МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА
Факультет економіки і підприємництва

Кафедра інформаційних технологій

Допущено до захисту
"_____" _____ 2023 р.
Зав. кафедри, к.т.н., доцент
_____ Ліщук Р.І.

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Програмний модуль діагностування пацієнтів з використанням нейронної мережі Кохонена

Виконав – студент 2 курсу, групи 21м-кн
освітньої програми "Комп'ютерні науки"
спеціальності 122 Комп'ютерні науки
Пенько Максим Віталійович
Науковий керівник – д.т.н., професор
Кучерук Володимир Юрійович

Умань – 2023

Реферат

на кваліфікаційну роботу «Програмний модуль діагностування пацієнтів з використанням нейронної мережі Кохонена»
студента факультету економіки та підприємництва
Уманського національного університету садівництва
Пенька Максима Віталійовича

Структура та обсяг роботи - пояснювальна записка до кваліфікаційної роботи магістра складається з 3 розділів, містить 83 сторінки, 20 рисунків, 6 додатків, 46 літературних джерел.

У кваліфікаційній роботі магістра розглянуто процес попереднього медичного діагностування пацієнтів за аналізом крові та артеріального тиску, розроблено програмний модуль попереднього діагностування пацієнтів на основі нейронної мережі Кохонена.

Проведено аналіз предметної області для створення програмного модуля. У ході аналізу предметної області були проаналізовані сучасні процеси та методи діагностування пацієнтів, запропонований метод медичного діагностування, що використовує значення показників загального аналізу крові та артеріального тиску для визначення групи захворювань, до яких відноситься хвороба пацієнта.

Проведено аналіз процесу кластеризації, також обґрунтовано вибір методу кластеризації. Розроблено структуру, математичну модель, порядок функціонування нейронної мережі Кохонена та алгоритм функціонування програмного модуля попереднього діагностування.

Обґрунтовано вибір мови та середовища програмування для розробки програмного модуля та створено діаграму класів для його функціонуваннння, реалізовані всі основні модулі програми. Протестовано можливості програмного модуля попереднього діагностування пацієнтів на основі нейронної мережі Кохонена, а саме: завантаження даних та навчання мережі на їх основі, проведення тестового діагностування з вводом даних та отримання результатів, щодо груп хвороб тестового пацієнта. Всі функції працюють коректно. Також проведено тестування розробленого програмного модуля та

програми-аналога YourDiagnosis. Розроблений програмний модуль має вищу достовірність попереднього діагностування пацієнтів (92%), ніж аналогічна програма (84%), а значить достовірність попереднього діагностування пацієнтів покращена на 8%.

Програмний модуль попереднього діагностування пацієнтів на основі нейронної мережі Кохонена повністю придатний до використання і може використовуватися у медичній сфері з метою попередньої діагностики пацієнтів та економії часу лікарів і пацієнтів. Програмний модуль може бути вдосконалений при потребі.

Анотація

У кваліфікаційній роботі магістра на основі проведеного аналізу пропонується підхід до медичного діагностування пацієнтів, що заснований на використанні показників загального аналізу крові та артеріального тиску. Даний підхід реалізується за допомогою кластеризації даних нейронною мережею Кохонена. Програмна реалізація запропонованого підходу виконується з використанням мови програмування C# та підвищує достовірність визначення діагнозу пацієнтів. Розроблений програмний модуль має вищу достовірність попереднього діагностування пацієнтів (92%), ніж аналогічна програма (84%), тобто достовірність попереднього діагностування пацієнтів покращена на 8%.

Проведено тестування програмного модуля попереднього діагностування пацієнтів.

Ключові слова: попереднє діагностування, загальний аналіз крові.

Annotation

In the master's qualification work, on the basis of the analysis, an approach to medical diagnosis of patients based on the use of indicators of complete blood count and blood pressure is proposed. This approach is implemented using data clustering by Kohonen neural network. The software implementation of the proposed approach is carried out using the C# programming language and increases the reliability of the diagnosis of patients. The developed software module has a higher reliability of preliminary diagnosis of patients (92%) than a similar program (84%), that is, the reliability of preliminary diagnosis of patients is improved by 8%.

Testing of the software module for preliminary diagnosis of patients was carried out.

Keywords: preliminary diagnosis, general blood test.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА

Факультет економіки і підприємництва
Кафедра _____Інформаційних технологій_____
Освітня програма _____Комп'ютерні науки_____
Спеціальність_____122 Комп'ютерні науки_____
(код, назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ ___Ліщук Р.І._____
(підпис) (ініціали, прізвище)
"_____" _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студенту_____*Пеньку Максиму Віталійовичу*_____
(прізвище, ім'я, по батькові)
1. Тема проекту *Інформаційна технологія підвищення стійкості текстових паролів*_____
керівник роботи__*Кучерук Володимир Юрійович, д.т.н., професор кафедри інформаційних технологій*_____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від "12"_червня_2023 року №_01-06/130
2. Строк подання студентом роботи_____8 пункт, 28.05.2023_____
3. Вихідні дані до роботи _*дані, на основі яких робиться діагноз – не менше 15 параметрів аналізу крові; кількість діагностованих груп хвороб – не менше 12; обсяг навчальної вибірки – не менше 800; обсяг текстової вибірки – не менше 100, середовище програмування – об'єктно-орієнтоване.*
4. Перелік питань, які потрібно розробити
_*Аналітичний огляд предметної області, алгоритмічне та програмне забезпечення засобу, тестування застосунку*
5. Перелік графічного матеріалу__2 таблиці, 20 рисунків, 6 додатків, 46 джерел
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ___07.09.2022_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка
1.	Вибір теми кваліфікаційної роботи	05.09.2022	
2.	Одержання індивідуального завдання	07.09.2022	
3.	Складання календарного плану написання кваліфікаційної роботи	08.09.2022	
4.	Підготовка до виконання кваліфікаційної роботи - підбір та вивчення літератури - участь у виконанні науково-дослідних робіт - виконання курсових робіт - інші заходи	Впродовж I семестру	
5.	Уточнення теми кваліфікаційної роботи та календарного плану-графіка, виходячи зі специфіки базового підприємства, установи	До початку II семестру	
6.	Підготовка: - першого розділу - другого розділу - третього розділу - висновків	 09.02.2023 23.03.2023 26.04.2023 10.05.2023	
7.	Доопрацювання кваліфікаційної роботи з урахуванням зауважень наукового керівника	18.05.2023	
8.	Написання та оформлення роботи в остаточному варіанті	25.05.2023	
9.	Перевірка роботи на академічний плагіат	12.06.2023	
10.	Одержання відгуку наукового керівника	13.06.2023	
11.	Попередній захист роботи на кафедрі	14.06.2023	
12.	Затвердження роботи завідувачем кафедри	15.06.2023	
13.	Одержання рецензії	16.06.2023	
14.	Захист кваліфікаційної роботи	19-21.06.2023	

Студент_____ ___**Пенько М.В.**_____
(підпис) (прізвище та ініціали)

Керівник роботи _____ ___**Кучерук В.Ю.**_____
(підпис) (прізвище та ініціали)

ЗМІСТ

Перелік умовних позначень, скорочень і термінів …… 8
ВСТУП 9
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПОПЕРЕДНЬОГО ДІАГНОСТУВАННЯ ПАЦІЄНТІВ 11
1.1. Процес медичного діагностування пацієнтів за аналізом крові ……..... 11
1.2. Обґрунтування методу розв’язання задачі 20
1.3. Постановка задачі..... 32
РОЗДІЛ 2. ПРОЕКТУВАННЯ ОСНОВНОГО КОМПОНЕНТА ПРОГРАМИ ПОПЕРЕДНЬОГО ДІАГНОСТУВАННЯ ПАЦІЄНТІВ 34
2.1. Аналіз процесу кластеризації 34
2.2. Обґрунтування вибору методу кластеризації..... 35
2.3. Структура, математична модель та порядок функціонування нейронної мережі Кохонена 38
РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ ПОПЕРЕДНЬОГО ДІАГНОСТУВАННЯ ПАЦІЄНТІВ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ КОХОНЕНА 49
3.1. Обґрунтування вибору мови та середовища програмування 49
3.2. Реалізація основних компонентів програмного модуля попереднього діагностування пацієнтів на основі нейронної мережі Кохонена 50
3.3. Тестування програмного модуля попереднього 54
ВИСНОВКИ 59
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 61
ДОДАТОК А. Загальний алгоритм роботи програмного модуля попереднього діагностування пацієнтів на основі нейронної мережі Кохонена..... 65
ДОДАТОК Б. Структура нейронної мережі Кохонена 66
ДОДАТОК В. Діаграма класів програмного модуля 67
ДОДАТОК Г. Початкове вікно програмного модуля..... 68
ДОДАТОК Д. Результати роботи програми 69
ДОДАТОК Е. Лістинг програми 70

Перелік умовних позначень, скорочень і термінів

ШОЕ	Швидкість осідання еритроцитів
MCV	Mean Corpuscular Volume
НПЗЗ	Нестероїдні протизапальні препарати
ВІЛ	Ві́рус імунодефіци́ту люди́ни
СНІД	Синдро́м набу́того імунодефіци́ту
СДУГ	синдром дефіциту уваги та гіперактивності
АТ	Артеріальний тиск
ХНН	Хронічна хвороба нирок
COVID	COronaVIrus Disease
RDW	Ширина розподілу еритроцитів
RBC	Еритроцити
LVQ	Learning Vector Quantization

ВСТУП

Актуальність теми дослідження полягає у відсутності точних, загальнодоступних та швидких програмних засобів діагностування є причиною невірного визначення хвороб пацієнтів у лікарнях, витрат часу на очікування у чергах, реєстрацію та проведення непотрібних аналізів. У даній ситуації бажаним є створення програмного засобу для діагностування пацієнтів, що зможе забезпечити високу достовірність та швидкодію діагностування різноманітних хвороб, який не потребує спеціального апаратного забезпечення для свого функціонування.

Об'єктом дослідження є процес автоматизованого попереднього діагностування пацієнтів за аналізом крові.

Предметом дослідження є методи та програмні засоби попереднього діагностування пацієнтів за аналізом крові на основі нейронних мереж.

Метою роботи є підвищення достовірності попереднього медичного діагностування пацієнтів шляхом створення програмного модуля кластеризації пацієнтів за аналізом крові та показником артеріального тиску на основі нейронної мережі Кохонена.

Для досягнення цієї мети необхідно вирішити такі **задачі:**

- проаналізувати відомі методи попереднього діагностування пацієнтів та обрати напрямок досліджень;

- обґрунтувати вибір типу нейромережі;

- проаналізувати модель попереднього діагностування пацієнтів на основі нейронної мережі Кохонена;

- розробити структуру програмного модуля;

- розробити алгоритм роботи програмного модуля попереднього діагностування пацієнтів;

- здійснити програмну реалізацію модуля попереднього діагностування пацієнтів;

- провести тестування програмного модуля попереднього діагностування пацієнтів.

Для досягнення поставленої в роботі мети використовуються такі **методи**

дослідження:

- системний аналіз;
- об'єктно-орієнтовані методи проектування;
- методи програмного моделювання.

Наукова новизна полягає в можливості пацієнта проходити попереднє діагностування за загальним аналізом крові та показником артеріального тиску швидше та більш достовірно, ніж зараз це представлено на ринку та виявити основну групу хвороб, якщо така є.

Практичне значення роботи полягає в розробці алгоритму та програмного модуля, який відрізняється покращеним результатом роботи, а саме матиме більшу достовірність ніж аналогічний програмний модуль, завдяки чому пацієнти та лікарі матимуть змогу зекономити час, який може витрачатися на додаткові обстеження, через хибний попередній діагноз.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПОПЕРЕДНЬОГО ДІАГНОСТУВАННЯ ПАЦІЄНТІВ

1.1. Процес медичного діагностування пацієнтів за аналізом крові

У сучасному світі глобальний процес автоматизації сягнув практично усіх сфер людської діяльності, включаючи медицину. Напрямками її комп'ютеризації та подальшого удосконалення є:

- створення баз даних пацієнтів лікарень;
- системи реєстрації пацієнтів;
- системи діагностування;
- системи телемедицини (використовують телекомунікаційні й електронні інформаційні технології для надання медичної допомоги й послуг у сфері охорони здоров'я в необхідному місці) [2, 3].

Традиційний процес діагностування хворих у лікарнях є доволі тривалим та багатоетапним - пацієнт має пройти реєстрацію, консультацію у свого сімейного лікаря, отримати направлення до лікаря-спеціаліста, який з деякою ймовірністю може визначитись хибно стосовно реальної хвороби пацієнта. Щоб підтвердити або спростувати діагноз потрібно пройти низку аналізів, що проводиться через довготривалість цього процесу до загострення хвороби або встановлення невірного діагнозу.

З огляду на те, що потенційними користувачами даної системи будуть працівники медичних закладів та їх пацієнти, а їх потребами буде якісне медичне обслуговування, то в сучасному світі це забезпечується комп'ютеризацією галузі. Отже, на сьогодні, коли більшість людей має мобільні девайси, раціональним є використання їх можливостей та широке розповсюдження мобільних інформаційних застосунків серед населення, тобто поліпшення загального процесу діагностування через застосування телемедицини. Це дозволить пацієнтам отримати діагноз без відвідування лікарні.

