресії генів, наприклад, k -NN може використовувати коефіцієнти кореляції у якості метрики. Оскільки алгоритм базується на вимірюванні відстаней між об'єктами, у випадках, коли ознаки мають різні фізичні одиниці вимірювання або відрізняються за масштабами, необхідна попередня нормалізація навчальних даних для підвищення точності класифікації [26].

Однією з основних проблем алгоритму k -NN при використанні «голосування більшості» є перекошений розподіл класів. У таких випадках об'єкти домінуючого класу частіше потрапляють до множини k найближчих сусідів i , відповідно, визначають результат класифікації. Одним зі способів подолання цієї проблеми є зважене голосування, при якому внесок кожного з k сусідів залежить від відстані до тестової точки: ваги зазвичай обираються обернено пропорційними до відстані. Іншим підходом є узагальнення або абстрагування представлення даних [27].

Метод k -NN зазвичай використовується для класифікації, оскільки процес простий. Алгоритм k -NN вимагає кількох типів вхідних даних, таких як кількість сусідів k , матриця відстані та набір даних для навчання. Якщо $k = 1$, це називається класифікацією найближчого сусіда, яка визначає клас тестових даних на основі найближчого сусіда навчальних даних. Матриця відстані обчислюється за допомогою евклідової відстані (d), яка визначається за формулою (3.1):

$$d = \sqrt{\sum_{i=1}^c (x_i - y_i)^2}, \quad (3.1)$$

де i – індекс обраної ознаки;

x_i – матриця характеристик даних вибірки;

y_i – матриця характеристик тестових даних.

3.1.2 Наївний класифікатор Байєса

Наївний класифікатор Байєса — це алгоритм машинного навчання зх учителем, який використовується для вирішення задач класифікації. Він належить до класу генеративних моделей, оскільки описує ймовірнісний розподіл вхідних ознак для кожного класу або категорії. Робота алгоритму базується на припущенні, що характеристики вхідних даних умовно не залежать від класу, що дозволяє спростити обчислення та забезпечити високу швидкість при достатній точності прогнозування.

Наївний класифікатор Байєса є простим ймовірнісним класифікатором, який застосовує теорему Байєса. Ця теорема базується на ймовірності гіпотези, враховуючи дані та деякі попередні знання. Наївний класифікатор Байєса припускає, що всі ознаки у вхідних даних незалежні одна від одної. Незважаючи на те, що очевидно не правда, наївний байєсівський підхід залишається ефективним і демонструє високі результати в багатьох практичних застосуваннях.

Наївні класифікатори Байєса належать до найпростіших моделей байєсівських мереж, проте за умови поєднання з методами оцінювання щільності, зокрема оцінкою щільності ядра, вони можуть досягати високої точності. Унаслідок цього наївний класифікатор Байєса вважається потужним алгоритмом для вирішення таких задач, як класифікація тексту, фільтрації спаму та аналізу тональності.

Модель NB легко побудувати і краще використовувати для великих наборів даних. Відомо, що поряд із простотою наївний Байєс перевершує навіть дуже складні методи класифікації [28].

Теорема Байєса забезпечує спосіб обчислення апостеріорної ймовірності $P(c|x)$ із $P(c)$, $P(x)$ і $P(x|c)$ формула (3.2):

$$P(c|x) = \frac{P(x|c) \cdot P(c)}{P(x)}, \quad (3.2)$$

де $P(c|x)$ – апостеріорна ймовірність класу (c , ціль) із заданим предиктором (x , атрибути);

$P(c)$ – апріорна ймовірність класу;

$P(x|c)$ – ймовірність, яка є ймовірністю даного класу предиктора;

$P(x)$ – це пріоритетна ймовірність предиктора.

3.1.3 Метод випадкового лісу

Випадковий ліс (random forest) – алгоритм машинного навчання, що полягає у використанні ансамблю дерев рішень. Алгоритм поєднує в собі дві основні ідеї: метод баггінга і метод випадкових підпросторів. Алгоритм застосовується для задач класифікації, регресії і кластеризації.

У задачах регресії їх відповіді усереднюються, а в задачах класифікації приймається рішення голосуванням за більшістю [29]. Нехай дана навчальна вибірка розміру n . Генерується нових вибірок і розміру n , вибором з випадково з поверненням. Деякі спостереження можуть потрапити до вибірки кілька разів, деякі можуть не потрапити взагалі. Якщо n і велике, то частка різних спостережень в n і буде $(1 - 1/n) \approx 63,2\%$. Далі вивчається m класифікаторів на кожній вибірці D_i . При класифікації нової точки, ці класифікатори голосують і відносять точку до класу, за який проголосувала більшість. У методі випадкових підпросторів (random subspace method, RSM) класифікатори навчаються на різних підмножинах признакового опису, які також виділяються випадковим чином. Розглянемо алгоритм побудови випадкового лісу. Нехай навчальна

вибірка складається з прикладів, розмірність простору ознак дорівнює n , і заданий параметр ϵ (в задачах класифікації зазвичай $\epsilon \approx \sqrt{n}$).

Всі дерева комітету будуються незалежно один від одного за такою процедурою:

1. Згенеруємо випадкову підвибірку з повторенням розміром n з навчальної вибірки. (Таким чином, деякі приклади потраплять в неї кілька разів, а приблизно $n/3$ прикладів не ввійдуть в неї взагалі);

2. Побудуємо вирішальне дерево, що класифікує приклади такої підвибірки, причому в ході створення чергового вузла дерева будемо вибирати ознаку, на основі якої проводиться розбиття, не з усіх ознак, а лише з випадково обраних. Вибір найкращого з цих ознак може здійснюватися різними способами. В оригінальному коді Бреймана використовується критерій Джині. У деяких реалізаціях алгоритму замість нього використовується критерій приросту інформації;

3. Дерево будується до повного вичерпання підвибірки. Оцінка ймовірності помилкової класифікації випадкового лісу здійснюється методом "Out-Of- Bag". Так як, кожна бутстреп вибірка не містить приблизно 37% спостережень вихідної навчальної вибірки (оскільки вибірка з поверненням, то деякі спостереження в неї не потрапляють, а деякі – декілька раз). Класифікуємо деякий вектор x . Для класифікації використовуються тільки ті дерева випадкового лісу, які будувалися по бутстреп вибіркам, що не містять x , і як звичайно використовується метод голосування.

3.1.4 Кластерний аналіз

Кластерний аналіз, є статистичною процедурою, задачею якої є поділ множини об'єктів на неперетинні підмножини — кластери. Кожен кластер складається з об'єктів, що мають високий ступінь подібності між собою, а об'єкти, що належать до різних кластерів, повинні суттєво відрізнятися. Завдання

кластеризації належить до методів статистичної обробки даних і входить до класу задач навчання без учителя [30].

Дається вибірка елементів $X=\{x_1, x_2, \dots, x_n\}$, де кожен елемент це точка у гіперпросторі. Кластерний аналіз намагається розбити гіперпростір на частини, що належать елементам одного класу. Основна ідея цього рішення полягає у тому, що об'єкти одного кластера є більш подібними між собою, ніж до об'єктів інших кластерів.

Залежно від способів віднесення об'єктів до кластерів розрізняють жорстку та м'яку кластеризації. У м'якій кластеризації об'єкт може з певною ймовірністю належати одному або кільком кластерам. У жорсткій кластеризації кожен об'єкт належить або одному кластеру, або не належить жодному з них.

У загальних випадках кластеризація використовується для дослідження даних і їх структури. Рекомендується проводити цю процедуру кілька разів для узагальнення результатів. Моделі, що використовують цей алгоритм, застосовують в таких областях, як аналіз зображень, машинне навчання, маркетингової сегментації, розпізнавання образів, детекції аномалій, стиснення даних. Цей алгоритм слугує допоміжним інструментом до інших методів, оскільки окрема кластеризація не завжди здатна повністю відобразити всю інформацію, що міститься у наборі даних.

Найпоширенішим методом кластеризації є метод к-середніх. Він намагається розділити елементи вибірки у різні ділянки гіперплощини, де кожен елемент належить до класу найближчого центроїду у гіперпросторі. Його перевагою є простір для кластеризації великих вибірок даних, на відміну від ієрархічних методів, де дендрограми швидко стають переповненими та втрачають наочність. Водночас до недоліків цього підходу належать чутливість до викидів та неможливість роботи з категоріальними даними.

Алгоритм виглядає так:

1. Введення кількості кластерів, що необхідно визначити.
2. Для кожного кластеру випадково обрати по об'єкту із вибірки – центроїдів кластерів.

3. Для кожного елементу, визначити євклідову відстань до кожного з центроїдів і надати мітку класу найближчого з них.

4. Для кожного кластеру обчислити нові центроїди як середнє значення всіх точок даних, що належать цьому кластеру.

5. Повторювати кроки 3 і 4, поки центроїди не пересунуться на визначену мінімальну відстань або не буде досягнуто максимальної кількості ітерацій.

Стандартний алгоритм визначає загальну варіацію в межах кластера як суму квадратичних євклідових відстаней між елементами і відповідним центроїдом формула (3.3):

$$W(C_k) = \sum_{x_i \in C_k} (x_i - u_k)^2, \quad (3.3)$$

де x_i – точка даних, що належить до кластера C_k ,

u_k – середнє значення кластеру C_k .

Кожен елемент вибірки присвоюється заданому кластеру таким чином, щоб сума квадратів відстані спостереження до їх призначених кластерних центрів u_k мінімізувалася.

Загальна сума, або загальна варіація в межах кластера визначається таким чином, формула (3.4):

$$\sum_{k=1}^k W(C_k) = \sum_{k=1}^k \sum_{x_i \in C_k} (x_i - u_k)^2 \quad (3.4)$$

3.1.5 Метод опорних векторів

Мащини опорних векторів – це набір алгоритмів машинного навчання з учителем та моделі створені на їх основі, які аналізують дані для класифікації та регресійного аналізу [31].

Надано деякі задані точки даних належать до одного з двох класів, і визначити клас нової точки даних. Головна задача алгоритму у цьому випадку — знайти лінійну площу у просторі у двомірному просторі, що розділяє множини

точок так, щоб у кожній були лише точки одного класу. У випадку методів опорних векторів точка даних подається у вигляді p -вимірному вектору (списку чисел p), після чого перевіряється, чи можливо розділити такі точки за допомогою $(p - 1)$ -вимірної гіперплощини. Це називається лінійним класифікатором. У випадку, коли знаходяться кілька підходящих площ, лінійним класифікатором обирається та, в якій найбільша відстань між найближчими точками класів — запас. Якщо така гіперплощина існує, то її називають гіперплощиною з максимальним запасом, а відповідний лінійний класифікатор — класифікатором з максимальним запасом.

Якісне розділення досягається коли гіперплощина має найбільшу відстань до найближчої точки навчальних даних будь-якого класу, що визначає відстань. Чим більша ця відстань, тим, меншою є помилка узагальнення класифікатора і нижче ризик перенавчання моделі.

Початкова множина може бути представлена у просторі з певною кількістю вимірів, де неможливо лінійно розділити множини. Для рішення цієї проблеми було розроблено алгоритм, у якому початковий простір був відображений у простір з більшою кількістю вимірів, де з більшою ймовірністю можна буде провести лінійний розподіл [32]. Для його виконання рекомендується визначати скалярні добутки пар векторів точок через обрану функцію ядра $k(x, y)$.

Гіперплощини у багатовимірному просторі визначаються як множина точок, скалярний добуток яких з вектором у цьому просторі є константою, де така множина векторів є ортогональною і мінімальною множиною векторів, що визначає гіперплощину. Вектори, що визначають гіперплощини, можна вибрати як лінійні комбінації з параметрами α_i зображень векторів ознак x_i , що зустрічаються в наданій вибірці. З таким вибором гіперплощини точки x у просторі ознак, які відображаються в гіперплощину, визначаються співвідношенням (3.5):

$$\sum_i \alpha_i k(x_i, x) = \text{constant}. \quad (3.5)$$

Коли функція ядра $k(x, y)$ зменшується при збільшенні відстані між x та y , кожен член у сумі вказує ступінь близькості тестової точки x до відповідної точки бази даних x_i . Це дозволяє використовувати суму ядер для вимірювання відносної близькості кожної тестової точки до точок даних, усіх множин, що необхідно розрізнити. Важливо зазначити, що множина точок відображена в будь-яку гіперплощину, може мати досить складну форму, що полегшує процес розрізнення множин, які взагалі не є опуклими у вихідному просторі.

Для розширення SVM на випадки, коли дані не є лінійно роздільними, можна застосувати функцію втрат шарніра (3.6):

$$\max(0, 1 - y_i(w^T x_i - b)), \quad (3.6)$$

де y_i — це i -та ціль;

$w^T x_i - b$ — це i -й вихід.

Ця функція дорівнює нулю, якщо x_i лежить на правильній стороні поля. Для даних на неправильній стороні поля значення функції пропорційне відстані від поля.