Наявні автоматизовані способи діагностики також є недостатньо ефективними - більшість систем орієнтовані на визначення конкретної хвороби,

що через специфіку медичної галузі, має високу ймовірність виявитись недостовірним; мають невиправдану складність алгоритмів та їх програмної реалізації; можуть використовуватись лише медичними фахівцями [4].

Таким чином, існує проблема недосконалості актуальних методів та засобів діагностування, які не забезпечують високу точність виявлення захворювання, швидкість (оперативність) та зручність використання із залученням персональних мобільних засобів (телефони, смартфони, планшети та ін.), не можуть використовуватись пацієнтами дистанційно без необхідності відвідування лікарень.

Багато пацієнтів сприймають періодичні аналізи крові, як частину про-активного та відповідального підходу до підтримки свого загального здоров'я. Думка лікаря про те, що аналізи крові слід проводити як допоміжні тести для виявлення симптомів, суперечить їхнім поглядам і здається їм недбалим. За останні кілька років у відносинах між лікарем і пацієнтом відбулися поступові зміни, які стали менш патерналістськими та більш орієнтованими на пацієнта. Багато пацієнтів звертаються до лікаря з певною метою отримання направлення на дослідження крові [4].

З метою покращення процесу діагностування пацієнтів доцільно створити програмний засіб, що відрізняється сумісністю з довільним програмним, апаратним забезпеченням та дружністю і інтуїтивністю інтерфейсу, який буде використовувати інтелектуальний алгоритм визначення діагнозу пацієнта, забезпечити його швидкодію та достовірність.

Для такого автономного діагностування доцільно застосувати підхід, заснований на використанні показників загального аналізу крові, що проводиться при більшості захворювань і профілактичних обстеженнях, і включає в себе 15 показників, кожен з яких при відхиленні від визначених нормальних значень даних показників здорової людини передбачає схильність до певних хвороб. Діагностування за допомогою обробки отриманих показників аналізу крові допоможе виявити тип захворювання пацієнта та надати рекомендації щодо лікування [5].

Загальний аналіз крові - це лабораторне дослідження, що включає

підрахунок всіх видів клітин крові (лейкоцитів, еритроцитів, тромбоцитів), визначення їх параметрів, лейкоцитарної формули, вимірювання рівня гемоглобіну, визначення співвідношення клітинної маси до плазми (гематокрит). Загальний аналіз крові проводиться при більшості захворювань і профілактичних обстеженнях і включає в себе 15 показників, кожен з яких при відхиленні від норми є індикатором про наявність певних хвороб [6].

Загальний аналіз крові - один із найважливіших діагностичних методів, що відображає реакцію кровотворних органів на вплив різноманітних фізіологічних і патологічних чинників.

Клінічне значення загального аналізу крові:

- дає змогу оцінити функціональний стан організму (реактивність організму за відповіддю лейкоцитарної формули на інфекцію та інші патологічні процеси; стан еритропоезу за кількістю ретикулоцитів при крововтратах);

- допомагає встановити діагноз (захворювання крові, запальний процес, гнійно-септичний стан, специфічний імунний процес);

- дає можливість провести диференціальну діагностику низки патологічних станів (наприклад, стенокардії та інфаркту міокарда);

- допомагає в оцінці тяжкості перебігу та активності гострого процесу, визначенні загострення хронічного захворювання, а також розвитку ускладнень;

- дає змогу контролювати ефективність проведеної терапії (захворювання системи крові, запальні та інфекційні процеси);

- дає змогу прогнозувати перебіг патологічного процесу (на підставі реактивності та опору організму за даними лейкограми).

Загальний аналіз крові включає виявлення кількісного та якісного складу формених елементів крові: визначення концентрації гемоглобіну, кількості еритроцитів, ретикулоцитів, обчислення колірного показника, визначення кількості лейкоцитів і підрахунок лейкоцитарної формули, визначення кількості тромбоцитів, визначення швидкості осідання еритроцитів (ШОЕ), опис можливих морфологічних змін периферичної крові.

Кожен формений елемент має своє значення для організму і його кількісні зміни відповідають за певні процеси в організмі.

Еритроцити - це невеликі еластичні клітини, що містять у своїй цитоплазмі гемоглобін. За рахунок еластичності вони з легкістю проходять через судини будь-якого калібру. Вони виробляються в кістковому мозку, життєздатність однієї клітини близько від 3 до 4 місяців.

Еритроцити виконують таку функцію: вони несуть кисень із легень до всіх тканин і органів людини, і зворотним шляхом від тканин до легень приносять вуглекислий газ. Усе це відбувається шляхом приєднання газів до гемоглобіну еритроцита.

Підвищення еритроцитів у загальному аналізі крові можливе в разі зневоднення організму внаслідок блювоти та діареї, хвороб системи крові (еритремія, хвороба Вакеза), серцевої та дихальної недостатності.

Їх зниження може бути за крововтрат, лейкозів і лімфом, вроджених дефектів кровотворення, гемолітичної анемії, онкології, недостатнього вживання білка, заліза та вітамінів.

Слід пам'ятати, що норма еритроцитів, як і інших показників, може відрізнятися в різних лабораторіях. У яких, до того ж, не виключені похибки. Тому не завжди прикордонний результат свідчить про серйозну хворобу.

Гемоглобін - кров'яний пігмент, який являє собою складний білок, що складається з гема і білка глобіну, і міститься в еритроцитах.

Основна функція гемоглобіну - перенесення кисню з легень у тканини, а також виведення вуглекислого газу з організму та регуляція кислотно-основного стану.

Олігохромемія, або гемоглобінопенія (зниження концентрації гемоглобіну в крові) спостерігається за анемій, гострих і "прихованих" кровотеч, злоякісних пухлин та їхніх метастазів, уражень кісткового мозку та нирок, а також у випадках застосування деяких лікарських препаратів, які можуть спричинити розвиток апластичної анемії (протипухлинні, протисудомні, важкі метали, деякі антибіотики, анальгетики) або сприяти розвитку гемолізу (пеніцилін, левоміцетин, сульфаніламіди).

Гіперхромемія (підвищення концентрації гемоглобіну в крові) - рідкісне явище, що трапляється за первинних і вторинних еритроцитозів, відносних

еритроцитозів за дегідратації, серцевої недостатності (вродженої, набутої), згущення крові.

Колірний показник це співвідношення між кількістю гемоглобіну і числом еритроцитів. Він показує ступінь насичення еритроцитів гемоглобіном.

Колірний показник вказує на вміст гемоглобіну в еритроцитах. Він може бути нормальний (нормохромний), знижений (гіпохромний), тобто нижчий за 0,8, або підвищений (гіперхромний), тобто вищий за 1.1, тому його використовують для диференціальної діагностики та патогенетичного лікування різних видів анемій.

Гематокрит - це частка від загального об'єму крові, яку складають еритроцити. Гематокрит відображає співвідношення еритроцитів і плазми крові, а не загальну кількість еритроцитів. Наприклад, у пацієнтів у стані шоку за рахунок згущення крові гематокрит може бути нормальним або навіть високим, хоча, внаслідок втрати крові, загальна кількість еритроцитів може значно знижуватися. Тому гематокрит не можна використовувати для оцінки ступеня анемії незабаром після втрати крові або гемотрансфузії. Гематокрит може дещо знижуватися при взятті крові в положенні лежачи. Хибно підвищені результати можуть спостерігатися в разі тривалого стискання вени джгутом під час взяття крові. Хибне зниження гематокриту може спостерігатися внаслідок розведення крові.

Тромбоцити - це невеликі плоскі клітини крові, що не мають кольору. Вони виконують кілька найважливіших функцій - беруть участь у згортанні крові, утворюють тромбоцитарний тромб, регулюють тонус судинної стінки, живлять капіляри.

Підвищення тромбоцитів під час розшифровки аналізу можливе в разі спленектомії (видалення селезінки), загострення хронічних аутоімунних захворювань, анемії різного ґенезу, запальних процесів, у післяопераційному періоді, третьому триместрі вагітності, в разі онкології, еритремії.

Знижуються тромбоцити в ЗАК у разі гемофілії, лікарської тромбоцитопенії, системного червоного вовчака, вірусних і бактеріальних інфекцій, апластичної анемії, синдрому Evans, аутоімунної тромбоцитопенічної пурпури, тромбозу

ниркових вен.

Ретикулоцити - це попередники еритроцитів, їхня проміжна форма. Вони виконують функцію газообміну, так само як і еритроцити, але з меншою ефективністю. У здорової людини ретикулоцити при розшифровці становлять від 0.2 до 1.2 % від загальної кількості еритроцитів.

Можуть бути підвищеними в разі постгеморагічного відновлення кровотворення, при переїзді в гірську місцевість або під час лікування анемії.

Знижуються ретикулоцити в загальному аналізі крові за ретикулоцитопенії (уповільнене кровотворення в кістковому мозку, що призводить до анемії).

Середній обсяг еритроцитів (MCV) - визначають практично всі сучасні гематологічні аналізатори. MCV вимірюється у фемтолітрах (фл). Значення, що перебувають у межах від 80 до 100 фл, характеризують еритроцит як нормоцит, нижче 80 фл - як мікроцит, а вище 100 фл - як макроцит.

Зміни MCV можуть слугувати для визначення порушень водно-електролітного обміну. Підвищення значень MCV свідчитиме про гіпотонічне порушення, тоді як зниження значень MCV - про гіпертонічне порушення. Тобто, обсяг еритроцитів зменшується при залізодефіцитної анемії. А підвищується - при В12 дефіцитної, при гепатитах, зниженні функції щитовидної залози.

Середній вміст гемоглобіну в еритроциті. Стабільна величина, яка не залежить від віку. Цей індекс прийшов на зміну застарілому колірному показнику. Зниження середнього вмісту гемоглобіну в еритроциті також свідчить про нестачу заліза. Підвищення може бути ознакою гіперхромної анемії.

Середня концентрація гемоглобіну в еритроцитах. Підвищення значень майже завжди свідчить про апаратну помилку, а зниження - про наявність залізодефіцитної анемії.

ШОЕ - швидкість осідання еритроцитів (ШОЕ) - це показник, що обчислюється в процесі лабораторної проби. Під впливом антикоагулянтів розраховується час осідання еритроцитів, який залежить від білкового складу плазми.

Підвищується за фізіологічних станів підвищення (вагітність, менструація), за інфекційних захворювань, злоякісних новоутворень, системних аутоімунних хвороб, хвороб нирок, у післяопераційному періоді, за травм та опіків.

Знижується при астено-невротичному синдромі, одужанні після інфекції, кахексії, тривалому прийомі глюкокортикоїдів, порушенні згортання крові, високій концентрації глюкози в крові, у разі черепно-мозкових травм, прийомі НПЗЗ, імунодепресантів, антибіотиків.

Лейкоцити - це білі кров'яні тільця. Основна їх функція полягає в захисті організму від мікробів і чужорідних речовин.

Підвищення числа лейкоцитів може говорити про лейкозі, запальному або інфекційному процесі, алергії, крововтраті, аутоімунних захворюваннях. Зниження числа лейкоцитів може говорити про деяких інфекціях (грип, краснуха, кір тощо), генетичної аномалії імунітету, підвищення функції селезінки, патології кісткового мозку.

Лейкоцитарна формула - це процентне співвідношення різних видів лейкоцитів.

Нейтрофіли - це найбільший підвид лейкоцитів, який залежно від зрілості клітин поділяється на такі групи - юні нейтрофіли, паличкоядерні та сегментноядерні.

Вони виконують антимікробну функцію, здатні до фагоцитозу, беруть участь у запальній реакції.

Підвищення нейтрофілів під час розшифровки аналізу крові можливе в разі фізіологічних станів (сонячний і температурний вплив, стрес, больовий синдром тощо), у разі перенесених інфекцій, хвороб кісткового мозку, онкології, вживання деяких медикаментів, кетоацидозу, отруєння отрутами й алкоголем, у разі паразитозу, алергії, гіперглікемії.

Знижуються в стані після хіміотерапії, за ВІЛ/СНІДу, апластичної анемії, тривалого інфекційного захворювання, впливу радіації, дефіциту вітаміну В12 і фолієвої кислоти.

Еозинофіли - чергова група лейкоцитів. Основна функція - боротьба з багатоклітинними паразитами. Крім того, еозинофіли беруть участь у

формуванні алергічних реакцій.

Підвищення кількості еозинофілів у крові - еозинофілія. На цей параметр потрібно звернути увагу, тому що здебільшого еозинофіли перебувають у тканинах, у крові їх зовсім мало - 0,5-2% від загальної кількості лейкоцитів.

Часті причини еозинофілії: алергічні захворювання, такі як бронхіальна астма, харчова або лікарська алергія, поліноз. У списку причин – ураження паразитами: аскаридами, ехінококом, лямбліями та іншими. Рідкісні випадки еозинофілії спостерігаються за раку - лімфоми та лейкемії.

Лімфоцити - це також підвид лейкоцитів, представлений у вигляді Т-лімфоцитів, В-лімфоцитів, К- і NK-лімфоцитів.

Усі вони беруть участь у набутому імунітеті, синтезують антитіла, руйнують не тільки чужорідні, а й власні патологічні клітини (онкологічні).

Підвищення в загальному аналізі крові може бути за вірусних інфекцій (мононуклеоз, вірусний гепатит тощо), токсопламозу, хвороб крові (лімфолейкоз хронічний і гострий, лімфома, лейкоз), за отруєння миш'яком, свинцем, приймання леводопи, наркотичних знеболювальних.

Знижуються лімфоцити за туберкульозу, ВІЛ, хвороб крові (лімфогранулематоз, апластична анемія), термінальної ниркової недостатності, онкологічних хвороб на термінальній стадії та в процесі лікування радіо- і хіміотерапією, прийому глюкокортикоїдів.

Базофіли - різновид лейкоцитів. Основна функція - участь у запальних та алергічних реакціях. Зміна кількості цих клітин у крові зустрічається рідко.

Підвищений вміст базофілів - базофілія. Може спостерігатися за гіпотиреозу - стійкої нестачі гормонів щитоподібної залози, а також за захворювань кісткового мозку.

Зменшення кількості базофілів - базопенія. Вона може з'являтися у відповідь на тиреотоксикоз - надлишок гормону щитоподібної залози, а також у разі реакції гострої гіперчутливості та інфекцій.

Моноцити - це вид найбільших лейкоцитів, що синтезуються також у кістковому мозку. Вони здатні фагоцитувати (поглинати) віруси, бактерії, пухлинні та паразитарні клітини. Регулюють кровотворну функцію, беруть

участь у згортанні крові.

Підвищення моноцитів під час розшифровки свідчить про вірусні, бактеріальні (туберкульоз, сифіліс, бруцельоз), грибкові та паразитарні інфекції, запалення в стадії регенерації, системні аутоімунні хвороби (системний червоний вовчак, ревматоїдний артрит), лейкоз.

Зниження моноцитів в аналізі крові можливе в разі гнійно-запальних процесів, апластичної анемії, у післяопераційному або післяпологовому періоді, під час прийому стероїдів [7].

Результати цього дослідження можуть мати суттєве практичне значення. Глибоке розуміння сприйняття та поглядів здорових пацієнтів, які звертаються за загальним аналізом крові, може допомогти розробити втручання, як індивідуальні, так і на рівні громади, розповісти пацієнтам про помилкові переконання щодо ролі аналізів крові, сприяти здоров'ю пацієнтів через здоровий спосіб життя. підтримувати стиль і проводити скринінгові тести з доведеною ефективністю, а також звести до мінімуму використання тестів, які можуть принести більше шкоди, ніж користі. Крім того, результати дослідження також можуть доповнити наше розуміння того, як лікарі можуть ефективніше розглядати ці запити, зберігаючи та покращуючи відносини між пацієнтом і лікарем.

Недоліки діагностування за загальним аналізом крові.

Репертуар аналізів крові, доступний лікуючому лікарю, величезний. Інші аналізи крові включають скринінг на утоімунні захворювання, моніторинг реакції раку на лікування онкомаркерами, оцінку репродуктивної функції, скринінг генетичних розладів під час вагітності та діагностику самої вагітності.

Незважаючи на це, іноді діагноз залишається невловимим, що викликає розчарування як у пацієнта, так і у лікаря. Неврологічні захворювання, такі як інсульт, хвороба рухових нейронів, хвороба Альцгеймера та розсіяний склероз, не піддаються діагностиці за допомогою аналізів крові. Подібним чином у діагнозах депресії, шизофренії, СДУГ та аутизму відсутній специфічний діагностичний маркер крові.

Величезний набір аналізів крові, доступний лікарю, у багатьох випадках

допомагає швидко поставити діагноз. Але вибір та інтерпретацію тесту слід розглядати з огляду на пацієнта та наявні у нього симптоми.

Таким чином, неможна досягти стовідсоткового попереднього діагнозу, так як аналіз крові не містить всіх показників, які відповідають за стан організму. Але в свою чергу попереднє діагностування цього і не вимагає, так як необхідно отримати максимально достовірний результат за короткий проміжок часу, ніж це доступно для пацієнтів в сьогоденні.

1.2. Обґрунтування методу розв'язання задачі

Для визначення діагнозу пацієнта потрібно знати нормальні значення кожного з показників загального аналізу крові (табл. 1.1), а також показники, визначені для кожної хвороби, яку буде діагностувати програмний засіб (табл. 1.2).

Нормальні величини - результати лабораторних досліджень у свідомо здорових людей.

Нормальні величини слугують цінними орієнтирами для лікарів, проте не можуть бути абсолютно точними показниками здоров'я і хвороби, оскільки їхні значення для здорових і хворих людей нерідко співпадають. Крім того, результати лабораторних досліджень можуть відрізнятися від справжніх значень через похибки вимірювань. Досвід впровадження гематологічних аналізаторів у клініко-діагностичні лабораторії показує, що результати, одержувані за їхньою допомогою, нерідко входять у протиріччя з усталеними в практиці ЛПЗ нормальними величинами.

Процедура встановлення нормальних величин будь-якого гематологічного параметра X включає в себе кілька етапів:

- вибір методу, за допомогою якого буде проводитиметься визначення нормальних величин параметра X;

- калібрування приладу, на якому буде проводитися визначення нормальних величин параметра X;

- підбір здорових донорів, у крові яких буде проводитися визначення нормальних величин параметра X;

- вимірювання параметра Х у донорів;

- статистичне опрацювання отриманих результатів.

Видно, що точне визначення норм - вельми складна і трудомістка процедура, чревата неоднозначністю і помилками:

- вибір методу вже несе в собі ту точність, із якою можуть бути встановлені нормальні величини параметра Х, якщо, наприклад, встановлювати норми концентрації еритроцитів за допомогою камери Горяєва, то межі цих норм будуть встановлені з більш ніж 15% похибкою, що відповідає такій камерного методу;

- калібрування приладу - окрема проблема;

- залежність значення багатьох параметрів від статі та віку вимагає обстеження великих однорідних статево-вікових груп, важкорозв'язною проблемою є встановлення нормальних значень у літніх людей, коли різні захворювання ускладнюють формування однорідних груп;

- під час вимірювання значень параметра Х необхідно ретельно контролювати правильність роботи приладу, на якому проводять вимірювання, під час усього періоду отримання результатів, також треба враховувати можливі помилки пре-аналітичного етапу взяття цих проб;

- у результаті статистичної обробки, як правило, за межі нормальних величин приймаються такі значення від нижньої нормальної межі до верхньої.

Тобто такі межі, у межі яких потрапляє 95% усіх виміряних значень. Це означає, що зі 100 виміряних здорових донорів у 5 осіб значення досліджуваного параметра може виходити за межі нормальних величин.

Необхідно враховувати, що на гематологічні показники, окрім патологічних чинників (травма, кровотеча, запалення, пухлина), впливають фізіологічні та конституціональні особливості хворих (стать, вік, фізичне та емоційне навантаження, споживання їжі, атмосферний тиск, голодування, температурні коливання тощо), а також різні хімічні та фізичні чинники (ліки, хімічні засоби, іонізуюча радіація).

Таким чином, загальний аналіз крові може забезпечити всебічний огляд пацієнта, групу захворювання та навіть його орієнтовний діагноз.

Таблиця 1.1 - Діапазони нормальних значень показників загального аналізу крові

№	Показники аналізу крові	позначення	Норма	
			чоловіки	жінки
1	Лейкоцити -white blood cells	WBC	4…9	
2	Еритроцити -red blood cells	RBC	4…5,1	3,7…4,7
3	Гемоглобін - hemoglobin	HGB	130…160	120…140
4	Гематокрит - hematocrit	HTC	0,38…0,50	
5	Середній об'єм еритроцитів - mean corpuscular volume	MCV	80…100	
6	Середній вміст гемоглобіну в еритроциті - mean concentration hemoglobin	MCH	24…34	
7	Середня концентрація гемоглобіну в еритроциті - mean corpuscular hemoglobin	MCHC	32…38	
8	Індекс розподілу еритроцитів — red cell distribution	RDW	4,5…14,5	
9	Тромбоциты - platelets	PLT	150…4000	
10	Тромбокрит - Thrombocrit	PTC	1…10	
11	Нейтрофіли (абс.) - Neutrophil	NEU	2…6	
12	Лімфоцити (абс.) - Lymphocyte	LYMP	1…3	
13	Моноцити (абс.) - Monocytes	MONO	0…0,5	
14	Еозінофіли (абс.) - Eosinophils	EO	0…0,5	
15	Базофіли (абс.) - Basophil	Baso	0…0,1	

Для реалізації даної методики доречно використати кластеризацію даних, яка зможе забезпечити високу точність їх обробки, при цьому мати нескладну програмну структуру. Показники загального аналізу крові у вигляді вхідного вектору даних будуть порівнюватись з векторами даних хвороб, найближчий вектор до вхідного визначає хворобу пацієнта. Якщо у хворого декілька захворювань, можна провести аналогічне порівняння, але з виключенням вже визначеної хвороби [8]. Кластеризація даних, що буде використовуватись для визначення попереднього діагнозу пацієнтів, є досить ефективною при використанні запропонованого підходу, оскільки вхідні дані представляють собою вектори чисел.

Діагностування пацієнтів на сьогоднішній день не часто проводиться із застосуванням програмних засобів. Зазвичай використовують апаратні засоби, орієнтовані на визначення однієї конкретної хвороби. Діагностування без спеціалізованого інструментарію найчастіше веде до визначення неправильного діагнозу, характеризується недостатньою швидкодією опрацювання даних.

Діагностування за загальним аналізом крові без використання програмних

засобів у сучасних умовах зміни глобального клімату, забруднення повітря, виникнення нових захворювань, їх еволюція, пристосування організму людини в залежності від її місцезнаходження до певних захворювань, оточуючого середовища та навіть їжі веде до поступової зміни нормальних значень показників загального аналізу крові людей, що проживають на певній території [9]. Тому може виникнути розбіжність при визначенні діагнозу при простому порівнянні лікарем аналізів крові пацієнта та нормальних значень загального аналізу крові. Це означає недостатню достовірність визначення діагнозу пацієнта за аналізом крові без використання інтелектуальних програмних засобів, які можуть опрацьовувати інформацію з урахуванням змін у показниках загального аналізу крові [10].

Артеріальний тиск (АТ) - це один із найважливіших життєвих показників стану здоров'я. Навіть незначне відхилення від нормального тиску може стати причиною розвитку несприятливих наслідків та тяжких ускладнень.

Артеріальний тиск відображає ступінь тиску крові на кровоносні судини під час систоли (скорочення серцевого м'яза) та діастоли (розслаблення серцевого м'яза).

В артеріальному тиску виділяють два показники:

- систолічний тиск (верхнє) - показує силу на стінки артерій під час викиду крові серцем;

- діастолічний тиск (нижній) - відбиває опір периферичних судин.

Артеріальний тиск вимірюється у міліметрах ртутного стовпа (мм.рт.ст.). При цьому першим вказується значення систолічного тиску, а другим – діастолічного.

Досить часто особи, які вперше зіткнулися з підвищеним чи зниженим тиском, не знають, який тиск має бути в нормі.

Рівні нормального артеріального тиску:

- оптимальний тиск, при якому верхній тиск становить не більше 120 мм.рт.ст., а діастолічний - менше 80 мм.рт.ст.;

- нормальний артеріальний тиск – це тиск у діапазоні 120-129 мм.рт.ст. (систолічне) та 80-84 мм.рт.ст. (діастолічний);

- підвищений нормальний тиск – тиск систоли становить 130-139 мм.рт.ст., а діастолічний – 85-89 мм.рт.ст.

На показники артеріального тиску впливають такі фактори:

- захворювання серця та кровоносних судин;
- сильні емоційні потрясіння;
- хвороби нирок;
- фізичні навантаження;
- регулярне вживання міцного чаю та кави;
- метеозалежність та ін.

Роль артеріального тиску в діагностуванні пацієнтів є дуже значимою. Навіть коли всі показники крові ще знаходяться в межах норми, артеріальний тиск може бути підвищеним, що є індикатором серцево-судинних хвороб. Показник тиску сам по собі не є надто цінною інформацією, але якщо він доповнюватиме показники аналізів крові, то допоможе з більшою вірогідністю діагностувати хвороби судин, серця, або нирок.

Високий кров'яний тиск, також званий гіпертонією, - це артеріальний тиск, який перевищує норму. Відомо, що артеріальний тиск змінюється протягом дня залежно від вашої діяльності. Якщо показники артеріального тиску постійно перевищують норму, це може призвести до діагнозу високого кров’яного тиску (або гіпертонії).

Високий кров'яний тиск зазвичай не має попереджувальних ознак або симптомів, і багато людей не знають, що він у них є. Вимірювання артеріального тиску - єдиний спосіб дізнатися, чи є у вас високий кров’яний тиск.

У рідкісних і важких випадках високий кров'яний тиск може спричинити: пітливість; занепокоєння; проблеми зі сном; почервоніння.

Якщо високий артеріальний тиск стає гіпертонічним кризом, людина може відчувати головний біль і носову кровотечу.

Високий кров'яний тиск, який не є наслідком іншого стану або захворювання, відомий як первинна або ессенціальна гіпертензія. Якщо основне захворювання є причиною підвищення артеріального тиску, лікарі називають

це вторинною гіпертензією.

Первинна артеріальна гіпертензія може бути наслідком кількох факторів, зокрема: наявність ожиріння; резистентність до інсуліну; велике споживання солі; надмірне вживання алкоголю; ведення малорухливого способу життя; куріння.