Мета оптимізації полягає в мінімізації (3.7):

$$\|w^2\| + C \left[\frac{1}{n} \sum_{i=1}^n \max(0, 1 - y_i(w^T x_i - b)) \right], \quad (3.7)$$

де параметр $C > 0$ визначає компроміс між збільшенням розміру поля та забезпеченням того, щоб x_i лежали на правильній стороні поля. Деконструюючи втрати шарніра, цю задачу оптимізації можна сформулювати наступним чином (3.8), (3.9):

$$\underset{w, b, \zeta}{\text{minimize}} \quad \|w^2\| + C \sum_{i=1}^n \zeta_i \quad (3.8)$$

$$\text{subject to } \zeta_i(w^T x_i - b) \geq 1 - \zeta_i, \zeta_i \geq 0 \text{ } \forall i \in \{1, \dots, n\} \quad (3.9)$$

Для великих значень C , алгоритм такий же, як і для SVM із жорстким запасом.

Оригінальний алгоритм гіперплощини з максимальним запасом, використовується для класифікації лінійних даних. Однак у 1992 році було запропоновано спосіб створення нелінійних класифікаторів, застосовуючи ядерний трюк до гіперплощин з максимальним запасом. Ядерний трюк — метод переведення елементів, що не піддаються лінійному розподілу, у простір з більшою кількістю вимірів. Його суть у тому, що скалярні добутки замінюються на ядра, які можна легко представити у лвомірному просторі. Це дозволяє алгоритму підігнати гіперплощину з максимальним запасом у перетвореному просторі ознак. Перетворення може бути нелінійним, а перетворений простір — багатовимірним; хоча класифікатор є гіперплощиною у перетвореному просторі ознак, він може бути нелінійним у вихідному вхідному просторі.

Робота у багатовимірному просторі ознак збільшує похибку узагальнення методів опорних векторів, для n .

Ядро пов'язане з перетворенням $\phi(x_i)$ рівнянням (3.10):

$$k(x_i, x_j) = \phi(x_i) \cdot \phi(x_j). \quad (3.10)$$

Значення w також знаходиться у перетвореному просторі, де (3.11):

$$w = \sum_i a_i y_i \phi(x_i). \quad (3.11)$$

Скалярні добутки з w для класифікації можна обчислити за допомогою ядерного трюку, тобто (3.12):

$$w = \sum_i a_i y_i k(x_i, x). \quad (3.12)$$

Обчислення SVM-класифікатора зводиться до мінімізації виразу виду (3.13):

$$\left[\frac{1}{n} \sum_{i=1}^n \max(0, 1 - y_i(w^T x_i - b)) \right] + \lambda \|w\|^2 \quad (3.13)$$

Тепер можна використати нелінійне правило класифікації, яке відповідає лінійному правилу класифікації для перетворених точок даних $\phi(x_i)$. Більше того, нам задана функція ядра k , яка задовольняє умову $k(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$.

Вектор класифікації w у перетвореному просторі задовольняє (3.14):

$$w = \sum_{i=1}^n c_i y_i \phi(x_i), \quad (3.14)$$

де c_i отримуються шляхом розв'язання задачі оптимізації (3.15), (3.16):

$$\begin{aligned} \text{maximize } f(c_1 \dots c_n) = \sum_{i=1}^n c_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n y_i c_i (\phi(x_i) \cdot \phi(x_j)) y_j c_j = \\ \sum_{i=1}^n c_j - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n y_i c_i k((x_i, x_j)) y_j c_j \end{aligned} \quad (3.15)$$

$$\text{subject to } \sum_{i=1}^n y_i c_i = 0, \text{ and } 0 \leq c_j \leq \frac{1}{2n\lambda} \text{ for all } i. \quad (3.16)$$

Коефіцієнти c_i можна розв'язати за допомогою квадратичного програмування, як і раніше. Можна знайти деякий індекс i такий, що $0 < c_i < \frac{1}{2n\lambda} - 1$, так що $\phi(x_i)$ лежить на межі поля у перетвореному просторі, а потім розв'язати (3.17):

$$b w^T \phi(x_i) - y_i = \left[\sum_{j=1}^n c_j y_j \phi(x_j, x_i) \right] - y_i. \quad (3.17)$$

І наостанок розв'язується (3.18):

$$z \rightarrow \operatorname{sgn}(w^T \varphi(z) - b) = \operatorname{sgn}([\sum_{i=1}^n c_i y_i k(x_i, z)] - b). \quad (3.18)$$

3.1.6 Ансамблі методів машинного навчання

Ансамблеві машинного навчання — це методи використання кількох алгоритмів навчання для підвищення їх точності до значення більшого за кожне з них окремо [33].

Ансамблі складаються з двох і більше алгоритмів, з одного або кількох видів, які називаються базовими. Головною умовою для вибору алгоритмів є висока різноманітність: це потрібно щоб, при хибному рішенні одного алгоритма у певній ситуації, інші могли виправити його помилку. Це призводить до підвищення їх точності класифікації, ціною усереднення [34].

Розрізняють три види ансамблевого навчання: «пакування» — використання одних й тих же моделей, «підвищення» — навчання базових моделей на помилках попередніх та «змішування» — використання одних й тих же моделей. Прикладами ансамблів слугують АдаБуст, градієнтний бустінг, випадковий ліс, паковані дерева рішень, паковані машини опорних векторів, блендінг.

Одним з видів ансамблів пакування та змішування є ансамбль голосування. Суть його роботи в тому, що кожна модель класифікує елемент вибірки і ансамбль сумує ці відповіді, обираючи кінцевий клас для елемента залежно від його типу: простого чи зваженого.

У простому голосуванні кожна модель має рівноцінний голос і елементу призначається клас, що було призначено більшістю з них. У зваженому глосуванні кожна модель має визначену ймовірність правильного вибору кожного класу. У цьому ансамблі моделі, сума ймовірностей сумується та усереднюється, після чого обирається клас з найбільшою ймовірністю

3.2 Огляд особливостей метрик інтелектуального аналізу

Виміри простору ознак складаються з наступних особливостей [35]:

Середнє арифметичне набору даних дорівнює сумі числових значень кожного елементу набору, що поділена на кількість елементів набору. Для набору даних $X=\{x_1,...,x_n\}$, середнє арифметичне визначається за формулою (3.19):

$$\bar{x} = \frac{1}{n} (\sum_{i=1}^n x_i) = \frac{x_1+x_2+\dots+x_n}{n} \quad (3.19)$$

де x - значення змінної;

n - загальна кількість змінних.

Середнє арифметичне використовується для визначення центральних тенденцій вибірки.

Стандартне відхилення або середнє квадратичне відхилення — це величина стандартного відхилення вибіркового середнього, розраховане по вибірці із генеральної сукупності. Воно вимірює ступінь розсіювання значень у наборі даних, відносно їх середнього значення. Чим більше значення відхилення, тим більш послідовні значення.

Стандартне відхилення σ від X визначається за формулою (3.20):

$$\sigma = \sqrt{\int_{-\infty}^{+\infty} (x - \mu)^2 f(x) dx} \quad (3.20)$$

де x - значення змінної;

μ - середнє значення змінних.

Гостровершинність (або ексцес) — це міра форми розподілу даних, що описує концентрацію значень навколо середнього.

Висока гостровершинність свідчить про те, що більшість даних згруповано близько до середнього, але є і відхилення, тоді як плаский розподіл вказує на більш рівномірний розподіл даних. Визначається за формулою (3.21):

$$K = \frac{(x-\mu)^2}{\sigma^4} \quad (3.21)$$

У теорії ймовірностей і статистиці асиметрія - це числова характеристика, яка показує наскільки розподіл ймовірностей випадкової величини відхиляється від симетричного відносно середнього значення. Асиметрія може бути додатною, від'ємною, нульовою або невизначеною.

При унімодальному розподілі додатня асиметрія означає, що "хвіст" розподілу витягнутий праворуч, а від'ємна що "хвіст" розподілу витягнутий ліворуч.

Однак є випадки, коли інтерпретація асиметрії не така однозначна. Це зазвичай відбувається при симетричному розподілі, коли один "хвіст" довгий і тонкий, а інший товстий і короткий.

Тому оцінювати симетричність розподілу лише за значенням асиметрії ризиковано — необхідно враховувати форму самого розподілу.

Асиметрія випадкової величини X є третім стандартизованим моментом, який визначається як (3.22):

$$\gamma_1 = \frac{k^3}{k_2^{3/2}}, \quad (3.22)$$

де μ – середнє;

σ – стандартне відхилення;

E – оператор очікування;

μ_3 – третій центральний момент;

kt – t -і кумулянти.

3.3 Висновки за розділом 3

У третьому розділі було розглянуто алгоритми пошуку шляхів і описані їх модифікації.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови застосування програми

Програма призначена для автоматизованої діагностики глаукоми на основі аналізу медичних зображень ока. Програма виконує наступні функції:

- завантаження та попередня обробка зображень;
- виділення зони інтересу та обчислення інформативних ознак;
- виконання класифікації за допомогою реалізованих алгоритмів (KNN, Naïve Bayes, Random Forest, SVM, ансамблеві методи);
- виведення результатів діагностування у графічному інтерфейсі.

Програма має наступні апаратні вимоги для нормального функціонування програми. Щоб забезпечити оптимальну продуктивність, важливо відповідати цим специфікаціям або перевищувати їх.

Ось короткий перелік вимог до обладнання:

- процесор: Pentium-сумісна архітектура x86 з частотою 5 ГГц або вище, підтримкою PAE і NX SSE2;
- пам'ять: 8 Гб оперативної пам'яті;
- відеокарта: DirectX 12 і WDDM версії 2.0 або новішої;
- дисковий простір: 20 ГБ або більше вільного дискового простору;
- роздільна здатність екрана: 800 x 600 або вище;
- пристрій введення: клавіатура та миша або аналогічний пристрій введення;
- операційна система Windows версії 10 або вище.

Для нормального функціонування програми потрібна наявність операційної системи Windows версії 10 або вище.

4.2 Характеристики програми

Додаток виконує наступні функції:

- графічне відображення завантажених медичних зображень ока;
- виділення зони інтересу;
- обчислення ознак зображення;
- навчання систем машинного навчання;
- класифікація зображення
- збереження обчислених ознак та навчених систем класифікації у файл
pkl;
- візуалізація результатів діагностування.

4.3 Звертання до програми

Для доступу до програми, потрібно запустити файл index.py як звичайний скрипт пайтон, впевнившись, що усі інші модулі пристуні у папці з програмою.

4.4 Початкові та вихідні дані

Початковими даними програми є дані зображення очей.

Вихідними даними є списки класифікованих зображень з діагнозами.

4.5 Повідомлення програмісту

Під час роботи розроблений ПП виводить у консоль та у вікно список зображень з діагнозом, що можна побачити на рис.4.1.

```

158.0657 21.7651 0.9979 -1.2126 1.0483
164.6778 30.6665 0.9989 -1.0141 0.2179
183.681 5.2778 0.9653 -0.6472 -0.5611
150.8942 10.0235 0.9901 -0.6551 0.2485
187.4023 21.2681 0.9978 -1.6864 1.8305
175.948 6.457 0.9766 -0.0801 -0.6375
188.1894 12.0584 0.9932 -1.4169 3.1734
186.5188 10.0798 0.9903 -1.5161 2.429
174.3071 5.5616 0.9687 -1.703 6.296
165.4327 24.3821 0.9983 -2.0282 3.3326
211.05 6.2027 0.9747 -1.4213 2.0436
176.1574 23.7146 0.9982 -0.9504 -0.71
Negative: 38, Positive: 63

```

Рисунок 4.1 – Приклад повідомлення програмісту

4.6 Висновок

У даному розділі було детально розглянуто призначення та функції розробленого програмного продукту. Визначено умови, необхідні для його встановлення та підтримки як на комп'ютері користувача, так і на сервері. Описано основні характеристики та особливості створеного програмного забезпечення. У додачу надано опис вхідних та вихідних даних.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми

Додаток призначений для часткової автоматизації процесу діагностування глаукоми шляхом аналізу медичних зображень ока. Одним із основних завдань системи є автоматичне виявлення ознак, характерних для глаукоми, та визначення належності зображення до класу «хворе» або «здорове» за допомогою реалізованих алгоритмів машинного навчання.

5.2 Умови виконання програми

Для роботи програми, потрібно запускати пайтон скрипт (.py), тож необхідне встановлене середовище python.

Програмний продукт призначений для запуску в середовищі Windows з наступними параметрами:

- процесор: Pentium-сумісна архітектура x86 з частотою 5 ГГц або вище, підтримкою PAE і NX SSE2;
- пам'ять: 8 Гб оперативної пам'яті;
- відеокарта: DirectX 12 і WDDM версії 2.0 або новішої;
- дисковий простір: 20 Гб або більше вільного дискового простору;
- роздільна здатність екрана: 800 x 600 або вище;
- пристрій введення: клавіатура та миша або аналогічний пристрій введення;
- операційна система Windows версії 10 або вище.

5.3 Виконання програми

Після вдалого запуску ПП відкривається вікно програми, що представлено на рис. 5.1. У цьому вікні відображається інтерфейс для аналізу медичних зображень ока. Основним елементом є область перегляду, в якій

користувач може бачити завантажені знімки та результати їх класифікації. Для зручності перегляду результати виділяються різними кольорами: зеленим позначаються зображення, класифіковані як “здорові”, а червоним – зображення, визначені алгоритмом як “хворі”.