Вторинна гіпертензія має специфічні причини і є ускладненням іншої проблеми зі здоров'ям.

Хронічна хвороба нирок (ХНН) є поширеною причиною високого кров'яного тиску, оскільки нирки більше не фільтрують рідину. Цей надлишок рідини призводить до гіпертонії. Артеріальна гіпертензія також може спричинити ХХН.

Інші умови, які можуть призвести до гіпертонії, включають: діабет через проблеми з нирками та пошкодження нервів; феохромоцитома, рідкісний рак надниркової залози; синдром Кушинга; вроджена гіперплазія надниркових залоз, розлад надниркових залоз, що виділяють кортизол; гіпертиреоз або гіперактивність щитовидної залози; гіперпаратиреоз , який впливає на рівень кальцію та фосфору; вагітність; ожиріння.

Ряд факторів підвищує ризик гіпертонії:

- вік: Гіпертонія частіше зустрічається у людей старше65. Артеріальний тиск може неухильно зростати з віком, оскільки артерії стають жорсткішими та звужуються через накопичення бляшок;

- етнічна приналежність: деякі етнічні групи більш схильні до гіпертонії, ніж інші;

- вага: ожиріння є основним фактором ризику гіпертонії;

- вживання алкоголю та тютюну: регулярне вживання великої кількості алкоголю чи тютюну може підвищити артеріальний тиск;

- стать: чоловіки мають вищий ризик розвитку гіпертонії, ніж жінки. Однак це лише до тих пір, поки жінки не досягнуть менопаузи;

- існуючі захворювання: серцево-судинні захворювання, діабет, хронічна хвороба нирок і високий рівень холестерину можуть призвести до гіпертонії, особливо у віці.

Високий кров'яний тиск може зашкодити здоров'ю багатьма способами. Це може серйозно пошкодити такі важливі органи, як серце, мозок, нирки та очі. Чим вищий рівень артеріального тиску, тим більший ризик інших проблем зі здоров’ям, таких як серцеві захворювання, інфаркт та інсульт.

Серцевий напад і хвороби серця. Високий кров’яний тиск може пошкодити артерії, зробивши їх менш еластичними, що зменшує приплив крові та кисню до серця та призводить до серцевих захворювань. Крім того, зниження кровотоку до серця може спричинити:

- біль у грудях, також званий стенокардією;

- серцевий напад, який трапляється, коли кровопостачання серця блокується, і серцевий м’яз починає відмирати без достатньої кількості кисню, чим довше блокується кровотік, тим більше пошкодження серця;

- серцева недостатність - стан, який означає, що серце не може перекачувати достатньо крові та кисню до інших органів.

Інсульт і проблеми з мозком. Високий кров’яний тиск може спричинити розрив або блокування артерій, які постачають мозок кров’ю та киснем, що спричинить інсульт. Клітини мозку гинуть під час інсульту, оскільки не отримують достатньо кисню. Інсульт може спричинити серйозні порушення мови, рухів та інших основних видів діяльності. Інсульт також може вбити.

Високий кров’яний тиск, особливо в середньому віці, пов’язаний із погіршенням когнітивних функцій і деменцією в подальшому житті.

Хвороба печінки. Дорослі з діабетом, високим кров’яним тиском або тим і іншим мають вищий ризик розвитку хронічної хвороби нирок, ніж ті, у кого немає цих захворювань. Ниркова форма може розвиватися гостро чи хронічно та характеризується розвитком ниркової недостатності. Гостра ураження нирок пов'язана з інфарктом при тромбозі або тромбоемболії ниркової артерії.

Гіпертонічні кризи. Раптове підвищення АТ у пацієнтів з артеріальною гіпертензією, що супроводжується порушеннями вегетативної нервової системи та посиленням розладів мозкового, коронарного та ниркового кровообігу. Не всяке підвищення артеріального тиску клінічно відповідає гіпертонічному кризу. Лише раптове підвищення артеріального тиску до індивідуально високих цифр,

що супроводжується певною симптоматикою і призводить до порушення кровообігу в життєво важливих органах (мозок, серце, нирки) можна віднести до гіпертонічного кризу.

Кризи першого типу. Розвиваються зазвичай у I стадії гіпертонічної хвороби. Протікають із вираженою вегетативною симптоматикою (головний біль, нудота, серцебиття, пульсація і тремтіння у всьому тілі, тремор рук, поява плям на шкірі обличчя, шиї, збудження тощо).

Кризи другого типу. Протікають значно важче, триваліші, ніж кризи першого типу. Вони пов'язані з викидом у кров норадреналіну. Розвиваються зазвичай у пізні стадії гіпертонічної хвороби. Симптоми: різкий головний біль, запаморочення, перехідне порушення зору, слуху, болі в області серця, що стискають, нудота, блювання, парестезії, сплутаність свідомості.

Якщо не лікувати високий кров'яний тиск, він може підвищити ризик розвитку серйозних захворювань у подальшому житті, таких як серцевий напад, ниркова недостатність та інсульт.

Від часу появи COVID-19 до нинішнього дня науковці з усього світу пробують дослідити невідомий коронавірус та його особливості.

Зараз є чимало напрямків, пов'язаних з коронавірусною хворобою, які ще потребують вивчення: від незвичайних симптомів та методів виявлення до способів зараження і тривалості наявності в організмі антитіл.

Коронавірусна хвороба 2019 (COVID-19) - гостре респіраторне захворювання, спричинене зараженням важким гострим респіраторним синдромом коронавірусу 2 (SARS-CoV-2). COVID-19 має високий рівень госпіталізації, потреби в інтенсивній терапії та смертності. Виявлення пацієнтів із найвищим ризиком важкого захворювання є важливим для сприяння ранньому, агресивному втручанню та управлінню місцевими лікарняними ресурсами для пом’якшення кризових ситуацій у реанімації, які вплинули на деякі лікарняні системи. Загалом COVID-19 асоціюється з лімфопенією, іноді тромбоцитопенією та загальною лейкопенією під час госпіталізації. Клінічний перебіг пацієнтів, які госпіталізовані, різко змінюється, з ранніми доказами того, що госпіталізація у відділення інтенсивної терапії та ризик смертності

пов’язані з підвищеним рівнем D-димеру (димеризованого фрагмента плазміну D) і зниженням кількості лімфоцитів. Терміново потрібні додаткові рутинні біомаркери для стратифікації ризику пацієнтів.

Ширина розподілу еритроцитів (RDW) є стандартним компонентом рутинного повного аналізу крові. RDW кількісно визначає варіацію окремих об’ємів червоних кров’яних тілець (RBC), які змінюються від однієї клітини до іншої та для тієї самої клітини, коли вона циркулює протягом приблизно 115-денного періоду життя. RDW пов’язаний із підвищеним ризиком смертності від усіх причин: смертність від захворювань серця, легень, сепсису, грипу та раку; ускладнення, пов’язані з серцевою недостатністю, тяжкістю ішемічної хвороби серця та вірусним гепатитом, прогресуючою стадією та ступенем багатьох видів раку; і розвиток діабету, хронічної обструктивної хвороби легень, інсульту, анемії та багатьох інших станів. RDW, здається, є неспецифічним маркером захворювання, який має потенціал для забезпечення загальної кількісної стратифікації ризику, що може бути особливо корисним для нової та невідомої хвороби.

RDW - це коефіцієнт варіації об’єму еритроцитів або SD, поділений на середнє значення. Тому збільшення RDW має відповідати зменшенню середнього об’єму еритроцитів (MCV), збільшенню дисперсії об’єму еритроцитів або обом. Попередні дослідження виявили докази в деяких конкретних умовах того, що підвищення RDW спричинене затримкою очищення старих еритроцитів. Оскільки еритроцити, як правило, зменшують клітинний об’єм протягом свого життя, збереження цих старших менших клітин збільшує дисперсію об’єму, і ця затримка очищення збігається з чистим зменшенням виробництва еритроцитів і компенсує його. Ці звіти припускають можливість того, що підвищений RDW за деяких обставин може відображати клінічний стан, при якому утворення та оборот еритроцитів сповільнився в умовах підвищеного виробництва та обороту лейкоцитів або тромбоцитів, як це має місце при запаленні. Хоча остаточного механізму підвищення RDW ще не встановлено, є докази того, що RDW може забезпечити надійну стратифікацію ризику серед пацієнтів, у яких діагностовано однакове гостре

захворювання.

Згідного з дослідженням науковців Гарвардської медичної школи, збільшений показник RDW передбачає більш високий ризик смертності серед госпіталізованих з COVID-19.

Всього під час дослідження було проаналізовано показники 1641 госпіталізованих пацієнтів з коронавірусною інфекцією.

Підвищення RDW (>14,5%) асоціювалося з підвищеним ризиком смертності у пацієнтів будь-якого віку. RR для всієї когорти становив 2,73, з рівнем смертності 11% у пацієнтів з нормальним RDW (1173) і 31% у пацієнтів з підвищеним RDW (468). RR у пацієнтів молодше 50 років становив 5,25 (нормальний RDW, 1% [n = 341]; підвищений RDW, 8% [n = 65]); 2,90 у віковій групі від 50 до 59 років (нормальний RDW, 8% [n = 256]; підвищений RDW, 24% [n = 63]); 3,96 у віковій групі від 60 до 69 років (нормальний RDW, 8% [n = 226]; підвищений RDW, 30% [104]); 1,45 у віковій групі від 70 до 79 років (нормальний RDW, 23% [n = 182]; підвищений RDW, 33% [n = 113]); і 1,59 у віці ≥80 років (нормальний RDW, 29% [n = 168]; підвищений RDW, 46% [n = 123]). RDW був пов'язаний із ризиком смертності в моделях пропорційних ризиків Кокса, скоригованих на вік, рівень D-димеру (димеризований фрагмент плазміну D), абсолютну кількість лімфоцитів і поширені супутні захворювання, такі як діабет і гіпертонія (співвідношення ризиків 1,09 на 0,5% збільшення RDW та 2,01 для RDW >14,5% проти ≤14,5%;$P < 0,001$). Пацієнти, у яких RDW збільшився під час госпіталізації, мали вищу смертність порівняно з тими, у кого RDW не змінився; для тих, хто мав нормальний RDW, смертність зросла з 6% до 24%, а для тих, хто мав підвищений RDW на момент госпіталізації, смертність зросла з 22% до 40%.

У цьому когортному дослідженні пацієнтів, госпіталізованих через COVID-19, RDW, виміряний під час госпіталізації та під час госпіталізації, асоціювався зі статистично значущим підвищенням смертності. RDW - це рутинний лабораторний тест, який може бути корисним для стратифікації ризику госпіталізованих пацієнтів із COVID-19.

Таким чином, інформація про зміни в крові може бути вагомим

інструментом при створенні програмного модуля попереднього діагностування пацієнтів, враховуючи те, що ця хвороба дійшла кожного куточку землі і про неї знає переважна більшість населення. Швидке діагностування COVID-19 допоможе зберегти здоров'я та життя багатьох людей, застосувавши міри на ранніх стадіях хвороби.

Системи онлайн діагностування на даний момент не мають достатньої ефективності, а саме - не забезпечують потрібну швидкодію та високу достовірність визначення діагнозу пацієнта.

Найпопулярніша систему онлайн діагностування YourDiagnosis [11], базується на класичних медичних алгоритмах, що на даний момент можуть бути неноваторськими. Вона має точність діагностування у середньому - 70%. Процес діагностування включає в себе занесення у відповідні комірки значення показників загального аналізу крові. Це є достатньо простим для користувача, але одночасно і повільним. Отже, якщо потрібно визначити діагнози для декількох пацієнтів, дана система є недостатньо зручною. Неможливим є збереження введених завчасно даних, тому, якщо на сайті системи або з Інтернет-з'єднанням відбулась помилка, це призведе до того, що всі показники потрібно перенабирати знов. Швидкодія діагностування може сягати двох хвилин.

Інтерактивна система діагностування Diagnos.ua [12] має найбільшу в світі базу захворювань (близько 240) та заснована на методі нечіткої логіки. Процес діагностування даної системи є дуже повільним. Пацієнт повинен переглянути всі симптоми можливих хвороб та відмітити їх при наявності. Процес обробки даних після відмічення симптомів може сягати 10 хвилин. Достовірність діагностування даної системи у середньому складає 67%, оскільки визначення діагнозу лише за симптомами хвороби є ненадійним. Діагностування за аналізом крові на даному сайті не пристосоване для мобільних пристроїв. Достовірність діагностування у середньому складає 79%, швидкість - до 5 хвилин.



Рисунок 1.1 - Вид головного вікна сервісу YourDiagnosis

Отже, сучасні популярні системи діагностування мають наступні недоліки:

- засновані на недостатньо ефективних алгоритмах;
- мають недостатньо велику точність діагностування;
- пристосовані для визначення лише одної або декількох хвороб;
- швидкість визначення діагнозу є достатньо великою через незручний процес введення даних та складні обчислення;
- незручність визначення діагнозу для декількох пацієнтів;
- неможливість використання для телемедицини.