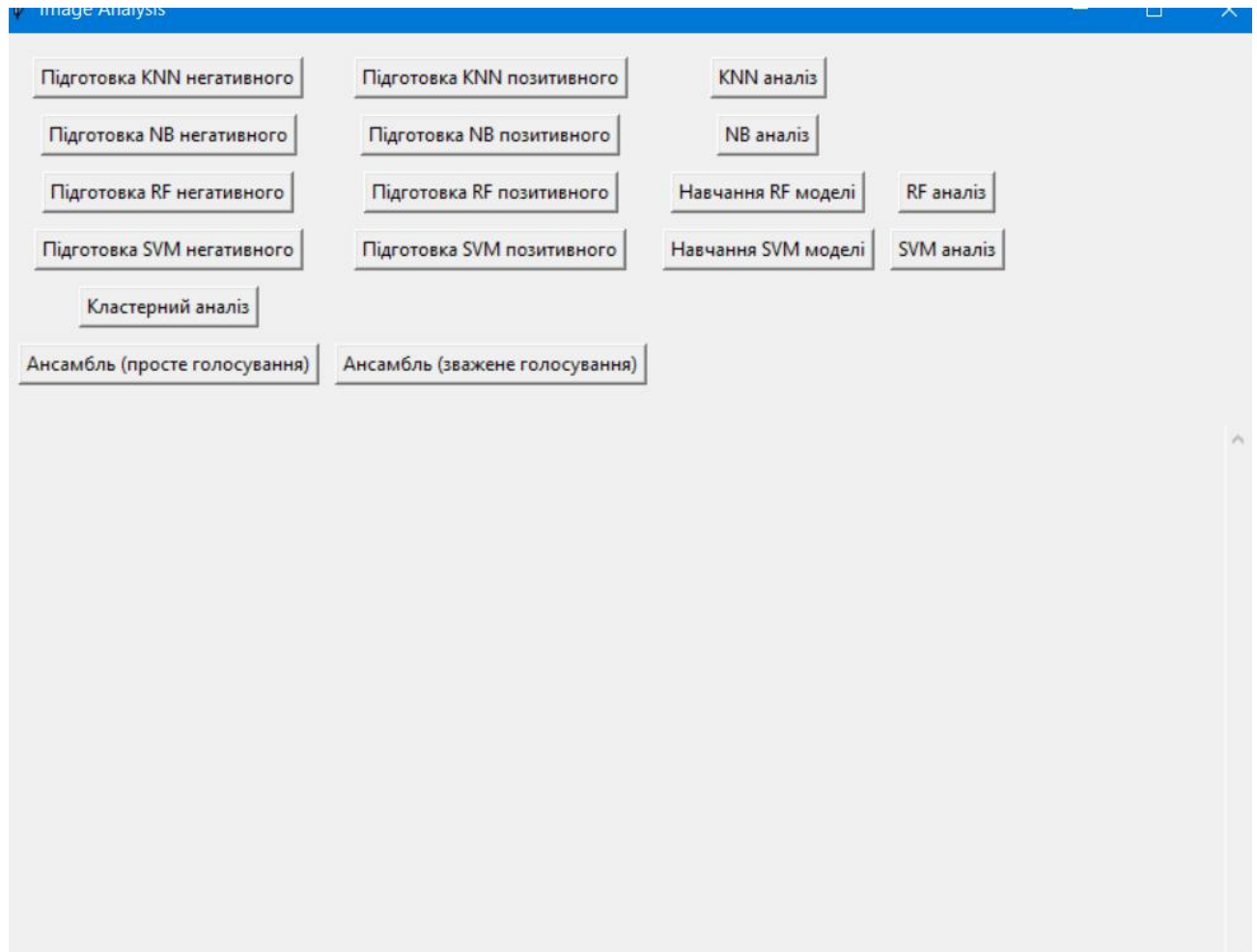


Рисунок 5.1 – Програма при запуску

Після запуску користувач може завантажити необхідні медичні зображення за допомогою відповідних кнопок інтерфейсу. Для коректної роботи програми зображення повинні бути попередньо підготовлені та розташовані у структурованих папках.

Далі користувач обирає потрібні алгоритми або натискає кнопки підготовки даних для кожного методу класифікації (KNN, Naive Bayes, Random Forest, SVM або ансамблеві методи). Після цього за допомогою кнопки аналізу

можна запустити класифікацію нових зображень. Усі результати обробки автоматично відображаються у вигляді списку, що можна побачити на рис. 5.2.

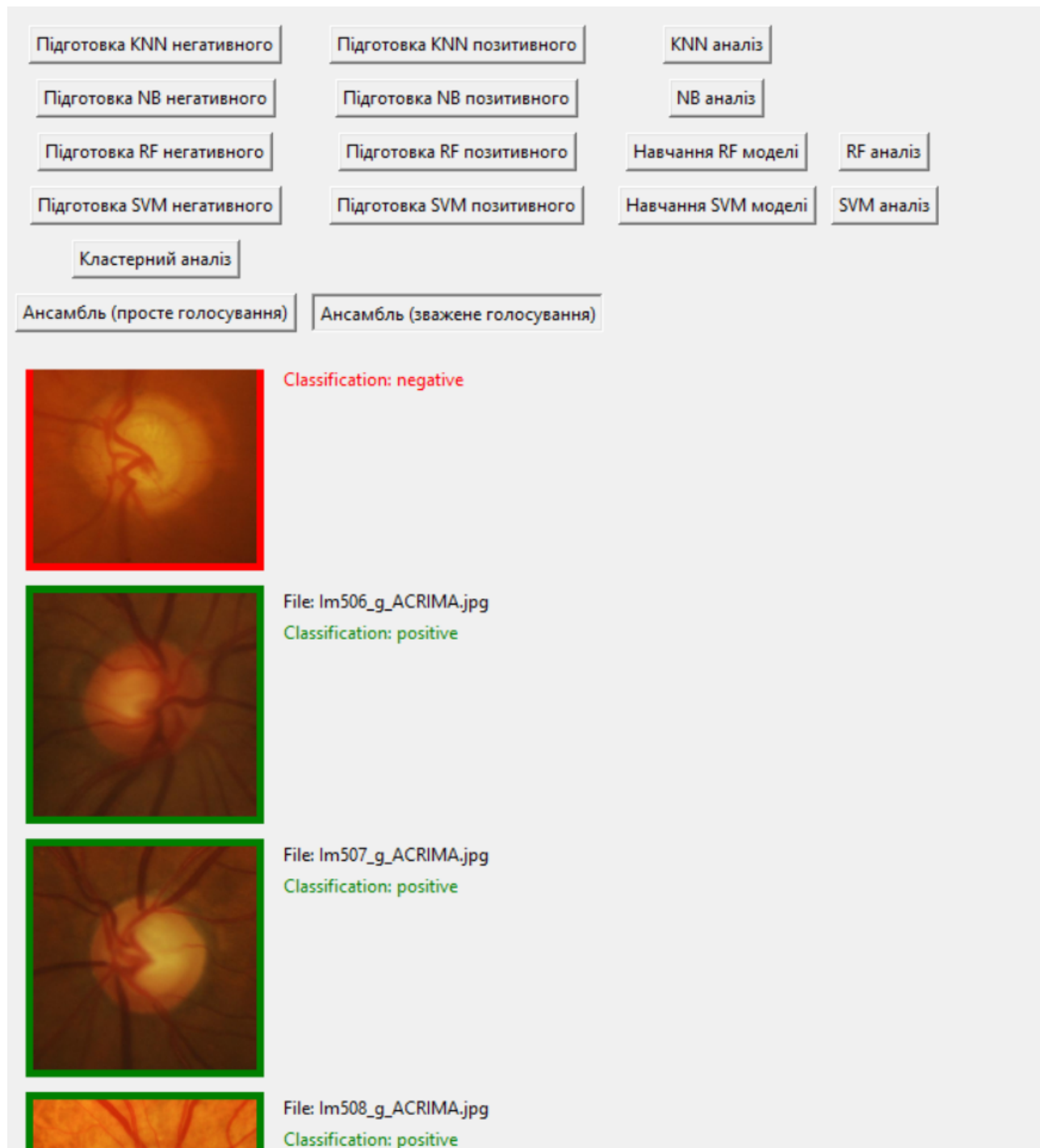


Рисунок 5.2 – Виведення результатів

Після завершення обчислень програма показує для кожного зображення його клас та мініатюру, а також дозволяє користувачу візуально оцінити отримані результати. Якщо потрібно повторно провести аналіз або додати нові дані, програма дає змогу це зробити без перезавпуску, зберігаючи попередні результати в інтерфейсі.

Таким чином, програма у реальному часі виконує класифікацію медичних зображень, забезпечуючи швидку первинну автоматизовану діагностику глаукоми.

5.4 Повідомлення оператору

Під час виконання розробленого програмного продукту автоматично формуються повідомлення для інформування оператора про виникнення помилок, початку та завершення процесу та інших. Ці повідомлення не потребують втручання оператора. Перелік можливих повідомлень, їх зміст і відповідні дії, що виконуються, описані у розділі "Керівництво програміста". Користувач бачить графічне відображення всіх кроків виконання програми.

5.5 Висновок

У цьому розділі було детально описано призначення та функції розробленого програмного продукту, а також мінімально необхідні апаратні вимоги для його використання. Описано принципи взаємодії оператора з програмним продуктом. Представлено перелік повідомлень, які генеруються під час роботи програми. Наведено приклади вхідних та вихідних даних. Інтерфейс користувача розробленого програмного продукту є простим та інтуїтивно зрозумілим і не вимагає додаткового навчання для його операційного персоналу.

6 ТЕСТУВАННЯ І РЕЗУЛЬТАТИ

У роботі наведено метод обробки особливостей зображень, заснований на вилученні спеціальних математичних особливостей заснованих на характеристиках пікселей зображення. Ці особливості дають інформацію про зміни у просторі зображення у вигляді точок, країв та об'єктів, що виділяються у зображенні. На рисунку 6.1 та рисунку 6.2 можна побачити значення особливостей оброблених зображень здорових та хворих очей відповідно. Рисунок 6.3 та рисунок 6.4 показують графіки їх стандартного відхилення та середнього значення, а рисунок 6.5 та рисунок 6.6 – середньоквадратичного відхилення, нахилу та симетричності зображень здорових та хворих очей. На рисунку 6.7 зображено середні значення особливостей хворих та здорових очей.

Середнє	Стандарт	Середньо	Нахил	Симетричність
120,1382	30,914	0,999	1,4756	1,8521
142,2805	22,433	0,998	1,8812	4,8178
142,4136	16,3359	0,9963	0,452	0,0575
143,8878	34,0084	0,9991	0,115	-0,2749
145,5946	38,082	0,9993	0,3235	-0,4503
148,4092	33,4064	0,9991	0,4077	-0,9211
148,6205	35,0093	0,9992	0,9008	0,2305
152,1596	32,636	0,9991	1,0236	0,2328
152,6652	35,7075	0,9992	0,3582	-0,687
155,1497	26,2275	0,9985	0,4612	-0,8074
156,8304	38,0055	0,9993	0,4389	-0,7978
156,9007	22,3374	0,998	0,0155	-0,3966
157,6289	26,8029	0,9986	0,8329	0,3782
157,7968	39,0685	0,9993	-0,0332	-0,8488
157,8877	26,3418	0,9986	0,8431	0,4507
159,5874	28,5238	0,9988	0,3665	0,0207
159,724	29,693	0,9989	0,077	-0,7324
160,4954	29,3574	0,9988	0,3948	0,0571
160,5681	26,2299	0,9985	0,3405	-0,1495
160,8676	34,391	0,9992	0,5231	0,2546
161,2228	41,0192	0,9994	0,0979	-0,9257
161,4685	36,1287	0,9992	0,488	-0,4346
163,61	44,2121	0,9995	0,6727	-0,4453
164,9867	36,8435	0,9993	0,348	-0,6075
165,2853	40,7958	0,9994	0,1539	-0,6958
165,6187	50,7763	0,9996	0,1716	-0,8692
165,7387	32,9399	0,9991	0,6919	0,3952
166,2879	37,3556	0,9993	0,3873	-0,8522
166,9214	24,5808	0,9983	0,4487	-0,2962

Рисунок 6.1 – Таблиця особливостей зображень здорового ока

Середнє	Стандартне	Середньо	Нахил	Симетрич
109,9145	24,8974	0,9984	0,7302	1,0033
118,8475	26,515	0,9986	1,0831	1,9741
118,8862	23,055	0,9981	1,44	3,8219
121,3836	16,3117	0,9963	2,1474	5,2323
124,7003	6,8113	0,9789	-0,256	-0,8429
131,2645	14,964	0,9956	2,1492	6,0521
131,6755	13,6545	0,9947	2,137	6,7526
132,0342	18,3878	0,9971	1,119	2,9128
133,2538	29,8589	0,9989	1,0616	0,8352
133,5907	14,3645	0,9952	0,7938	0,0879
135,8177	14,215	0,9951	1,1616	1,7715
136,2263	33,8043	0,9991	1,0261	1,0906
136,5863	19,6971	0,9974	0,5143	4,9551
139,5907	28,8155	0,9988	1,2055	0,5933
140,4851	31,4879	0,999	0,9259	0,8188
145,0682	32,7039	0,9991	0,7913	-0,1902
145,773	22,864	0,9981	0,5069	-0,5143
147,6759	33,2959	0,9991	1,2988	0,9395
148,0591	38,892	0,9993	0,7399	-0,2339
150,6764	26,1283	0,9985	1,9908	4,5609
151,1074	31,948	0,999	1,1418	1,4957
153,368	38,7868	0,9993	0,3543	-0,6806
158,2881	29,1061	0,9988	0,5992	-0,5209
159,9547	40,18	0,9994	0,3456	-0,8605
160,4964	39,6162	0,9994	0,6513	-0,3536
160,9071	38,38	0,9993	0,847	-0,5287
161,558	33,1798	0,9991	0,632	-0,458
161,8373	28,8104	0,9988	0,1482	-0,8056
161,8744	24,0992	0,9983	0,4553	-0,5675

Рисунок. 6.2 – Таблиця значень особливостей зображень хворих очей

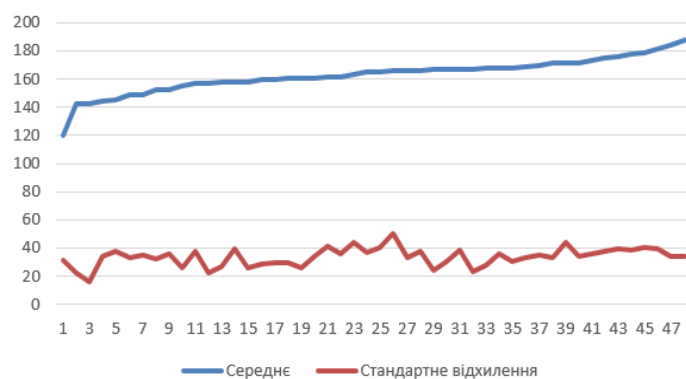


Рисунок 6.3 – Графік стандартного відхилення та середнього значення зображень здорових очей

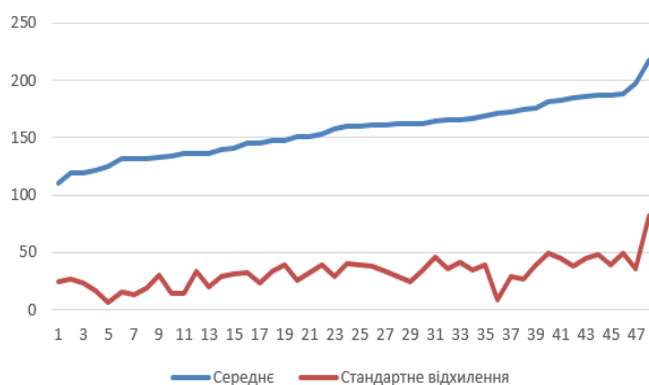


Рисунок. 6.4– Графік стандартного відхилення та середнього значення зображень хворих очей

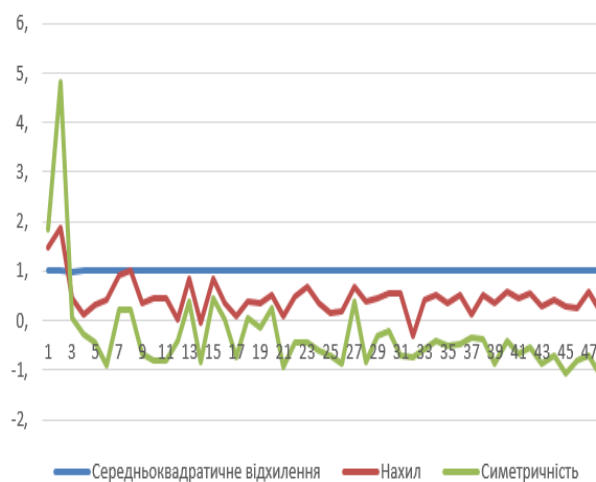


Рисунок 6.5 – Графік значень середньоквадратичного відхилення, нахилу та симетричності зображень здорових очей

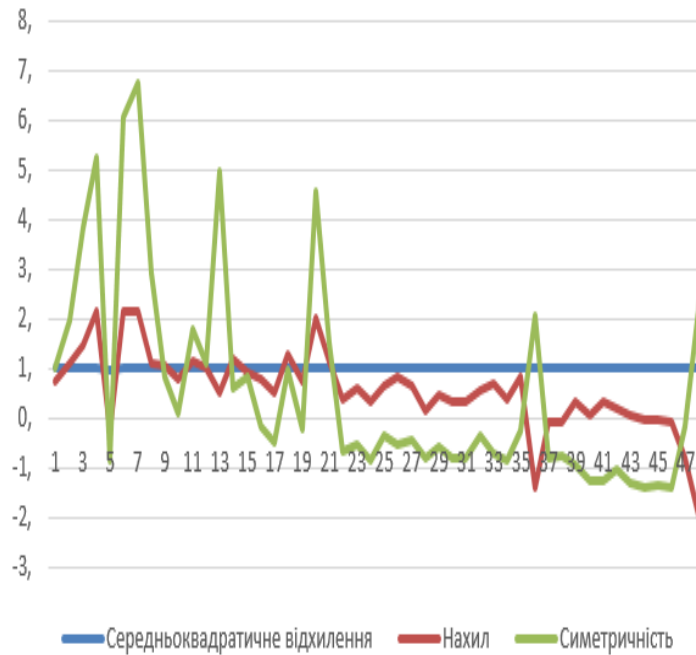


Рисунок 6.6 – Графік значень середньоквадратичного відхилення, нахилу та симетричності зображень хворих очей

	Середнє			Стандартне			Середньоквадратичне			Нахил			Симетричність		
	Мінімум	Максимум	Середнє	Мінімум	Максимум	Середнє	Мінімум	Максимум	Середнє	Мінімум	Максимум	Середнє	Мінімум	Максимум	Середнє
Здорове	120,1382	187,5988	162,736	16,3359	50,7763	33,67374	0,996	1,	0,999	-0,3059	1,8812	0,455044	-1,1245	4,8178	-0,30484
Хворе	109,9145	218,1582	156,503	6,8113	82,4447	32,14302	0,979	1,	0,998	-2,0578	2,1492	0,5668	-1,4001	6,7526	0,56639

Рисунок 6.7 – Значення особливостей

Як можна побачити на зображеннях, середнє значення здорових очей має менший діапазон значень, ніж у хворих. У стандартному відхиленні здорові очі також мають менший діапазон значень, однак їх значення у цілому трохи різноманітніші за хворі, які ростуть рахом із середнім значенням. Середньоквадратне відхилення майже однакове у обох випадках. Значення нахилу здорових очей має більший мінімум та менше флуктуацій і має набагато менший спад відносно росту середнього значення. Значення симетричностей у хворих очей має високий максимум та дуже багато флуктуацій і дуже швидко

спадає із ростом середнього значення.

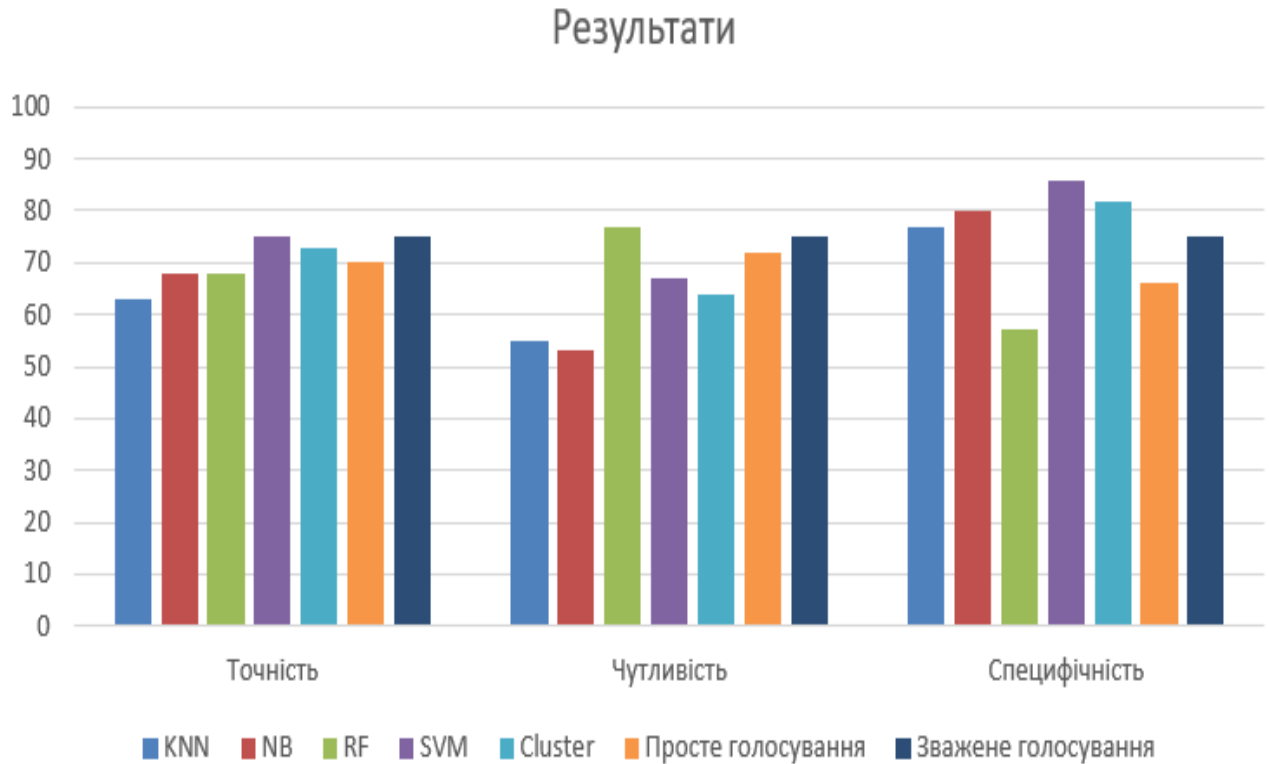


Рисунок 6.8 – Порівняння результатів моделей

На рисунку 6.8 зображено точність, чутливість та специфічність кожного методу. Як можна побачити у кожного з них є свої переваги і недоліки: метод найближчих сусідів має непогану специфічність, наївний байес вирізняється чутливістю, опорні вектори мають точність та специфічність, випадковий ліс непоганий у всіх категоріях, а кластерний аналіз відстає від інших. Просте голосування очікувано злегка відстає у точності від інших моделей, у той час як зважене голосування або знаходиться на одному рівні, або трохи переважає інші моделі.

Результати дослідження показали, що застосування екстракції особливостей для аналізу знімків очей збільшують швидкість та надійність діагностування. Окрім цього, отримані результати дозволяють зробити висновок про універсальність методу та можливості його ефективного застосування для

класифікації зображення.

ВИСНОВКИ

У ході виконання дипломної кваліфікаційної роботи магістра, проведено детальне ознайомлення із предметною областю – автоматизацією діагностування глаукоми. Також було розглянуто доступний інструментарій вирішення задачі і реалізовано декілька алгоритмів машинного навчання і їх ансамблів.

У першому розділі було проведено аналіз предметної області, включаючи її історію та існуючі аналоги проекту, обґрунтовано актуальність виконання роботи.

Другий розділ присвячено вибору середовища розробки програми, мови програмування Python та опису відповідних бібліотек, таких як numpy, openCV та tkinter, а також опис модульної структури проекту. Описано процес роботи програми.

У третьому розділі розглядались застосовані алгоритми та виділені особливості проекту.

У четвертому розділі описано керівництво програміста: визначено призначення програми та вимоги до системи, її характеристики, початкові і вхідні дані, можливі повідомлення програмісту.

У п'ятому розділі описано процес виконання програми, призначення програми, процес роботи та взаємодії з програмою.

У шостому розділі розібрано результати роботи програми. Порівняно особливості зображень залежно від діагнозу та оцінка використаних алгоритмів і ансамблів.