Таким чином, розробка програмного модуля для діагностування пацієнтів з використанням визначених показниках загального аналізу крові та методу кластеризації даних є перспективним підходом до попередньої перевірки діагнозу хворого, що зможе забезпечити достовірність визначення діагнозу пацієнта, високу швидкодію роботи, легкість та зручність для освоєння та використання, а також через реалізацію алгоритму кластеризації даних не буде потребувати особливих програмних та апаратних засобів, що дозволить провести впровадження системи кластеризації пацієнтів за аналізом крові в сучасні лікарні та замінить частину паперових записів, а також спростить організацію

ресурсів лікарні для визначення одразу кількох діагнозів пацієнтів та подальшої рекомендації їх спеціалізованим лікарям.

1.3. Постановка задачі

Даний програмний засіб передбачає виявлення попереднього діагнозу пацієнта за допомогою аналізу крові та артеріального тиску як надійного, швидкого та простого засобу виявлення хвороби, та кластеризації даних, що передбачає розподіл показників хвороб на групи (кластери), які визначають хворобу пацієнта, чиї показники крові підлягають обробці та, відповідно, спеціалізованого лікаря згідно із визначеною хворобою.

Враховуючи сучасний стан лікарень, розроблюваний програмний модуль повинен бути дешевим, простим у програмній реалізації, легко впроваджуваним та мати невеликий розмір на диску для можливого використання програми на непотужних комп'ютерах, мобільних гаджетах або через Інтернет. Програма не повинна потребувати додаткового програмного та апаратного забезпечення, причому підхід до реалізації повинен містити точні розрахунки та видавати безпомилкові результати. Тому даний програмний модуль повинен поєднувати раціональність, вірність розрахунків, невимогливість до обладнання, а також дружній інтуїтивний інтерфейс, чого потребують користувачі - пацієнти і працівники лікарень.

Отже, розроблюваний програмний модуль повинен забезпечувати наступні властивості:

- висока швидкодія та достовірність роботи за рахунок ефективного підходу та методу його реалізації процес розрахунків та обробки інформації суттєво пришвидшується та забезпечує велику достовірність визначення діагнозу пацієнта;

- простий і зручний доступ до потрібної інформації та функцій, у програмному модулі повинен бути ергономічний інтерфейс та нескладний процес роботи з ним, оскільки орієнтований на користувачів будь-якого рівня досвідченості роботи з комп'ютером.

- можливість роботи з великими обсягами даних, щоб використовуваний в

програмному засобі метод кластеризації даних дозволяє виконувати обчислення та подальшу обробку результатів для великих обсягів занесеної в систему інформації, при цьому достовірність визначення діагнозу не зменшується та зручність введення даних для їх подальшої обробки;

- невибагливість до програмного забезпечення та апаратного обладнання, особливо приймаючи до уваги обмеженість ресурсів для впровадження будь-якої системи у сучасні лікарні та для подальшого використання пацієнтами, програмний модуль буде розроблений таким чином, щоб витрати на його впровадження були як найменші, але не за рахунок можливостей, функцій, параметрів та загальної функціональності;

- безвідмовність роботи - програмний модуль за рахунок порівняної простоти реалізації майже не потребує втручання спеціалістів з технічного обслуговування, не викликаючи фатальних збоїв системи;

- безпека та конфіденційність - показники загального аналізу крові та визначений діагноз пацієнта, що використовує програмний модуль, не повинні бути доступні іншим користувачам.

Програмний модуль попереднього діагностування пацієнтів створюється для автоматизації процесу визначення діагнозу пацієнтів згідно з показниками їх загального аналізу крові. Його застосування оптимізує цей процес шляхом підвищення достовірності діагностування за рахунок використаного підходу, заснованого на аналізі крові та методу його програмної реалізації - кластеризації даних.

Для розв’язання задачі діагностування цим програмним засобом користувачу потрібно мати результати загального аналізу крові, які можна отримати в будь-якій державній лікарні. Виміряні значення показників аналізу крові заносяться у програмний модуль для подальшої обробки.

Вихідними даними є визначений попередній діагноз. Попередній означає те, що формулюється не точна назва хвороби, а група хвороб з тим, щоб пацієнт у подальшому звернувся до відповідного лікаря-фахівця з метою уточнення діагнозу.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ОСНОВНОГО КОМПОНЕНТА ПРОГРАМИ ПОПЕРЕДНЬОГО ДІАГНОСТУВАННЯ ПАЦІЄНТІВ

Класифікація - це процес віднесення невідомого об’єкта до одного з наперед заданих класів. Для вирішення задачі класифікації виявляються ознаки, які характеризують групи об'єктів досліджуваного набору даних - класи; за цими ознаками новий об'єкт можна віднести до того чи іншого класу.

Кластеризація - це автоматичне розбиття елементів деякої множини на групи схожих між собою об’єктів. Синонімами терміну "кластеризація" є "автоматична класифікація", "навчання без учителя" та "таксономія".

Завдання кластеризації подібне до завдання класифікації, є її логічним продовженням, але її відмінність полягає в тому, що класи досліджуваного набору даних заздалегідь не відомі. Таким чином, кластеризація призначена для розбиття сукупності об'єктів на однорідні групи (кластери або класи). Якщо дані вибірки представити як точки в просторі ознак, то завдання кластеризації зводиться до визначення "локалізації точок". Метою кластеризації є пошук існуючих структур даних. Кластеризація є описовою процедурою, вона не робить ніяких статистичних висновків, але дає можливість провести ознаковий аналіз та визначити структуру даних.

У штучних нейронних мережах під поняттям кластер розуміється підмножина «близьких один до одного» (схожих між собою) об'єктів із множини векторів-характеристик. Отже, кластер характеризує групу об'єктів, що мають загальні властивості [13].

2.1. Аналіз процесу кластеризації

Ефективність процесу кластеризації залежить від обраного методу, а отримані результати потребують подальшої інтерпретації, дослідження і вивчення властивостей і характеристик об'єктів для можливого точного опису сформованих кластерів. Кластеризація даних складається лише з трьох етапів, які включають в себе визначення характеристик, вибір метрики і подання результатів.

Спочатку обираються властивості, які характеризують задані об'єкти, ними можуть бути кількісні характеристики (інтервали, координати та ін.), якісні характеристики (статус, колір, звання та ін.) та ін. Потім проводяться спроби зменшення розмірності простору характеристичних векторів, тобто виділення найбільш важливих властивостей об'єктів. Зменшення розмірності прискорює процес кластеризації, але у ряді випадків дозволяє візуально оцінювати результат. Виділені характеристики нормалізуються. Далі всі об'єкти представляються у вигляді характеристичних векторів. В даній роботі буде використовуватись ототожнення об'єкту з його характеристичним вектором.

Наступним етапом кластеризації є вибір метрики, по якій буде визначатись близькість об'єктів. Метрика вибирається залежно від:

- простору розташування об'єктів;
- неявних характеристик кластерів.

Якщо всі координати об'єкта неперервні та дійсні, а кластери повинні являти собою щось на кшталт гіперсфери, то використовується класична евклідова метрика, математична формула якої представлена у вигляді залежності (2.1):

$$d_2(x_i, x_j) = \left[\sum_{k=1}^{d}(x_{i,k} - x_{j,k})\right]^{1/2} = \|x_i - x_j\|_2. \quad (2.1)$$

Результати кластеризації мають бути подані в зручному для обробки вигляді, щоб здійснити оцінку якості кластеризації. Зазвичай використовується один з таких способів:

- представлення кластерів набором характерних точок;
- представлення кластерів центроїдами;
- представлення кластерів їх обмеженнями.

Різні методи кластеризації можуть створювати різні кластери, і це є нормальним явищем. Однак створення схожих кластерів різними методами говорить про правильність кластеризації [14, 15].

2.2. Обґрунтування вибору методу кластеризації

У результаті застосування різних методів кластеризації даних можуть

бути отримані кластери різної конфігурації. Наприклад, можливі кластери ланцюгового типу, коли кластери представлені довгими "ланцюжками", кластери подовженої форми та інші, а деякі методи можуть створювати кластери довільної форми, що задаються користувачем. Різні методи можуть прагнути створювати кластери певних розмірів (наприклад, тільки малих або тільки великих), або дозволяти в наборі даних наявність кластерів різної величини. Деякі методи кластеризації особливо чутливі до шумів або викидів, інші – менш чутливі. У результаті застосування різних методів кластеризації можуть бути отримані неоднакові результати і це є нормальним явищем та особливістю роботи того чи іншого алгоритму. Всі ці особливості слід враховувати при виборі методу кластеризації даних. На сьогоднішні розроблено більше сотні різних алгоритмів кластеризації [14, 15].

Основні алгоритми кластеризації, серед яких будемо обирати: алгоритм k-means та його похідні; група ієрархічних алгоритмів; дерево мінімального покриття; алгоритм нечіткої кластеризації; метод найближчого сусіда; генетичні алгоритми; нейронні мережі.

Все частіше для вирішення завдань кластеризації використовують саме нейронні мережі.

У даного підходу є ряд переваг та особливостей:

- штучні нейронні мережі легко працюють в розподілених системах з великою паралелізацією в силу своєї природи;

- оскільки штучні нейронні мережі підлаштовують свої вагові коефіцієнти, ґрунтуючись на вихідних даних, це допомагає зробити вибір значущих характеристик менш суб'єктивним.

Існує велика кількість парадигм штучних нейронних мереж, наприклад, персептрон, радіально-базисні мережі, LVQ - мережі, самоорганізувальні мапи Кохонена, які можна використовувати для розв’язання задачі кластеризації. Але найкраще себе зарекомендувала мережа із застосуванням карт Кохонена, що саморганізуються.

Всі зазначені вище алгоритми мають свої переваги та недоліки, на основі яких буде зроблений подальший вибір для реалізації підходу до діагностування

із використанням показників загального аналізу крові.

Основним недоліком методу *k*-means є те, що потрібно заздалегідь задавати *k* - кількість кластерів та еталонів, що не завжди можливо зробити раціонально. Метод є дуже чутливим до початкових наближень значень центрів, можливі похибки при визначенні середнього значення кластера, заданої кількості ітерацій може бути недостатньо для точного визначення результату, якщо стабілізація центрів кластерів відбувається повільно, тому алгоритм працює досить повільно через велику кількість розрахунків.

Ієрархічні алгоритми іноді об'єднують під назвою структурного чи геометричного підходу, що володіє більшою формалізованістю поняття близькості. Всі вони спираються на вихідну «гіпотезу компактності»: у просторі об'єктів всі близькі об'єкти повинні ставитися до одного кластеру, а всі різні об'єкти відповідно повинні знаходитися в різних кластерах, що може призвести до великої похибки при кластеризації даних.

Алгоритм найближчого сусіда простий у реалізації, швидко виконується, але, як і інші «жадібні» алгоритми, може видавати неоптимальні рішення.

Дерево мінімального покриття може виділяти кластери довільної форми, але він має громіздку структуру, що ускладнить процес програмної реалізації та подальшого можливого застосування програмного засобу у телемедицині.

Нечіткі алгоритми мають високу обчислювальну складність і можуть бути непридатними, якщо кількість кластерів заздалегідь невідома або якщо кожен об'єкт потрібно однозначно віднести до одного кластера, що значно ускладнює діагностику декількох захворювань у пацієнта. Разом з тим, існує проблема невизначеності об'єкта, який є досить віддаленим від центрів усіх кластерів.

Для реалізації кластеризації даних мережею Кохонена, передбачається навчання мережі без учителя та самоорганізація мережі, існує суттєва простота реалізації та гарантоване отримання відповіді після проходження даних по шарах. Недоліками є робота тільки з числовими даними, мінімізація розмірів мережі та необхідність задання кількості кластерів, що не впливають на якість реалізації запропонованого підходу.

Генетичні алгоритми відрізняються від традиційних методів, оскільки

обробляють не значення параметрів самого завдання, а їх закодовану форму, здійснюють пошук рішення, виходячи не з єдиної точки, а з їх деякої популяції, використовують тільки цільову функцію, а не її похідні або іншу додаткову інформацію, застосовують імовірнісні, а не детерміновані правила вибору. Крім того, вони мають суттєві недоліки - не гарантують виявлення глобального рішення за прийнятний час, не гарантують, що знайдене рішення буде оптимальним та мають переваги перед іншими алгоритмами лише при дуже великих розмірах завдань і відсутності упорядкованості у вхідних даних, коли альтернативою є метод повного перебору варіантів [15].

Розроблюваний програмний модуль попереднього діагностування пацієнтів повинен займати невеликий розмір, забезпечувати високу швидкодію роботи та точність обробки числових даних. Отже, обраний метод повинен мати негроміздку структуру, забезпечувати достовірність діагностування при подальшій програмній реалізації. Найкращим методом в даній ситуації є використання нейронних мереж, а саме - нейронної мережі Кохонена.

2.3. Структура, математична модель та порядок функціонування нейронної мережі Кохонена

Мережа Кохонена використовує неконтрольоване навчання та навчальна множина складається лише із значень вхідних змінних. Мережа розпізнає кластери в навчальних даних і розподіляє дані до відповідних кластерів. Якщо в подальшому мережа зустрічається з набором даних, що не схожий ні з одним із відомих зразків, вона відносить його до нового кластеру. Якщо в даних містяться мітки класів, то мережа спроможна вирішувати задачі класифікації. Мережі Кохонена можна використовувати і в задачах, де класи відомі - перевага буде у спроможності мережі виявляти подібність між різноманітними класами [8, 16].

Мережа Кохонена має лише два шари: вхідний і вихідний, що називають само-організованою мапою. Елементи мапи розташовуються в деякому просторі - як правило двовимірному (рис. 2.1).

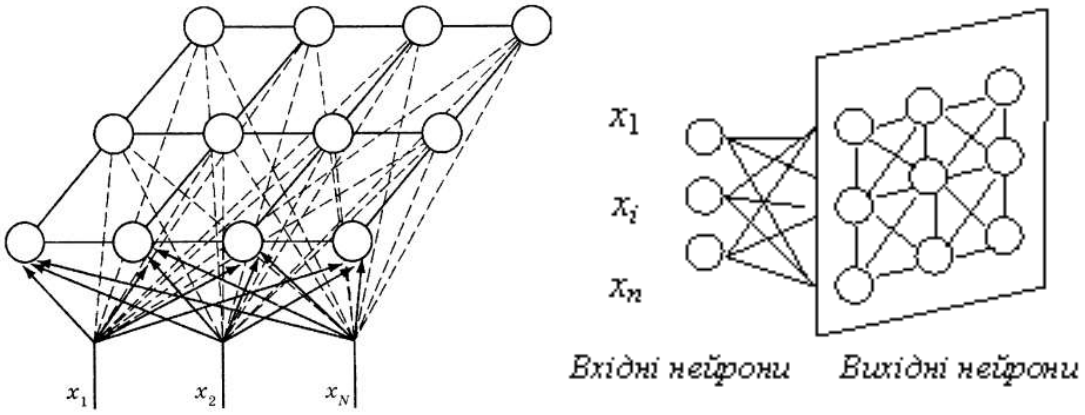


Рисунок 2.1 - Двовимірна структура мережі Кохонена

Мережа Кохонена навчається методом послідовних наближень. Починаючи з випадковим чином обраного вихідного розташування центрів, алгоритм поступово покращується для кластеризації навчальних даних.

Основний ітераційний алгоритм Кохонена послідовно проходить ряд епох, на кожній з яких опрацьовується один навчальний приклад. Вхідні сигнали - вектори дійсних чисел - послідовно пред'являються мережі. Бажані вихідні сигнали не визначаються. Після пред'явлення достатньої кількості вхідних векторів, синаптичні ваги мережі визначають кластери. Крім того, ваги організуються так, що топологічно близькі вузли є чутливими до схожих вхідних сигналів.

Для реалізації алгоритму необхідно визначити зону сусідства нейронів (окіл нейрона-переможця). На рис. 2.2 показані зони топологічного сусідства нейронів на карті ознак у різні моменти часу. $NE_j(t)$ - множина нейронів, що вважаються сусідами нейрона j у момент часу t. Зони сусідства зменшуються з часом.

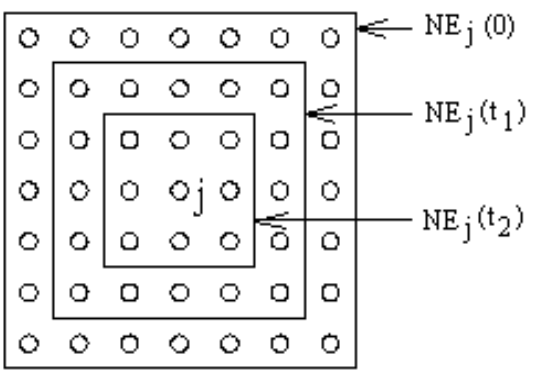


Рис. 2.2 - Зони топологічного сусідства на карті ознак у різні моменти часу

Потенційними функціями сусідства можуть бути визначені такі, як функція «мексиканський капелюх», яка обчислюється за формулою (рис. 2.3);

$$f_{gauss2}(d,\sigma) = \left(1 - \left(\frac{d}{\sigma}\right)^2\right) \cdot exp\left(-\left(\frac{d}{\sigma}\right)^2\right), \quad (2.2)$$

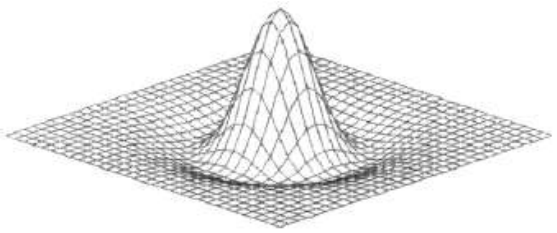


Рисунок 2.3 - Функція «мексиканський капелюх»

Дзвоноподібна функція Гауса, яка обчислюється за формулою (рис. 2.4);

$$f_{gauss1}(d,\sigma) = exp\left(\frac{d^2}{2\sigma^2}\right), \quad (2.3)$$

де σ^2 - дисперсія відхилення.

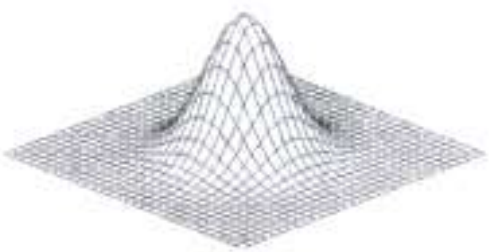


Рисунок 2.4 - Дзвоноподібна функція Гауса

Косинусоїдна функція, яка обчислюється за формулою (рис. 2.5);

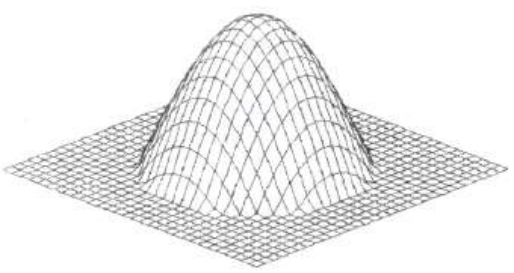


Рисунок 2.5 - Косинусоідна функція

$$f_{cos}(d,\sigma) = \begin{cases} cos\left(\frac{d\pi}{2\sigma}\right), d < \sigma; \\ 0, d \geq \sigma. \end{cases} \quad (2.4)$$

Циліндрична функція, яка обчислюється за формулою (рис. 2.6);

$$f_{cylinder}(d,\sigma) = \begin{cases} 1, d < \sigma; \\ 0, d \geq \sigma. \end{cases} \quad (2.5)$$

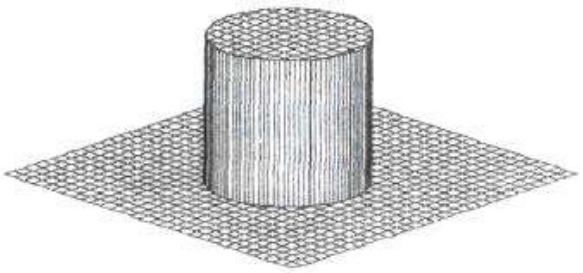


Рисунок 2.6 - Циліндрична функція

Конусоподібна функція, яка обчислюється за формулою (рис. 2.7);

$$f_{cone}(d,\sigma) = \begin{cases} 1 - \frac{d}{\sigma}, d < \sigma; \\ 0, d \geq \sigma. \end{cases} \quad (2.6)$$

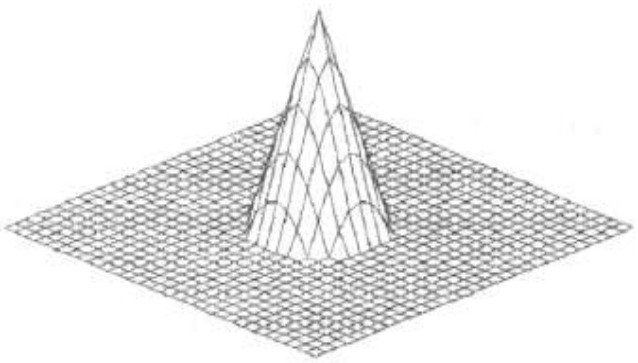


Рисунок 2.7 - Конусоподібна функція

Функція «французький капелюх», яка обчислюється за формулою (рис. 2.8);

$$h(p) = \begin{cases} 1, |p| \leq \propto; \\ -\frac{1}{3}, \propto < |p| \leq 3 \propto; \\ 0, |p| < 3 \propto. \end{cases} \quad (2.7)$$

Загальний алгоритм функціонування мережі Кохонена:

- ініціалізація (створення) мережі, де ваговим коефіцієнтам мережі надаються малі випадкові значення;

- пред'явлення мережі нового вхідного вектору;

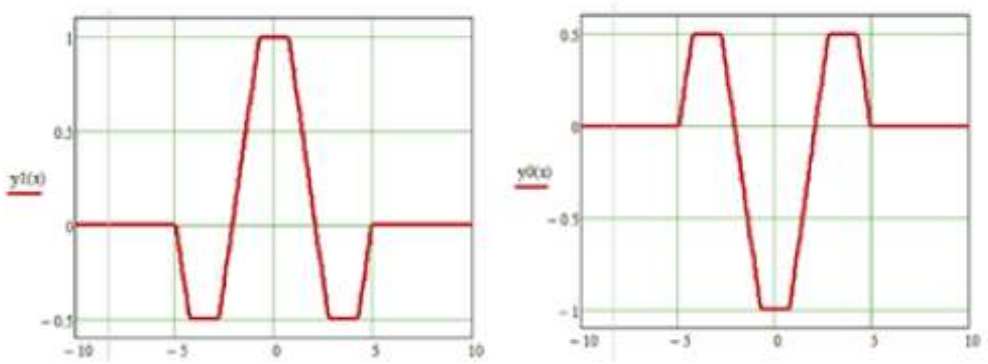


Рисунок 2.8 - Функція «французький капелюх»

- обчислення відстані вхідного вектора до всіх нейронів мережі, відстані d_j від вхідного вектору до кожного нейрона j визначаються за формулою (2.8):

$$d_j = \sum_{i=1}^{N} \left(x_i(t) \cdot \omega_{i,j}(t) \right)^2, \tag{2.8}$$

де x_i - *i-ий* елемент вхідного сигналу в момент часу t, $w_{ij}(t)$ - вага зв'язку від *i- го* елемента вхідного сигналу до нейрона j у момент часу t.

- вибір нейрона з найменшою відстанню (рис. 2.9) де, вибирається нейрон-переможець j^*, для якого відстань d_j мінімальна;

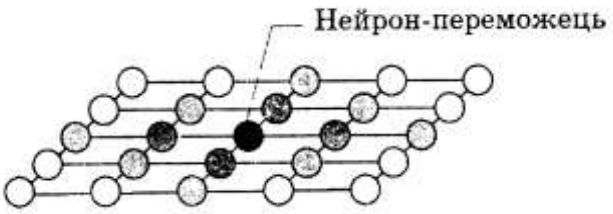


Рисунок 2.9 - Нейрон-переможець на шарі мапи Кохонена

- налаштування ваг нейрона j^* і його сусідів (рис. 2.10), де робиться налаштування ваг для нейрона j^* і всіх нейронів з його околу, а нові значення ваг обчислюються за формулою (2.9), де $r(t)$ - швидкість навчання, що зменшується з часом (додатне число, менше одиниці):

$$w_{ij}(t+1)=w_{ij}(t)+r(t)(x_i(t)-w_{ij}(t)) \tag{2.9}$$

- повернення до кроку пред'явлення мережі нового вхідного вектору.

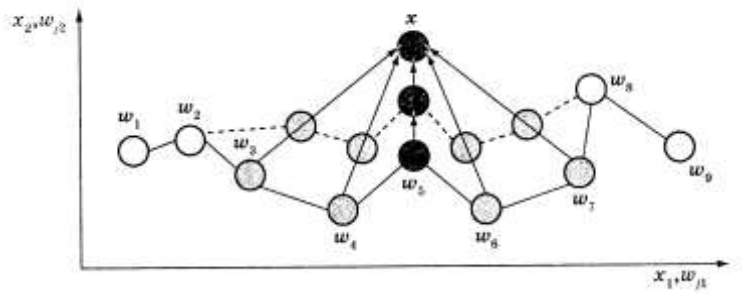


Рисунок 2.10 - Зміна ваг мапи Кохонена при навчанні

В алгоритмі використовується коефіцієнт швидкості навчання, який поступово зменшується для тонкішої корекції на новій епосі. В результаті позиція центру встановлюється в певній позиції, яка задовільним чином кластеризує приклади, для яких даний нейрон є переможцем.

Властивість топологічної впорядкованості досягається в алгоритмі за допомогою використання поняття околу. Окіл (радіус сусідства) - це декілька нейронів, що оточують нейрон-переможець. Відповідно до швидкості навчання, розмір околу поступово зменшується, так, що спочатку до нього належить досить велика кількість нейронів (можливо вся мапа), а на самих останніх етапах окіл стає нульовим і складається лише з нейрона- переможця. В алгоритмі навчання корекція застосовується не тільки до нейрона-переможця, але і до всіх нейронів з його поточного околу. У результаті такої зміни околу початкові доволі великі ділянки мережі мігрують у бік навчальних прикладів. Мережа формує укрупнену структуру топологічного порядку, при якій схожі приклади активують групи нейронів, що знаходяться поруч на топологічній мапі. З кожною новою епохою коефіцієнт швидкості навчання і розмір околу зменшуються, тим самим всередині ділянок мапи виявляються більш тонкі розходження, що, зрештою, призводить до точнішого налаштування кожного нейрона. Часто навчання зумисне розбивають на дві фази: більш коротку, з великою швидкістю навчання і великих околів, і більш тривалу з малою швидкістю навчання і нульовими або майже нульовими околами [16, 17].