Результати тестування програмного забезпечення підтвердили ефективність обраних алгоритмів та методів, які дозволяють досягти покращення результатів оцінки зображень при роботі з тестовими даними. Функціонал створеного проекту можна буде розширити шляхом додавання автоматичного введення даних з файлів різних форматів та запис результатів до файлів, для зменшення часу обробки зображень та можливості вибору ваг алгоритмів

користувачем, що сприятиме тонкому налаштування ансамблю для збільшення точності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Quigley, H. A. The number of people with glaucoma worldwide in 2010 and 2020 [Text] / H. A. Quigley, A. T. Broman // British Journal of Ophthalmology. – 2006. – Vol. 90, № 3. – P. 262–267.
2. Global causes of blindness and distance vision impairment 1990–2020: A systematic review and meta-analysis [Text] / [S. R. Flaxman, R. R. A. Bourne, S. Resnikoff et al.] // The Lancet Global Health. – 2017. – Vol. 5, № 12. – P. 1221–1234.
3. Goldbaum, M. H. Comparing machine learning classifiers for diagnosing glaucoma from standard automated perimetry [Text] / M. H. Goldbaum, P.A. Sample, K. Chan et al. // Investigative Ophthalmology & Visual Science. – 2002. Vol. 43, № 1. – P. 162–169.
4. Computer-aided diagnosis of glaucoma using fundus images: A review [Text] / [Y. Hagiwara, J. E. W. Koh, J. H. Tan et al.] // Computer Methods and Programs in Biomedicine. – 2018. – Vol. 165. – P. 1–12.
5. Various glaucomatous optic using opencvappearances [Text] / [M. T. Nicolela et al.] // Ophthalmology. – 1996. – Vol. 103, № 4. – P. 640–649.
6. Chai, Y. Glaucoma diagnosis based on both hidden features and domain knowledge through deep learning models [Text] / Y. Chai, H. Liu, J. Xu // Knowledge-Based Systems. – 2018. – Vol. 161, № – P. 147–156.
7. Lee, E. J. Progressive retinal nerve fiber layer atrophy associated with enlarging peripapillary pit [Text] / E. J. Lee, T. W. Kim // Journal of Glaucoma. – 2017. – Vol. 26, № 2. – P. e79–e81.
8. Glaucoma detection from fundus image using OpenCV [Text] / [K. Narasimhan, K. Vijayarekha, K. A. JogiNarayana et al.] // Research Journal of Applied Sciences, Engineering and Technology. – 2012. – Vol. 4, № 24. – P. 5459–5463.
9. Septiarini, A. Optic disc and cup segmentation by automatic thresholding with morphological operation for glaucoma evaluation [Text] / A. Septiarini, A. Harjoko, R. Pulungan, R. Ekantini // Signal, Image and Video Processing. – 2017. – Vol. 11, № 5. – P. 945–952.

10. Detection of the optic nerve head in fundus images of the retina with Gabor filters and phase portrait analysis [Text] / [R. M. Rangayyan, X. Zhu, Ayres et al.]// Journal of Digital Imaging. – 2010. – Vol. 23, № 4. – P. 438–453
11. Khalid, N,E. Fuzzy c-means (FCM) for optic cup and disc segmentation with morphological operation [Text] / N. E. Khalid, N. M. Noor, N. Ariff // Procedia Computer Science. – 2014. – Vol. 42. – P. 255–262.
12. Virbukaitė, S. Deep learning methods for glaucoma identification using digital fundus images [Text] / S. Virbukaitė, J. Bernatavičienė // Baltic Journal of Modern Computing. – 2020. – Vol. 8, № 4. – P. 520–530.
13. Glaucoscan.ai [Electronic resource]. – Access mode : <https://glaucoscan.ai/>
14. iHealthScreen — glaucoma screening platform [Electronic resource]. – Access mode : <https://ihealthscreen.org/glucoma>
15. AI OCT optic disc analysis for glaucoma — Altris.ai [Electronic resource]. – Access mode : <https://www.altris.ai/article/ai-oct-optic-disc-analysis-for-glaucoma/>
16. Eclipse IDE: The leading open platform for professional developers [Electronic resource]. - Access mode: <https://help.eclipse.org/2024-03/index.jsp>
17. Visual Studio Code Documentation [Electronic resource]. - Access mode: <https://code.visualstudio.com/docs>
18. Visual Studio IDE [Electronic resource]. - Access mode: <https://learn.microsoft.com/uk-ua/visualstudio/ide/?view=vs-2022>
19. Python Language Reference [Electronic resource] / Python Software Foundation. – Access mode: <https://docs.python.org/3/reference/index.html>
20. Bradski, G. Learning OpenCV: Computer vision with the OpenCV library [Text]/ G. Bradski., A. Kaehler. // – Sebastopol : O'Reilly Media, 2008. – 6 p.
21. VanderPlas, J. Python Data Science Handbook: Essential Tools for Working with Data [Text]/ J. VanderPlas.// - Sebastopol : O'Reilly Media, 2016. - P. 33-96. - ISBN 978-1-4919-1205-8.
22. Septiarini, A. The contour extraction of cup in fundus images for

glaucoma detection [Text] / A. Septiarini, Hamdani, Khairina D. M. // International Journal of Electrical and Computer Engineering. – 2016. – Vol. 6, № 6. – P. 2797–2804.

23. Muramatsu, C. Diagnosis of glaucoma on retinal fundus images using deep learning: detection of nerve fiber layer defect and optic disc analysis [Text] / C. Muramatsu// Advances in Experimental Medicine and Biology. – 2020. – Vol. 1213. – P. 121–132.

24. Marin, D. Obtaining optic disc center and pixel region by automatic thresholding methods on morphologically processed fundus images [Text] / D. Marin, M. E. Gegundez-Arias, A. Suero, J. M. Bravo // Computer Methods and Programs in Biomedicine. – 2015. – Vol. 118, № 2. – P. 173–185.

25. Krishna, N. Understanding and visualization of different feature extraction processes in glaucoma detection [Text] / N. Krishna, K. Nagamani // J. Phys.: Conf. Ser. – 2022. – Vol. 2327. – P. 012023.

26. Balasubramanian, K. Automatic diagnosis and classification of glaucoma using hybrid features and k-nearest neighbor [Electronic resource] / K. Balasubramanian, N. P. Ananthamoorthy, K. Gayathridevi. – Access mode: https://www.researchgate.net/publication/329189930_Automatic_Diagnosis_and_Classification_of_Glaucoma_Using_Hybrid_Features_and_k_-Nearest_Neighbor/

27. Comparative analysis of k-nearest neighbor and modified k-nearest neighbor algorithm for data classification [Text] / [I. Okfalisa, Mustakim, N. G. I. Reza et al.] // Proc. 2nd International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE). – 2017. – P. 294–298.

28. Rebinth, A. Wavelet packet transform based image classification for computer aided glaucoma diagnosis using Naïve Bayes classifier [Electronic resource] / A. Rebinth, S. M. Kumar// – Access mode: https://www.researchgate.net/publication/340948990_Image_Processing_Using_Naive_Bayes_Classifier/

29. Amit, Y. Shape quantization and recognition with randomized trees [Text]/ Amit, Y., Geman, D. // Neural Computation. – 1997. – Vol. 9, № 7. – P. 1545–

1588.

30. Achtert, E. DeLi-Clu: Boosting Robustness, Completeness, Usability, and Efficiency of Hierarchical Clustering by a Closest Pair Ranking [Text] / E. Achtert, C. Böhm, P. Kröger // *Advances in Knowledge Discovery and Data Mining*. – 2006. – Vol. 3918. – P. 119–128.

31. On the ability of the optimal perceptron to generalise [Text] / [M. Opperr, W. Kinzel, J. Kleinz et al.] // *Journal of Physics A: Mathematical and General* – 2013. – Vol. 23, №11. – P. L581.

32. Boser, B. E.. A training algorithm for optimal margin classifiers. [Text] / B. E. Boser, I. M. Guyon, V. N. Vapnik // *Proceedings of the fifth annual workshop on Computational learning theory – COLT '92*. – 1992. – P. 144.

33. Opitz, D. Popular ensemble methods: An empirical study [Text] / D. Opitz, R. Maclin // *Journal of Artificial Intelligence Research*. – 1999. – Vol. 11. – P. 169–198.

34. Mienye, I. D. A survey of ensemble learning: Concepts, algorithms, applications and prospects [Text] / I. D. Mienye, Y. Sun // *IEEE Access*. – 2022. – Vol. 10. – P. 99129–99149.

35. A Bootstrap Framework for Aggregating within and between Feature Selection Methods [Text] / [R. Salman, A. Alzaatreh, H. Sulieman, et al.] // *Entropy*. – 2021. – Vol. 23, № 2. – P. 200.

ДОДАТОК А
Текст програми

Index.py

```
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk
import os
import cv2 as cv
import time
from matplotlib import pyplot as plt
from newk import cropImg
from fetureextraction import extract
from handleknn import handle_knnneg, handle_knnpos, handleKnnTest, handleknn
from handlenb import handlenbNeg, handlenbPos, handleNbTest, handlenbLearn
from handleRf import handleRfNeg, handleRfPos, handleRfTrain, handleRfTest
from handlesvm import handleSvmNeg, handleSvmPos, handleSvmTrain,
handleSvmTest
from handleC import handleClusterTest
from handleensemble import handleEnsembleSimple, handleEnsembleWeighted

class ImageAnalyzer:
    def __init__(self):
        self.window = tk.Tk()
        self.window.title("Image Analysis")
        self.window.minsize(800, 600)
        self.container = tk.Frame(self.window)
        self.container.pack(fill=tk.BOTH, expand=True)
        self.button_frame = tk.Frame(self.container)
        self.button_frame.pack(side=tk.TOP, fill=tk.X, padx=10, pady=10)
        self.image_frame = tk.Frame(self.container)
        self.image_frame.pack(side=tk.TOP, fill=tk.BOTH, expand=True, padx=10,
pady=10)
```

```

self.canvas = tk.Canvas(self.image_frame)
        self.scrollbar = tk.Scrollbar(self.image_frame, orient="vertical",
command=self.canvas.yview)
self.scrollable_frame = tk.Frame(self.canvas)

self.canvas.configure(yscrollcommand=self.scrollbar.set)
self.setup_scrolling()
self.create_buttons()
def testModel(self, processing_func):
    folderPath = filedialog.askdirectory()
    if not folderPath:
        return

    results = processing_func(folderPath)
    if results:
        self.displayResults(results)
#KNN
def handle_knnneg(self, event):
    handleknn("neg")

def handle_knnpos(self, event):
    handleknn("pos")

def handle_knnTest(self, event):
    folder_path = filedialog.askdirectory()
    if not folder_path:
        return

    start_time = time.time()
    results = handleKnnTest(folder_path)

```

```
end_time = time.time()
print("KNN time:", end_time - start_time)
if results:
    self.displayResults(results)

#Naive Bayes
def handle_nbNeg(self, event):
    handleNbNeg()
def handle_nbPos(self, event):
    handleNbPos()
def handle_nbTest(self, event):
    self.testModel(handleNbTest)

#Random Forest
def handle_rfNeg(self, event):
    handleRfNeg()
def handle_rfPos(self, event):
    handleRfPos()
def handle_rfTrain(self, event):
    handleRfTrain()
def handle_rfTest(self, event):
    self.testModel(handleRfTest)
#SVM
def handle_svmNeg(self, event):
    handleSvmNeg()
def handle_svmPos(self, event):
    handleSvmPos()
def handle_svmTrain(self, event):
    handleSvmTrain()
def handle_svmTest(self, event):
```

```

        self.testModel(handleSvmTest)

#K-means
def handle_cluster(self, event):
    self.testModel(handleClusterTest)
def handleEnsembleSimple(self, event):
    self.testModel(handleEnsembleSimple)
def handleEnsembleWeighted(self, event):
    self.testModel(handleEnsembleWeighted)

#GUI
def create_buttons(self):
    # KNN
    btn_knnNeg = tk.Button(self.button_frame, text="Підготовка KNN
негативного")
    btn_knnNeg.grid(row=0, column=0, padx=5, pady=5)
    btn_knnPos = tk.Button(self.button_frame, text="Підготовка KNN
позитивного")
    btn_knnPos.grid(row=0, column=1, padx=5, pady=5)
    btn_knnAna = tk.Button(self.button_frame, text="KNN аналіз")
    btn_knnAna.grid(row=0, column=2, padx=5, pady=5)

    #NB
    btn_nbNeg = tk.Button(self.button_frame, text="Підготовка NB негативного")
    btn_nbNeg.grid(row=1, column=0, padx=5, pady=5)
    btn_nbPos = tk.Button(self.button_frame, text="Підготовка NB позитивного")
    btn_nbPos.grid(row=1, column=1, padx=5, pady=5)
    btn_nbAna = tk.Button(self.button_frame, text="NB аналіз")
    btn_nbAna.grid(row=1, column=2, padx=5, pady=5)

    # RF

```

```

btn_rfNeg = tk.Button(self.button_frame, text="Підготовка RF негативного")
btn_rfNeg.grid(row=2, column=0, padx=5, pady=5)
btn_rfPos = tk.Button(self.button_frame, text="Підготовка RF позитивного")
btn_rfPos.grid(row=2, column=1, padx=5, pady=5)
btn_rfTrain = tk.Button(self.button_frame, text="Навчання RF моделі")
btn_rfTrain.grid(row=2, column=2, padx=5, pady=5)
btn_rfTest = tk.Button(self.button_frame, text="RF аналіз")
btn_rfTest.grid(row=2, column=3, padx=5, pady=5)

# SVM
        btn_svmNeg = tk.Button(self.button_frame, text="Підготовка SVM
негативного")
        btn_svmNeg.grid(row=3, column=0, padx=5, pady=5)
        btn_svmPos = tk.Button(self.button_frame, text="Підготовка SVM
позитивного")
        btn_svmPos.grid(row=3, column=1, padx=5, pady=5)
        btn_svmTrain = tk.Button(self.button_frame, text="Навчання SVM моделі")
        btn_svmTrain.grid(row=3, column=2, padx=5, pady=5)
        btn_svmTest = tk.Button(self.button_frame, text="SVM аналіз")
        btn_svmTest.grid(row=3, column=3, padx=5, pady=5)

# KL-means
btn_cluster = tk.Button(self.button_frame, text="Кластерний аналіз")
btn_cluster.grid(row=4, column=0, padx=5, pady=5)

# Прив'язки
btn_knnNeg.bind("<Button-1>", self.handle_knnneg)
btn_knnPos.bind("<Button-1>", self.handle_knnpos)
btn_knnAna.bind("<Button-1>", self.handle_knnTest)
btn_nbNeg.bind("<Button-1>", self.handle_nbNeg)