Загальний алгоритм роботи програмного модуля попереднього діагностування пацієнтів на основі нейронної мережі Кохонена наведений на рис. 2.11.

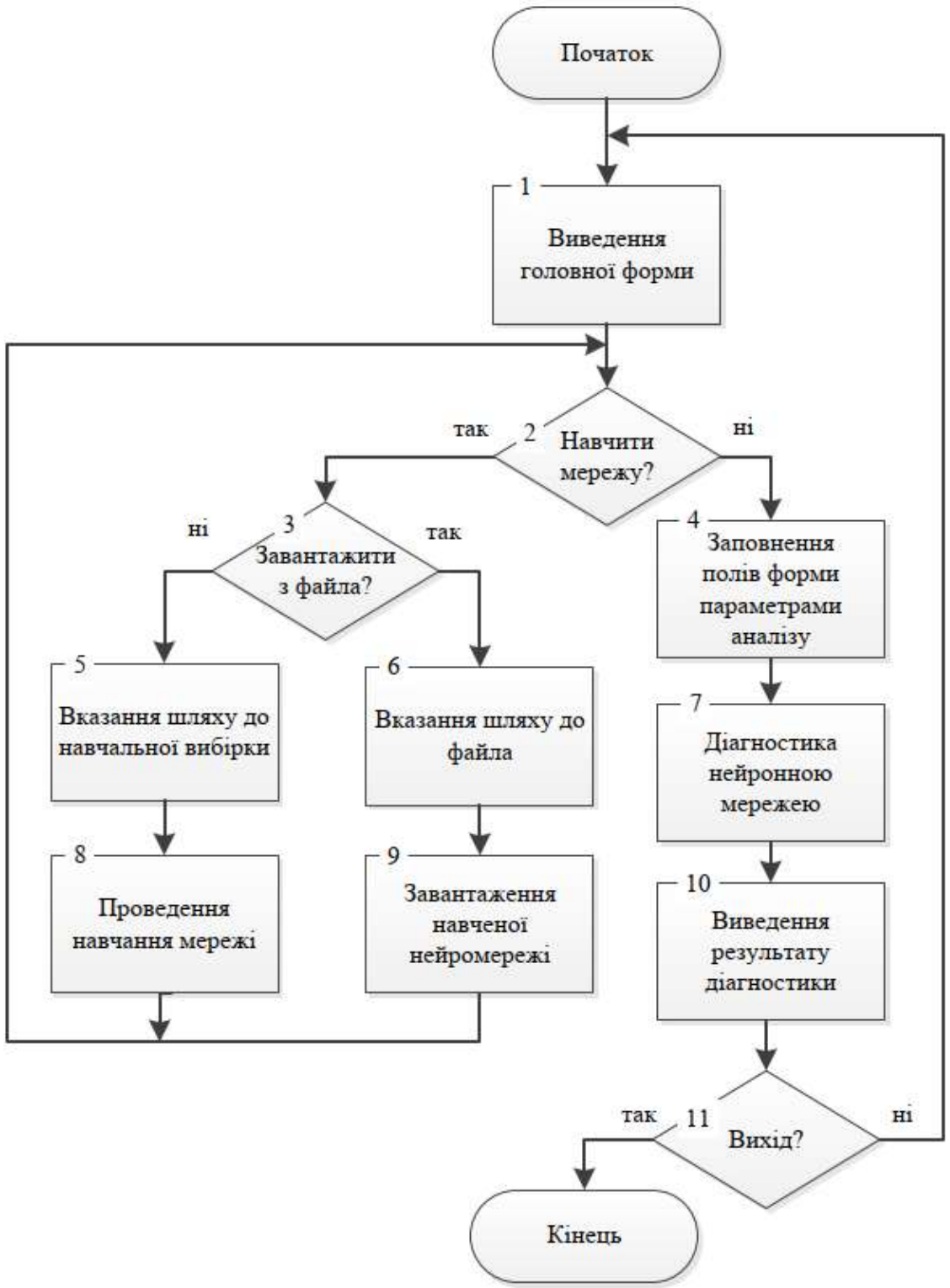


Рисунок 2.11 - Загальний алгоритм роботи програмного модуля попереднього діагностування пацієнтів на основі нейронної мережі Кохонена

Архітектура розроблюваного програмного модулю попереднього діагностування пацієнтів на основі нейронної мережі Кохонена складається з двох шарів нейронів (рис. 2.12). Перший шар не є фактично нейронним шаром,

він тільки отримує вхідні дані і передає їх на другий шар. Кожен нейрон другого шару має свій вектор ваг, розмірність якого збігається з розмірністю вхідного шару. Нейрони з'єднуються з сусідніми нейронами, що визначає топологію і структуру мапи Кохонена. Нейрони вихідного шару ще називають кластерними елементами, їх кількість визначає максимальну кількість груп, на які система може розділити вхідні дані. Збільшуючи кількість нейронів другого шару, можна збільшувати деталізацію результатів процесу кластеризації. Оскільки програмний модуль буде визначати 12 груп хвороб для чоловіків та 12 для жінок, доцільним є задання кількості нейронів шару Кохонена не менше 24 (по одному нейрону накожен кластер). А оскільки для надійної роботи мережі потрібно передбачити певну кількість «запасних» нейронів, то оберемо кількість нейронів шару Кохонена 30.

Оскільки стандартна реалізація мережі Кохонена не передбачає визначеної кількості кластерів, та їх початкові центри приймають випадкові малі значення. Проведена модифікація цієї мережі, що полягає в попередньому визначенні кількості кластерів та їх центрів, що відповідають нормальним та аномальним значенням загального аналізу крові (табл. 1.1).

Мережа має два шари: вхідний та вихідний (шар Кохонена) (рис. 2.12). Шар Кохонена складається з деякої кількості n паралельно діючих лінійних елементів. Всі вони мають однакову кількість входів m та отримають на свої входи один і той самий вектор вхідних сигналів $x=(x_1 \ldots x_m)$. На виході j-го лінійного елемента отримуємо сигнал, що розраховується за формулою:

$$y_j = \omega_{j0} \sum_{i=1}^{m} \omega_{ji} x_i, \qquad (2.10)$$

де ω_{ji} - ваговий коефіцієнт i-го входу j-го нейрону, ω_{j0} - пороговий коефіцієнт.

Після проходження шару лінійних елементів сигнали подаються на обробку за правилом «переможець забирає все»: серед вихідних сигналів y_i шукається максимальний; його номер $j_{max} = arg\max_j\{y_j\}$. Остаточно, на виході сигнал з номером j_{max} дорівнює одиниці, а всі інші - нулю. Якщо максимум одночасно досягається для декількох j_{max}, то приймають всі відповідні сигнали рівними одиниці.

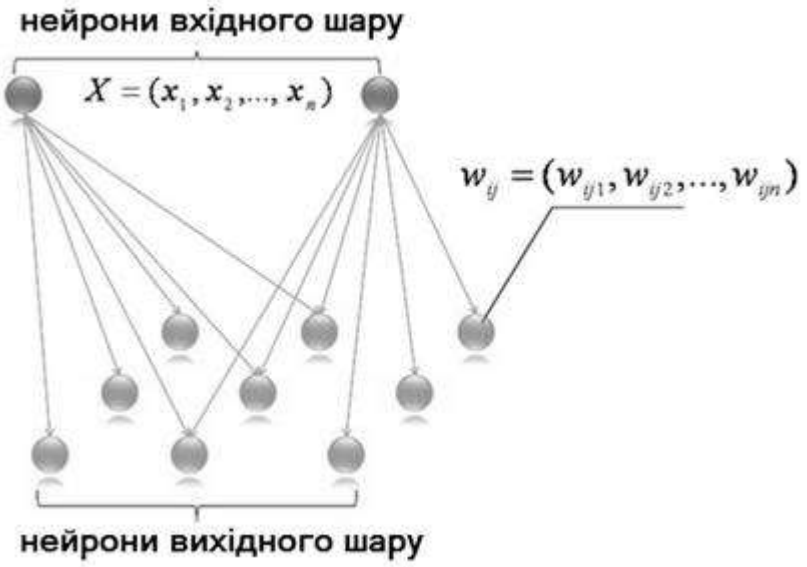


Рисунок 2.12 - Структура мережі Кохонена розроблюваного програмного модулю

Навчання розроблюваної мережі відбувається так. На початку роботи визначається кількість кластерів та їх центри. Дана інформація визначена заздалегідь: кількість кластерів становить 30, початкові центри кластерів відповідають наборам показників хвороб для визначених груп хвороб. Після цього деякий вхідний вектор з набору навчальних вибирається і встановлюється на вході нейронної мережі. На цьому етапі відмінності між вхідним вектором та всіма векторами обчислюються за формулою:

$$D_{ij} = \left|X^l - W_{ij}\right| = \sqrt{\left(x_1 - \omega_{ij1}\right)^2 + \cdots + \left(x_n - \omega_{ijn}\right)^2}, \quad (2.11)$$

де i та j - показники нейронів у вихідному шарі. Після цього нейронна мережа обирає нейрон-переможець з переліку визначених центрів кластерів, тобто такий, щоб його ваговий вектор був схожий на вхідний за формулою:

$$D(k_1, k_2) = min_{i,j} D_{i,j}, \quad (2.12)$$

де k_1 та k_2 - показники нейрона-переможця. Після цього проводиться корекція вагових векторів переможця та сусідніх з ним нейронів. Ближні нейрони до переможця визначаються топологічною функцією сусідства «Мексиканський капелюх», яка розраховується за формулою:

$$h(p,t) = exp\left(-\frac{p^2}{\sigma^2}\right)\left(1 - \frac{2}{\sigma^2(t)}p^2\right), \quad (2.13)$$

де p - відстань до нейрона переможця, яка знаходиться за формулою:

$$p = \sqrt{(k_1 - i)^2 + (k_2 - j)^2}, \quad (2.14)$$

де σ - функція, що визначає радіус сусідства. На початку функціонування програмного модулю вона включає весь простір сенсорного поля (сітки), але з часом значення її зменшується.

Після обчислення топологічної функції ваги нейронів переобчислюються за формулою (2.15):

$$W_{ij}(t+1) = W_{ij}(t) + \propto (t) \cdot h(p,t) \cdot \left(X^l(t) - W_{ij}(t)\right), \quad (2.15)$$

де $\propto (t)$ - функція швидкості навчання, яка також змінюється з часом.

Якщо нейрон є переможцем або сусіднім до нього, його вектор ваг оновлюється або залишається незмінним в іншому випадку. На кожному кроці нейронна мережа визначає нейрон, чий ваговий вектор найбільш подібний до вхідного, та коригує його ваги та ваги сусідів, щоб наблизити їх до вхідного вектора (рис. 2.13).

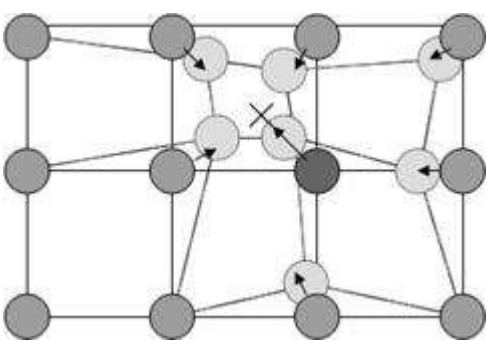


Рисунок 2.13 - Оновлення нейрона-переможця та його сусідів та «підштовхування» в сторону вхідного вектора, що на рисунку позначений «×». Суцільні та пунктирні лінії визначають ситуацію до та після оновлення, відповідно.

Кожен вхідний вектор з навчальної вибірки представляється нейронній мережі і навчання триває або деяке фіксоване число циклів, або доки різниця між вхідним і ваговим векторами досягне заданого значення ε. Різниця між сусідніми нейронами зменшується з плином часу, і, отже, вони організовуються у групи (кластери), які відповідають одному з класів з навчальної множини.

З метою оптимізації обчислень запропонованої мережі при розробці

програмного модуля також була введена процедура попередньої обробки вхідних даних, при якій значення ознак, що складають вхідний вектор, апроксимуються до певного діапазону. Нормалізація необхідна, оскільки вихідні значення ознак змінюються в достатньо великому діапазоні, тому існує ймовірність некоректної роботи нейронної мережі з такими даними. Так, в одному вхідному векторі можуть міститися значення, що відрізняються один від одного на кілька порядків: стандартний рівень гемоглобіну - 120-180 г/л, колірний показник - 0,85-1,15%. Такий дисбаланс між значеннями показників може викликати нестійкість роботи моделі, погіршувати результати навчання і уповільнювати його процес. Після нормалізації всі значення вхідних ознак будуть приведені до деякого вузького діапазону, а саме - [0 ... 1], що дозволяє мережі працювати з даними коректніше [17].

Для реалізації запропонованого підходу до діагностування була використана нормалізація вхідних даних за формулою:

$$y = \frac{(x - x_{min}) + (d_2 - d_1)}{x_{max} - x_{min}} + d_1, \quad (2.16)$$

де: x - значення, що підлягає нормалізації; $[d_2, d_1]$ - інтервал значень x; $[x_{max}, x_{min}]$ - інтервал, до якого буде зведене значення x.