```

```

btn_nbPos.bind("<Button-1>", self.handle_nbPos)
btn_nbAna.bind("<Button-1>", self.handle_nbTest)
btn_rfNeg.bind("<Button-1>", self.handle_rfNeg)
btn_rfPos.bind("<Button-1>", self.handle_rfPos)
btn_rfTrain.bind("<Button-1>", self.handle_rfTrain)
btn_rfTest.bind("<Button-1>", self.handle_rfTest)
btn_svmNeg.bind("<Button-1>", self.handle_svmNeg)
btn_svmPos.bind("<Button-1>", self.handle_svmPos)
btn_svmTrain.bind("<Button-1>", self.handle_svmTrain)
btn_svmTest.bind("<Button-1>", self.handle_svmTest)
btn_cluster.bind("<Button-1>", self.handle_cluster)

# Ансамблі
    btn_ensSimple = tk.Button(self.button_frame, text="Ансамбль (просте
голосування)")
    btn_ensSimple.grid(row=5, column=0, padx=5, pady=5)
    btn_ensWeighted = tk.Button(self.button_frame, text="Ансамбль (зважене
голосування)")
    btn_ensWeighted.grid(row=5, column=1, padx=5, pady=5)
    btn_ensSimple.bind("<Button-1>", self.handleEnsembleSimple)
    btn_ensWeighted.bind("<Button-1>", self.handleEnsembleWeighted)

# Прокрутка
def setup_scrolling(self):
    self.scrollable_frame.bind(
        "<Configure>",
        lambda e: self.canvas.configure(scrollregion=self.canvas.bbox("all"))
    )
    self.canvas.create_window((0, 0), window=self.scrollable_frame, anchor="nw")
    self.canvas.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
    self.scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

```

```

self.canvas.bind_all("<MouseWheel>", self._on_mousewheel)

def _on_mousewheel(self, event):
    self.canvas.yview_scroll(int(-1 * (event.delta / 120)), "units")

def clear_images(self):
    for widget in self.scrollable_frame.winfo_children():
        widget.destroy()

def displayResults(self, results):
    if isinstance(results, tuple):
        initial_results, callback = results
        self.clear_images()
        for file_path, classification in initial_results.items():
            container = tk.Frame(self.scrollable_frame)
            container.pack(fill=tk.X, padx=5, pady=5)
            content_frame = tk.Frame(container)
            content_frame.pack(fill=tk.X, padx=5)

            img = cv.imread(file_path)
            img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
            img = cv.resize(img, (150, 150))
            img = Image.fromarray(img)
            img = ImageTk.PhotoImage(img)

            img_label = tk.Label(frame, image=img, bg=frame_color)
            img_label.image = img
            img_label.pack()

```

```

text_frame = tk.Frame(content_frame)
text_frame.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

file_name = os.path.basename(file_path)
tk.Label(text_frame, text=f"File: {file_name}", anchor="w",
justify=tk.LEFT).pack(fill=tk.X)
tk.Label(text_frame, text=f"Classification: {classification}",
fg=frame_color, anchor="w", justify=tk.LEFT).pack(fill=tk.X)
final_results = callback()
if final_results:
    self.displayResults(final_results)
else:
    self.clear_images()
for file_path, classification in results.items():
    container = tk.Frame(self.scrollable_frame)
    container.pack(fill=tk.X, padx=5, pady=5)
    content_frame = tk.Frame(container)
    content_frame.pack(fill=tk.X, padx=5)
    frame_color = "red" if "neg" in classification else "green"
    frame = tk.Frame(content_frame, bg=frame_color, bd=3)
    frame.pack(side=tk.LEFT, padx=(0, 10))

img = cv.imread(file_path)
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
img = cv.resize(img, (150, 150))
img = Image.fromarray(img)
img = ImageTk.PhotoImage(img)

img_label = tk.Label(frame, image=img, bg=frame_color)

```

```

img_label.image = img
img_label.pack()

text_frame = tk.Frame(content_frame)
text_frame.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

file_name = os.path.basename(file_path)
tk.Label(text_frame, text=f"File: {file_name}", anchor="w",
justify=tk.LEFT).pack(fill=tk.X)
tk.Label(text_frame, text=f"Classification: {classification}",
fg=frame_color, anchor="w", justify=tk.LEFT).pack(fill=tk.X)

def run(self):
    self.window.mainloop()

if __name__ == "__main__":
    app = ImageAnalyzer()
    app.run()

```

```

newk.py
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
def show_image(image, title="Image"):
    plt.imshow(cv.cvtColor(image, cv.COLOR_BGR2RGB))
    plt.title(title)
    plt.axis('off')
    plt.show()
def cropImg(file):

```

```

img = cv.imread(file, cv.IMREAD_COLOR)
if img is None:
    raise ValueError("Image not found or unable to load.")

#show_image(img, "Original Image")

lab = cv.cvtColor(img, cv.COLOR_BGR2LAB)
l_channel, a, b = cv.split(lab)
# show_image(l_channel)

clahe = cv.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
cl = clahe.apply(l_channel)
#show_image(cl)
limg = cv.merge((cl, a, b))
enhanced_img = cv.cvtColor(limg, cv.COLOR_LAB2BGR)
#show_image(enhanced_img, "Enhanced Image")

hsv = cv.cvtColor(enhanced_img, cv.COLOR_BGR2HSV)
h, s, v = cv.split(hsv)
#show_image(v)
max_val = np.max(v)
per = 0.65
_, th = cv.threshold(v, max_val * per, 255, cv.THRESH_BINARY)
#show_image(th, "Thresholded Image")

contours, hierarchy = cv.findContours(th, cv.RETR_EXTERNAL,
cv.CHAIN_APPROX_SIMPLE)

max_area = 0
largest_contour = None

```

```

for cnt in contours:
    area = cv.contourArea(cnt)
    if area > max_area:
        max_area = area
        largest_contour = cnt

contour_image = img.copy()
if largest_contour is not None:
    cv.drawContours(contour_image, [largest_contour], -1, (0, 255, 0), 2)
    #show_image(contour_image, "Contours on Image")

    x, y, w, h = cv.boundingRect(largest_contour)
    roi = img[y:y+h, x:x+w]
    #show_image(roi, "Region of Interest")

    return roi

else:
    print("No contours found.")
    return None

```

featureextraction.py

```

import cv2 as cv
import numpy as np
import math
from scipy.stats import skew
from scipy.stats import kurtosis
def extract(img):
    b,g,r=cv.split(img)

    e=0

```

```

count=0

for pixel in g:
    for pixel2 in pixel:
        e+=pixel2
        count+=1
u=e/count
e=0
for pixel in g:
    for pixel2 in pixel:
        e+=(math.pow(pixel2-u,2))
sigma=e/count

r=1-(1/(1+sigma))

row=[]
for pixel in g:
    for pixel2 in pixel:
        row.append(pixel2)
sk=(skew(row, axis=0, bias=True))
ks=(kurtosis(row, axis=0, bias=True))
std=math.sqrt(sigma)
u=round(u,4)
std=round(std,4)
r=round(r,4)
sk=round(sk,4)
ks=round(ks,4)
print(u,std,r,sk,ks)
return u,std,r,sk,ks
handleKnn.py

```

```

import json
import os
from tkinter import filedialog
from newk import cropImg
from matplotlib import pyplot as plt
from fetureextraction import extract
import cv2 as cv
import math
import json
import os
from tkinter import filedialog
from newk import cropImg
from fetureextraction import extract
import time
def handleknn(name):
    folderPath = filedialog.askdirectory()
    neg = []
    i = 0
    for filename in os.scandir(folderPath):
        i += 1
        if filename.is_file():
            img = cropImg(filename.path)
            ex = extract(img)
            neg.append(ex)
        if i == 50:
            break

    with open('Studies/knn.json') as f:
        data = json.load(f)

```

```

data[name] = neg
with open('Studies/knn.json', 'w') as file:
    json.dump(data, file, indent=4)

def handle_knnneg(event):
    handleknn("neg")
def handle_knnpos(event):
    handleknn("pos")
def handleKnnTest(folderPath=None, features=None, event=None):
    f = open('Studies/knn.json')

    data = json.load(f)
    negc=0
    posc=0
    stats=[]
    objs={}
    results = {}
    f.close()
    if features is None:
        if folderPath is None:
            folderPath = filedialog.askdirectory()
            if not folderPath:
                return None

    count=0
    for filename in os.scandir(folderPath):
        if filename.is_file():
            if (features is None):
                img=cropImg(filename.path)

```

```

    ex=extract(img)
else:
    ex=features[filename.path]
count+=1
#print(count)
stats.append(ex)
objs[filename]="
neg,pos=[],[]
for point in data["neg"]:
    dist=0
    for i in range(5):
        dist+=math.pow(ex[i]-point[i],2)
    dist=round(math.sqrt(dist),4)
    neg.append(dist)
for point in data["pos"]:
    dist=0
    for i in range(5):
        dist+=math.pow(ex[i]-point[i],2)
    dist=round(math.sqrt(dist),4)
    pos.append(dist)
countn,countp=0,0
pos.sort()
neg.sort()

for i in range(10):
    if(pos[0]>neg[0]):
        countn+=1
        neg.pop()
    else:
        countp+=1

```

```
pos.pop()
```

```
if(countn>countp):
    objs[filename]="negative"
    negc+=1
else:
    objs[filename]="positive"
    posc+=1
results[filename.path] = objs[filename]
```

```
for entry in objs:
    print(entry,objs[entry])
print(f"Negative: {negc}, Positive: {posc}")
return results
```

handleNb.py

```
import json
```

```
import os
```

```
from tkinter import filedialog
```

```
from newk import cropImg
```

```
from fetureextraction import extract
```

```
import math
```

```
import time
```

```
def handlenbNeg(event=None):
```

```
    handlenbLearn("neg")
```

```
def handlenbPos(event=None):
```

```
    handlenbLearn("pos")
```

```
def handlenbLearn(name):
```

```

folderPath = filedialog.askdirectory()
if not folderPath:
    return

neg = []
for i, filename in enumerate(os.scandir(folderPath)):
    if i >= 50:
        break
    if filename.is_file():
        img = cropImg(filename.path)
        ex = extract(img)
        neg.append(ex)

stats = []
for i in range(len(neg[0])):
    col = [entry[i] for entry in neg]
    mean = sum(col) / len(col)
    variance = math.sqrt(sum([(x - mean) ** 2 for x in col]) / (len(col) - 1))
    stats.append([mean, variance, len(col)])

try:
    with open('Studies/nb.json', 'r') as f:
        data = json.load(f)
except (FileNotFoundError, json.JSONDecodeError):
    data = {}

data[name] = stats

with open('Studies/nb.json', 'w') as f:
    json.dump(data, f, indent=4)

```

```

def gaussianProbability(x, mean, stdev):
    exponent = math.exp(-((x - mean) ** 2 / (2 * stdev ** 2)))
    return (1 / (math.sqrt(2 * math.pi) * stdev)) * exponent

def predictClass(stats1, stats2, row):
    total_rows = stats1[0][2] + stats2[0][2]
    probabilities = {
        "neg": stats1[0][2] / total_rows,
        "pos": stats2[0][2] / total_rows,
    }

    for i in range(len(stats1)):
        probabilities["neg"] *= gaussianProbability(row[i], stats1[i][0], stats1[i][1])
        probabilities["pos"] *= gaussianProbability(row[i], stats2[i][0], stats2[i][1])

    return probabilities

def handleNbTest(folderPath=None, features=None, event=None):
    start_time = time.time()
    if folderPath is None:
        folderPath = filedialog.askdirectory()
    if not folderPath:
        return None

    try:
        with open('Studies/nb.json', 'r') as f:
            data = json.load(f)
    except (FileNotFoundError, json.JSONDecodeError) as e:
        #print(f"Error loading model: {e}")

```

```

return

objs = {}
negc, posc = 0, 0
results = {}
count=0
for filename in os.scandir(folderPath):
    if filename.is_file():
        if (features is None):
            img = cropImg(filename.path)
            ex = extract(img)
        else:
            ex=features[filename.path]
        count+=1
        #print(count)
        probabilities = predictClass(data["neg"], data["pos"], ex)

        if probabilities["pos"] > probabilities["neg"]:
            objs[filename.path] = "positive"
            posc += 1
        else:
            objs[filename.path] = "negative"
            negc += 1

        results[filename.path] = objs[filename.path]

for entry in objs:
    print(entry, objs[entry])
print(f'Negative: {negc}, Positive: {posc}')

```

```
return results
```

```

handleRf.py
import json
import os
from tkinter import filedialog
from featureextraction import extract
from newk import cropImg
import cv2 as cv
from sklearn.ensemble import RandomForestClassifier
import numpy as np
import joblib
import time

MODEL_PATH = 'Studies/rf_model.pkl'

def handleRfNeg(event=None):
    handle_rfLearn("neg")
def handleRfPos(event=None):
    handle_rfLearn("pos")
def handle_rfLearn(name):
    folderPath = filedialog.askdirectory()
    if not folderPath:
        return

    samples = []
    for i, filename in enumerate(os.scandir(folderPath)):
        if i >= 50:

```

```

        break
    if filename.is_file():
        img = cropImg(filename.path)
        ex = extract(img)
        samples.append(ex)
os.makedirs("Studies", exist_ok=True)
try:
    with open("Studies/rf.json", "r") as f:
        data = json.load(f)
except (FileNotFoundError, json.JSONDecodeError):
    data = {}

data[name] = samples
with open("Studies/rf.json", "w") as f:
    json.dump(data, f, indent=4)
def handleRfTrain(event=None):
    try:
        with open("Studies/rf.json", "r") as f:
            data = json.load(f)
    except (FileNotFoundError, json.JSONDecodeError):
        return

    X = np.array(data["neg"] + data["pos"])
    y = np.array(["negative"] * len(data["neg"]) + ["positive"] * len(data["pos"]))
    model = RandomForestClassifier(n_estimators=100, random_state=42)
    model.fit(X, y)
    joblib.dump(model, MODEL_PATH)

def handleRfTest(folderPath=None, features=None, event=None):
    start_time = time.time()

```

```

if folderPath is None:
    folderPath = filedialog.askdirectory()
if not folderPath:
    return None

if not os.path.exists(MODEL_PATH):
    return

model = joblib.load(MODEL_PATH)
results = {}
negc, posc, count = 0, 0, 0
for filename in os.listdir(folderPath):
    if filename.is_file():

        if(features is None):
            img = cropImg(filename.path)
            ex = np.array(extract(img)).reshape(1, -1)
        else:
            ex = np.array(features[filename.path]).reshape(1, -1)
        count+=1
        #print(count)
        pred = model.predict(ex)[0]
        results[filename.path] = pred
        if pred == "negative":
            negc += 1
        else:
            posc += 1

print(f"Negative: {negc}, Positive: {posc}")
return results

```

```
handleSvm.py
import os
import json
import numpy as np
from sklearn.svm import SVC
from tkinter import filedialog
from featureextraction import extract
from newk import cropImg
import joblib
import time

MODEL_PATH = "Studies/svm_model.pkl"

def handleSvmNeg(event=None):
    handle_svmLearn("neg")

def handleSvmPos(event=None):
    handle_svmLearn("pos")

def handle_svmLearn(name):
    folderPath = filedialog.askdirectory()
    if not folderPath:
        return

    samples = []
    for i, filename in enumerate(os.scandir(folderPath)):
        if i >= 50:
            break
```

```

if filename.is_file():
    img = cropImg(filename.path)
    ex = extract(img)
    samples.append(ex)

try:
    with open("Studies/svm.json", "r") as f:
        data = json.load(f)
except (FileNotFoundError, json.JSONDecodeError):
    data = {}

data[name] = samples
with open("Studies/svm.json", "w") as f:
    json.dump(data, f, indent=4)
print(f"SVM {name} samples saved.")

def handleSvmTrain(event=None):
    try:
        with open("Studies/svm.json", "r") as f:
            data = json.load(f)
    except (FileNotFoundError, json.JSONDecodeError):
        return

    X = np.array(data["neg"] + data["pos"])
    y = np.array(["negative"] * len(data["neg"]) + ["positive"] * len(data["pos"]))

    model = SVC(kernel="rbf", probability=True)
    model.fit(X, y)
    joblib.dump(model, MODEL_PATH)

```

```

def handleSvmTest(folderPath=None, features=None, event=None):
    if folderPath is None:
        folderPath = filedialog.askdirectory()
    if not folderPath:
        return None

    if not os.path.exists(MODEL_PATH):
        return

    model = joblib.load(MODEL_PATH)
    results = {}
    negc, posc = 0, 0
    count=0
    for filename in os.scandir(folderPath):
        if filename.is_file():
            if (features is None):
                img = cropImg(filename.path)
                ex = np.array(extract(img)).reshape(1, -1)
            else:
                ex= np.array(features[filename.path]).reshape(1, -1)
            count+=1
            #print(count)
            pred = model.predict(ex)[0]
            results[filename.path] = pred
            if pred == "negative":
                negc += 1
            else:
                posc += 1

    print(f"Negative: {negc}, Positive: {posc}")

```

return results

import os

import numpy as np

from featureextraction import extract

from newk import cropImg

from tkinter import filedialog

import tkinter as tk

import time

def kmeansClustering(X, n_clusters, max_iters=100, random_state=42):

 np.random.seed(random_state)

 n_samples, n_features = X.shape

 centroids_indices = np.random.choice(n_samples, n_clusters, replace=False)

 centroids = X[centroids_indices]

 for _ in range(max_iters):

 distances = np.sqrt(((X - centroids[:, np.newaxis])**2).sum(axis=2))

 labels = np.argmin(distances, axis=0)

 new_centroids = np.array([X[labels == k].mean(axis=0) for k in
range(n_clusters)])

 if np.all(centroids == new_centroids):

 break

 centroids = new_centroids

return labels

def handleClusterTest(folderPath, features=None, event=None):

 if not folderPath:

 return

```

samples = []
paths = []
count = 0

for filename in os.scandir(folderPath):
    if filename.is_file():
        if features is None:
            img = cropImg(filename.path)
            ex = extract(img)
        else:
            ex = features[filename.path]
        count += 1
        print(count)
        samples.append(ex)
        paths.append(filename.path)
samples = np.array(samples)
labels = kmeansClustering(samples, n_clusters=2, random_state=42)
initial_results = {}
for path, label in zip(paths, labels):
    initial_results[path] = f"cluster_{label}"
return initial_results

```

handleensemble.py

```

from collections import Counter
import os
from tkinter import filedialog
from handleknn import handleKnnTest
from handlenb import handleNbTest
from handleRf import handleRfTest
from handlesvm import handleSvmTest

```

```

from handleC import handleClusterTest
from newk import cropImg
from fetureextraction import extract

def classifyClusters(results,preds):
    clusterCounts = {
        "cluster_0": {"negative": 0, "positive": 0},
        "cluster_1": {"negative": 0, "positive": 0},
    }
    for path, cluster_label in results.items():
        if cluster_label not in clusterCounts:
            continue
        pred = next((res[path] for res in preds if path in res), None)
        if pred is None:
            continue
        if pred == "negative":
            clusterCounts[cluster_label]["negative"] += 1
        else:
            clusterCounts[cluster_label]["positive"] += 1

    negCluster = "cluster_0" if (
        clusterCounts["cluster_0"]["negative"] /
max(sum(clusterCounts["cluster_0"].values()), 1) >=
        clusterCounts["cluster_1"]["negative"] /
max(sum(clusterCounts["cluster_1"].values()), 1)
    ) else "cluster_1"
    return {
        path: "negative" if cluster_label == negCluster else "positive"
        for path, cluster_label in results.items()
    }

```

```

def majority_vote(preds):
    positive_count = 0
    negative_count = 0
    for p in preds:
        if p == "positive":
            positive_count += 1
        else:
            negative_count += 1
    if positive_count >= negative_count:
        return "positive"
    else:
        return "negative"

```

```

def weightedVote(preds):
    weights = [0.23, 0.25, 0.25, 0.3, 0.27]
    positiveSc=0.0
    negativeSc=0.0
    for i in range(len(preds)):
        #print(preds[i])
        if preds[i] == "positive":
            positiveSc += weights[i]
        else:
            negativeSc += weights[i]
    ensNeg=0
    ensPos=0
    if positiveSc >= negativeSc:
        return "positive"
    else:
        return "negative"

```

```

def extractF (folderPath):
    features = {}
    #print(folderPath)
    count=0
    for file in os.scandir(folderPath):
        if file.is_file():
            count+=1
            #print(count)
            img = cropImg(file.path)
            ex = extract(img)
            features[file.path] = ex
    return features

def handleEnsembleSimple(folderPath):
    knn_results = handleKnnTest(folderPath)
    nb_results = handleNbTest(folderPath)
    rf_results = handleRfTest(folderPath)
    svm_results = handleSvmTest(folderPath)
    c_results=handleClusterTest(folderPath)
    #print(knn_results)
    #print(nb_results)
    #print(rf_results)
    #print(svm_results)
    #print(c_results)
    kmean_results = classifyClusters(
        c_results,
        [knn_results, nb_results, rf_results, svm_results]
    )
    ensembleResults = {}
    ensPos=0
    ensNeg=0

```

```

for f in knn_results.keys():
    preds = [
        knn_results.get(f),
        nb_results.get(f),
        rf_results.get(f),
        svm_results.get(f),
        kmean_results.get(f)
    ]
    ensembleResults[f] = majority_vote(preds)
    if(ensembleResults[f]=="positive":
        ensPos+=1
    else:
        ensNeg+=1
print(f"Negatives: {ensNeg}, Positives: {ensPos}")
return ensembleResults

```

```

def handleEnsembleWeighted(folderPath):
    features=extractF(folderPath)
    knn_results = handleKnnTest(folderPath,features)
    nb_results = handleNbTest(folderPath,features)
    rf_results = handleRfTest(folderPath,features)
    svm_results = handleSvmTest(folderPath,features)
    c_results=handleClusterTest(folderPath,features)

    kmean_results = classifyClusters(
        c_results,
        [knn_results, nb_results, rf_results, svm_results]
    )
    ensemble_results = {}
    ensPos=0

```

```
ensNeg=0
for f in knn_results.keys():
    preds = [
        knn_results.get(f),
        nb_results.get(f),
        rf_results.get(f),
        svm_results.get(f),
        kmean_results.get(f)
    ]

    ensemble_results[f] = weightedVote(preds)
    if(ensemble_results[f]=="positive":
        ensPos+=1
    else:
        ensNeg+=1
print(f"Negative: {ensNeg}, Positive: {ensPos}")
return ensemble_results
```

ДОДАТОК Б
Слайди презентації

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ДІАГНОСТИКИ ГЛАУКОМИ

Виконав: Ст. групи КНТ

І.І.М. Петунін Олександр

Керівник: К.т.н., Професор ну "Запорізька політехніка "

Табунщик Галина

Рисунок Б.1 – Слайд 1



Рисунок Б.2 – Слайд 2

3

Мета і задачі проєкту

Мета роботи:
підвищення точності
комп'ютерної
діагностики шляхом
застосування та
порівняння алгоритмів та
ансамблів алгоритмів
машинного навчання.



Задачі проєкту:

- Забезпечити можливість введення даних для діагностики;
- Розробити алгоритми видалення зони інтересу та виділення особливостей зображення;
- Розробити алгоритми вирішення класифікації;
- Вивести результати у графічному форматі;
- Виведення попереджень при помилках

Рисунок Б.3 – Слайд 3

4

Огляд особливостей середовища розробки

Критерій порівняння	Середовища розробки		
	Visual studio code	Visual Studio	Eclipse
Розмір та функціональність	Важкий, багатфункціональний, призначений для серйозної розробки	Простий та швидкий	Простий, швидкий
Підтримувані мови	.NET (C#, VB.NET, F#), C++, Python, JavaScript	Js, Python, PHP, Java, C/C++/C#	Java, C/C++, PHP, Python, JavaScript тощо
Інтерфейс користувача	Комплексний і багатфункціональний	Простий та зрозумілий	Простий та зрозумілий
Вбудовані інструменти	налагодження, тестування, профілювання	Базове налагодження, рефакторинг, тестування	Розширене налагодження, тестування, профілювання
Інтеграція з системами контролю версій	Вбудовані Git, GitHub, Azure DevOps	Вбудований Git, розширюваний іншими системами	Вбудовані Git, SVN, CVS та інші системи
Спільнота та підтримка	Сильна, офіційна та спільнотна підтримка	Сильна, офіційна та спільнотна підтримка	Сильна, офіційна та <u>спільнотна</u> підтримка

Рисунок Б.4 – Слайд 4



Рисунок Б.5 – Слайд 5

6

Видалення Зони інтересу

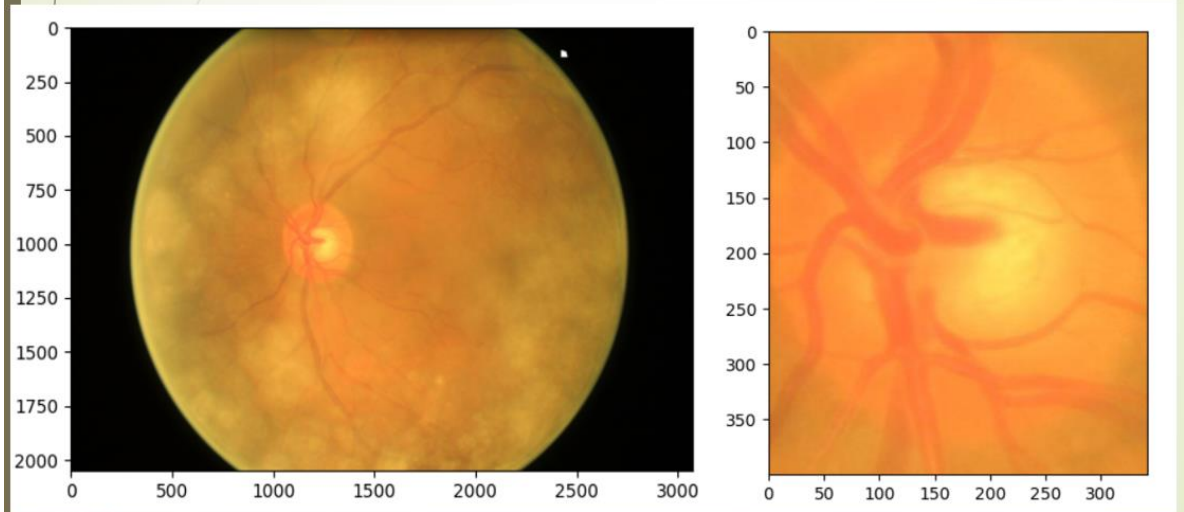


Рисунок Б.6 – Слайд 6

7

Особливості зображень

Середнє

$$\bar{X} = \frac{\sum X}{N}$$

Стандартне відхилення

$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$$

Середньоквадратичне

$$R = 1 - \frac{1}{(1 + \sigma^2)}$$

Гостровершинність

$$kurtosis = \frac{(x - \mu)^4}{\sigma^4}$$

Асиметрія

$$sk_1 = \frac{x - Mode}{s}$$

Рисунок Б.7 – Слайд 7

8

Методи машинного навчання

- Алгоритми машинного навчання – це обчислювальні процедури, які дозволяють системі навчатися на основі даних, виявляти закономірності та робити прогнози чи рішення без явного програмування. Вони складають основу систем штучного інтелекту та класифікуються на основі їхнього підходу до навчання.

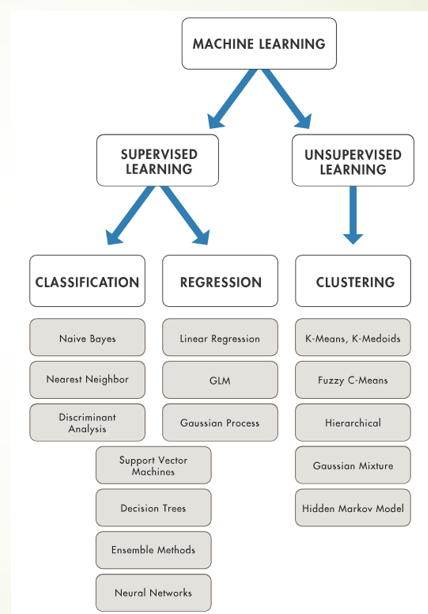


Рисунок Б.8 – Слайд 8

9

Ансамблі машинного навчання

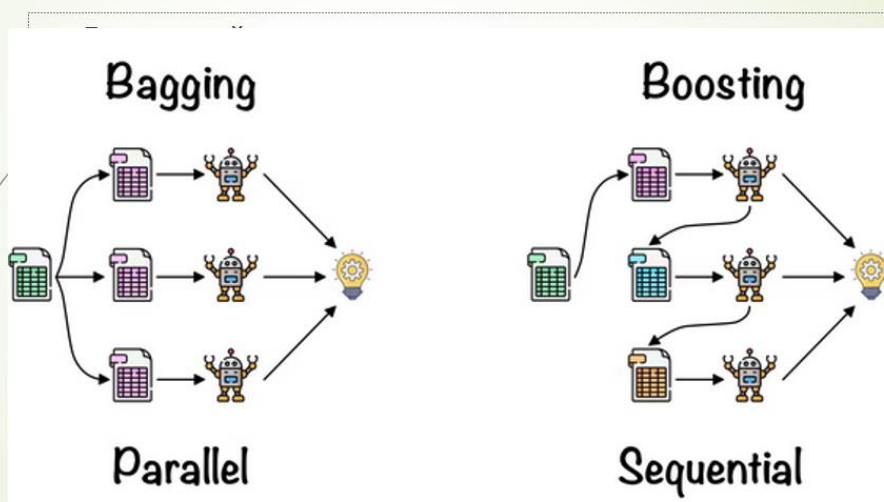


Рисунок Б.9 – Слайд 9

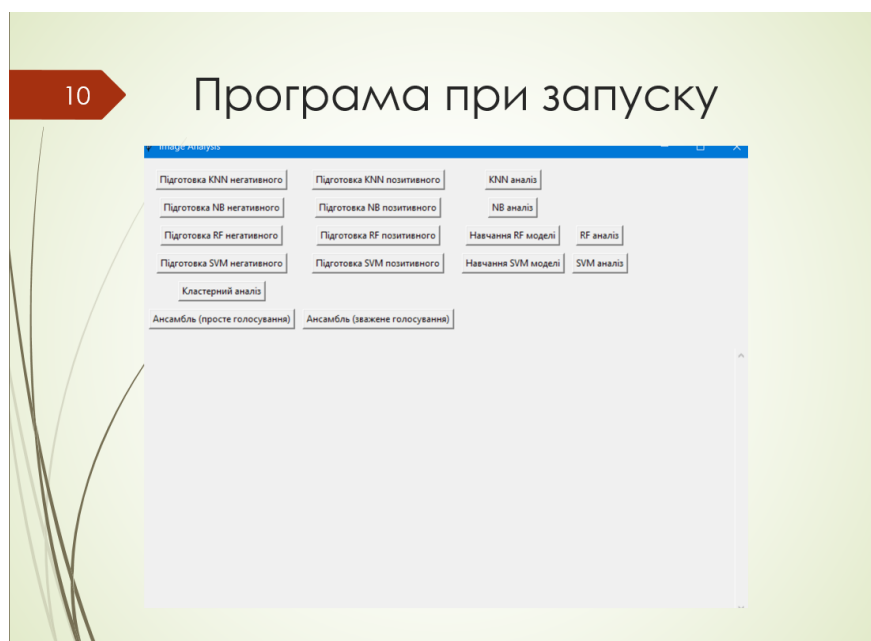


Рисунок Б.10 – Слайд 10

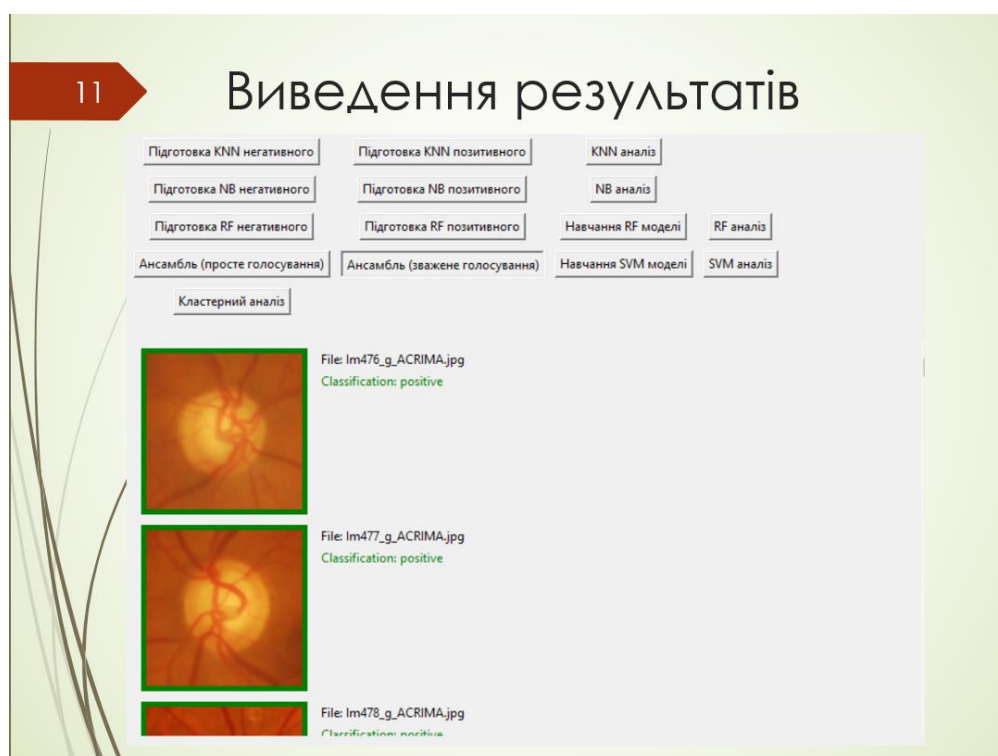


Рисунок Б.11 – Слайд 11

12

Порівняння методів

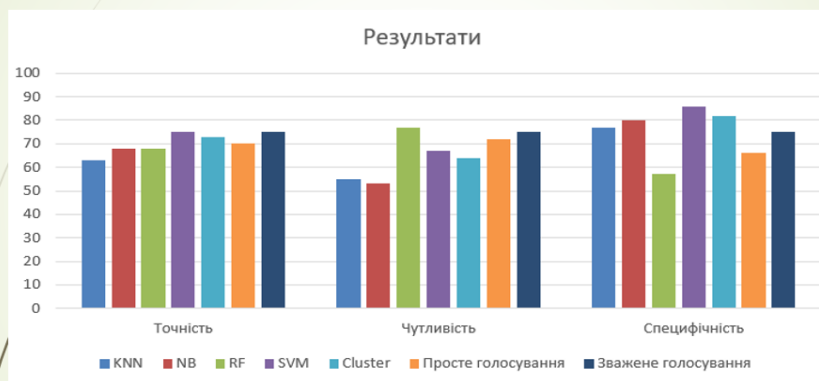


Рисунок Б.12 – Слайд 12

13

Проблеми розроблюваної системи та рішення

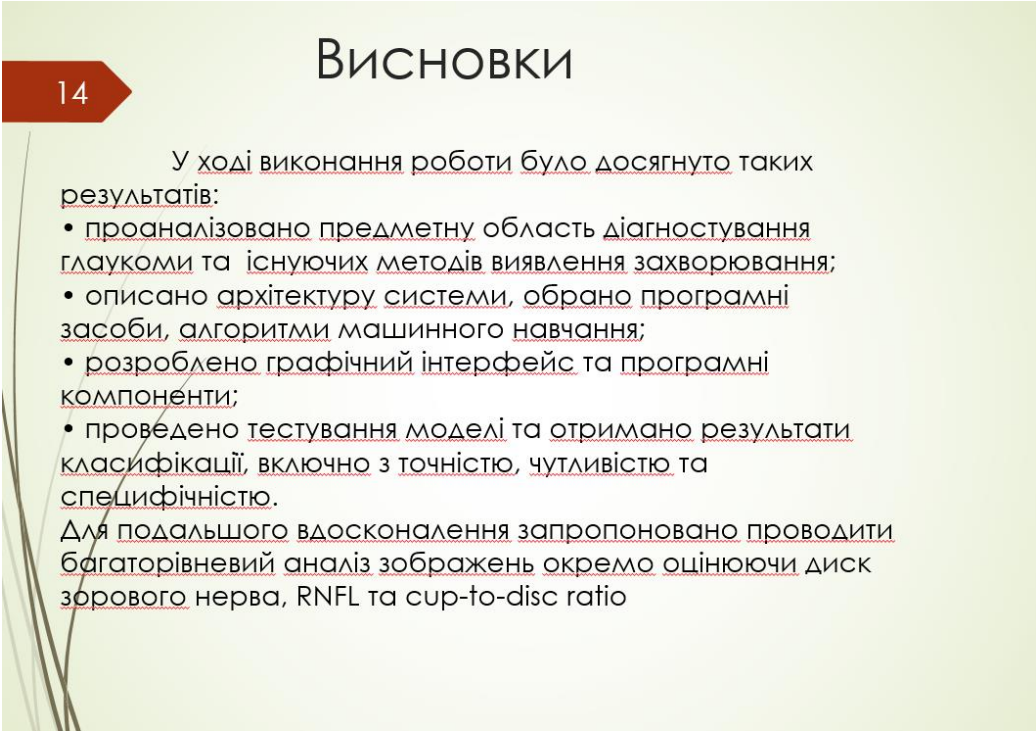
Проблеми:

- недостатня точність окремих алгоритмів машинного навчання, особливо на невеликих наборах даних;
- ризик хибних позитивних та хибних негативних результатів, що впливає на надійність діагностики;
- висока чутливість моделей до шумів, освітлення та артефактів на зображенні;
- неоднорідність форматів і розмірів зображень, що вимагає додаткової нормалізації.

Рішення:

- розширити навчальний датасет за рахунок аугментації зображень (дзеркалення, повороти, зміна контрастності);
- застосовувати багатокрокове виділення ознак, включаючи гістограмні ознаки, текстурні та кольорові характеристики;
- запровадити механізм автоматичного виявлення пошкоджених або неякісних зображень, щоб запобігти неправильним прогнозам.

Рисунок Б.13 – Слайд 13



Висновки

У ході виконання роботи було досягнуто таких результатів:

- проаналізовано предметну область діагностування глаукоми та існуючих методів виявлення захворювання;
- описано архітектуру системи, обрано програмні засоби, алгоритми машинного навчання;
- розроблено графічний інтерфейс та програмні компоненти;
- проведено тестування моделі та отримано результати класифікації, включно з точністю, чутливістю та специфічністю.

Для подальшого вдосконалення запропоновано проводити багаторівневий аналіз зображень окремо оцінюючи диск зорового нерва, RNFL та cup-to-disc ratio

Рисунок Б.14 – Слайд 14