Таким чином, був розроблений алгоритм, за яким буде у подальшому працювати програмний модуль попереднього діагностування пацієнтів на основі нейронної мережі Кохонена.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ ПОПЕРЕДНЬОГО ДІАГНОСТУВАННЯ ПАЦІЄНТІВ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ КОХОНЕНА

3.1. Обґрунтування вибору мови та середовища програмування

На даний час існує багато програмних середовищ, що підтримують можливість створення нейронних мереж, основними з них є середовища та мови програмування Deductor, MatLAB, Java, C++, C#, кожна з яких має свої недоліки та переваги.

Програмне середовище Deductor передбачає вбудовану реалізацію нейронних мереж, кількість яких в даній програмі обмежена. Дане середовище слугує, насамперед, для користувацьких цілей та може бути використане, зокрема, у якості програми для тестування написаного коду нейронної мережі.

Програмне середовище MatLAB за допомогою вбудованих бібліотек у повній мірі підтримує створення нейронних мереж, але має досить негнучку структуру, обмежені можливості для використання та створення візуального інтерфейсу для написаної програми, додатки, створені за допомогою даного середовища, зазвичай, складні у редагуванні та запуску на різних платформах.

Мова програмування Java доволі проста, безпечна та передбачає запуск на будь-яких платформах. Вона має широку бібліотеку, яка дозволяє повністю реалізувати будь-яку нейронну мережу, але у той же час програми, написані даною мовою є громіздкими та залучають багато ресурсів та часу для виконання, що може бути недоцільним при утворенні програмного модуля, орієнтованого на пересічного користувача.

Мови програмування C++ та C# мають схожі переваги з мовою Java, але бібліотеки цих мов є більш функціональними саме для створення нейронних мереж, програми, написані на C++ та C# виграють в швидкодії та запуску на різних платформах, а також легко можуть модифікуватись за бажанням.

Всі наведені вище програмні середовища мають можливості реалізації створюваного програмного модуля, у той же час програмне середовище, що обирається, повинно бути, по-перше, кросплатформним, по-друге, програми,

створені за допомогою нього повинні легко редагуватись та модифікуватись за бажанням користувача, та по-третє, мати порівняно високу швидкодію виконання програм [14, 17, 18].

Таким чином, взявши до уваги вимоги до мови програмування та оцінивши переваги та недоліки представлених програмних середовищ, було обрано мову C# та середовище Visual Studio. Оскільки мова C# має більше можливостей для реалізації нейронних мереж, ніж C++, а також для неї було створено безліч розширень та бібліотек нейронних мереж, незважаючи на незначну перевагу по швидкості C++, саме мова програмування C# була обрана для реалізації програмного модуля попереднього діагностування пацієнтів на основі нейронної мережі Кохонена.

Також була використана бібліотека Accord.NET [19] - це фреймворк для наукових обчислень в .NET. Фреймворк містить набір бібліотек, доступних у вихідному коді, а також через виконувані установники і пакети NuGet. Основні області застосування включають чисельну лінійну алгебру, числову оптимізацію, статистику, машинне навчання, штучні нейронні мережі, обробку сигналів і зображень та допоміжні бібліотеки (такі як побудова графіків і візуалізація). Спочатку проект був створений для розширення можливостей AForge.NET Framework, але з тих пір він включає в себе AForge.NET. Нові версії об'єднали обидві платформи під ім'ям Accord.NET.

3.2. Реалізація основних компонентів програмного модуля попереднього діагностування пацієнтів на основі нейронної мережі Кохонена

Опишемо оператори, функції та класи, які використовуються для представлення нейронних мереж за допомогою мови програмування C#.

Приклад класів для подання елементів мережі:

```
// Вхід
public class Input
{
  // Зв’язки з нейронами
  public Link[]  OutgoingLinks;
}
// Зв'язок входу з нейроном
```

```
public class Link
{
  // Нейрон
  public Neuron Neuron;
  // Вага зв'язку
  public double Weight;
}
public class Neuron
{
  //Всі входи нейрону
  public Link[]  IncomingLinks;
  // Накопичуваний нейроном заряд
  public double Power { get; set; }
}
```

Для мережі Кохонена приклад задання масивів вхідних векторів та ваг нейронів:

```
public class KohonenNetwork
{
  private readonly Input[] _inputs;
  private readonly Neuron[] _neurons;
...
}
```

Приклад обрахунку ваг нейронів:

```
// Надати вектор на вхід нейронної мережі
public int  Handle(int[]  input)
{
  for (var i = 0; i < _inputs.Length;  i++)
  {
  var inputNeuron  = _inputs[i];
  foreach (var outgoingLink  in inputNeuron.OutgoingLinks)
  {
   outgoingLink.Neuron.Power  += outgoingLink.Weight  * input[i];
  }
  }
  var maxIndex  = 0;
  for (var i = 1; i < _neurons.Length; i++)
  {
  if (_neurons[i].Power > _neurons[maxIndex].Power)
  maxIndex  = i;
  }
  //Зняти імпульс з усіх нейронів
  foreach (var outputNeuron in  _neurons)
  {
   outputNeuron.Power = 0;
  }
  return maxIndex;
}
```

Приклад навчання:

```
public void Study(int[]  input, int correctAnswer)
{
  var neuron = _neurons[correctAnswer];
  for (var i = 0; i < neuron.IncomingLinks.Length;  i++)
  {
  var incomingLink  = neuron.IncomingLinks[i];
  incomingLink.Weight = incomingLink.Weight  + 0.5 * (input[i]  - incomingLink.Weight);
  }
}
```

Основними класами програмного модуля попереднього діагностування

пацієнтів на основі нейронної мережі Кохонена є Neuron та NeuralNetwork, що представлені на рис. 3.1. Наведена діаграма класів, побудована у середовищі Rational Rose, демонструє поля кожного класу (Fields), його атрибути (Properties), методи (Methods) та події (Events). Дані класи знаходяться у відносинах залежності, оскільки клас NeuronNetwork, фактично, оголошує клас Neuron та створює 86 його екземплярів, тобто існування класу Neuron залежить від існування класу NeuralNetwork.

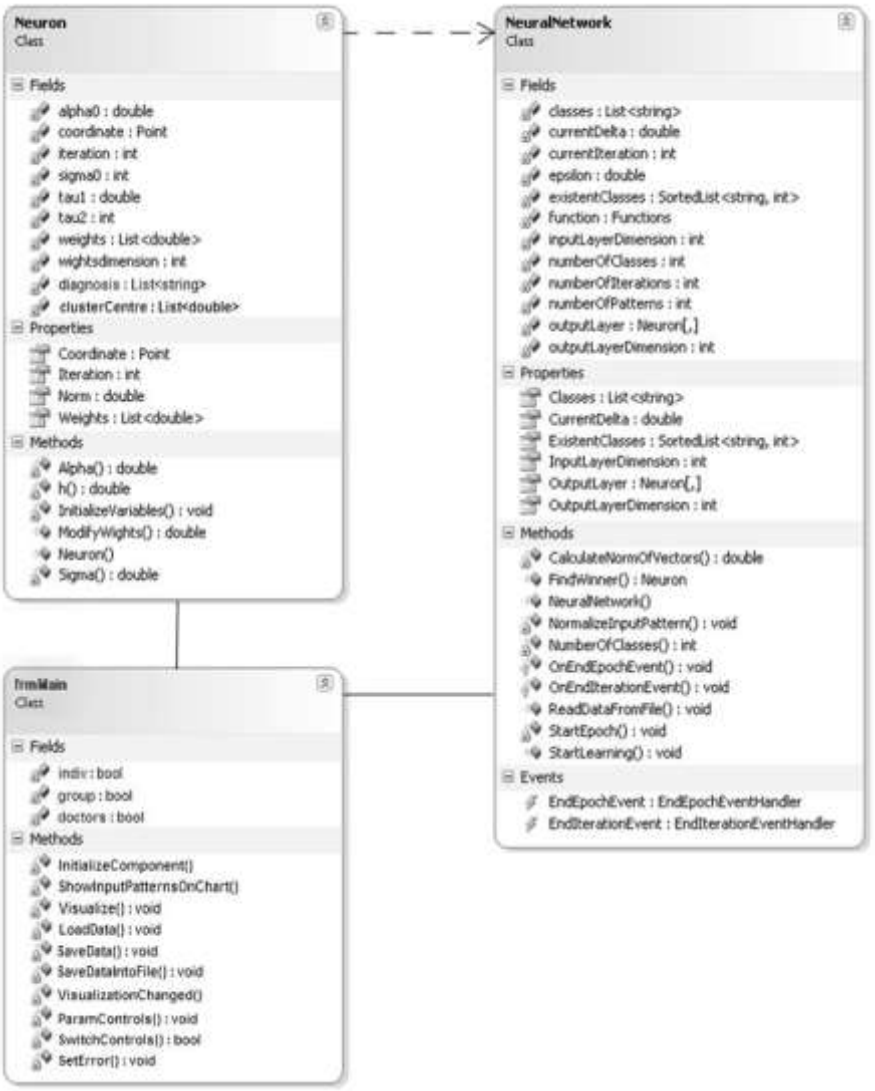


Рисунок 3.1 - Діаграма класів SOFM.Neuron та SOFM.NeuralNetwork

Кожен класс Neuron має поле Coordinate, що показує позицію нейрона в двовимірній сітці. Поле Weights - це вектор ваг типу generic, що містить значення типу double. Клас Neuron може виконувати операцію ModifyWeights, що переобчислює вектор ваг нейрону, згідно з обраною функцією сусідства «Мексиканський капелюх» (h), формула якої занесена у програмний код,

відстанню до нейрона-переможця або значення епсілон, яке за замовченням дорівнює 10^{-4}.

Клас NeuralNetwork містить двовимірну сітку нейронів та може виконувати певний набір операцій:

public NeuralNetwork(int 30, int numberOfIterations, double epsilon, Functions f) // Створює нейронну мережу, що містить 30 нейронів на другому шарі, та визначену кількість ітерацій, епсілон та функцію топологічного сусідства «Мексиканський капелюх».

public void ReadDataFromFile(string inputDataFileName)

// Читає інформацію вхідних даних з певного файлу

public void StartLearning() // Починає навчати мережу.

Обчислення функції «Мексиканський капелюх» відбувається за наступною процедурою:

```
Function.MexicanHat:
{
  distance = Math.Sqrt(Math.Pow((winnerCoordinate.X  - coordinate.X), 2) +
Math.Pow((winnerCoordinate.У  - coordinate.У), 2));
  result = Math.Exp(-(distance  * distance) / Math.Pow(Sigma(iteration),  2)) * (1
- (2 / Math.Pow(Sigma(iteration),  2)) * (distance * distance));
break;
```

Всі шаблони з навчального набору продовжують подаватися доки вектор ваг не буде переобчислюватись на незначну величину, що не відрізняється від початкової (currentEpsilon <= Epsilon).

```
public void StartLearning()
{
  int iterations  = 0;
  while  (currentEpsilon  > epsilon)
  {
  List <List<double>>patternsToLearn = new
List<List<double>>(numberOfPatterns);
  foreach (List<double> pArray in  patterns)
  patternsToLearn.Add(pArray);
  Random randomPattern = new Random();
  List<double> pattern = new List<double>(inputLayerDimension);
  for (int i = 0; i < numberOfPatterns; i++)
  {
  pattern = patternsToLearn[randomPattern.Next(numberOfPatterns  - i)];

StartEpoch(pattern);

   patternsToLearn.Remove(pattern);
  }
  iterations++;
   OnEndIterationEvent(new EventArgs());
  }
}
```

Метод типу private StartEpoch() означає знаходження нейрона- переможця та початку процесу перед обчисленням ваг нейронів на всіх зовнішніх шарах

outputLayer.

```
private void StartEpoch(List<double> pattern)
{
 Neuron Winner = this.FindWinner(pattern);
 currentEpsilon = 0;
 for (int i = 0; i < outputLayerDimension; i++)
 for (int j = 0; j < outputLayerDimension; j++)
 {
 currentEpsilon +=
 outputLayer[i, j].ModifyWights(pattern, Winner.Coordinate,
 currentIteration, function);
  }
 currentIteration++;
 currentEpsilon = Math.Abs(currentEpsilon /
  (outputLayerDimension * outputLayerDimension));
 EndEpochEventArgs e = new EndEpochEventArgs();
 OnEndEpochEvent(e);
}
```

Таким чином, епоха визначена як процес подання одного з шаблонів на вхід нейронної мережі, а ітерація являє собою набір епох, коли всі шаблони із навчальної множини представлені на входи нейронної мережі. Щоразу, коли епохи або ітерації закінчуються, відбуваються відповідні події [35, 37-40].

```
protected virtual void OnEndEpochEvent(EndEpochEventArgs e)
{
 if (EndEpochEvent != null)
  EndEpochEvent(this, e);
}

protected virtual void OnEndIterationEvent(EventArgs e)
{
 if (EndIterationEvent != null)
  EndIterationEvent(this, e);
}
```

Таким чином, в даному розділі були зазначені переваги обраної мови програмування, яка була використана для програмної реалізації запропонованого підходу, визначені класи програмного модуля та представлені елементи програмного коду, що демонструють основні виконувані операції.

3.3. Тестування програмного модуля попереднього діагнозу

Завданням розробленого програмного модуля - встановлення попереднього діагнозу пацієнта на основі показників його загального аналізу крові.

Після запуску програми відкриється головне вікно (рис. 3.2).

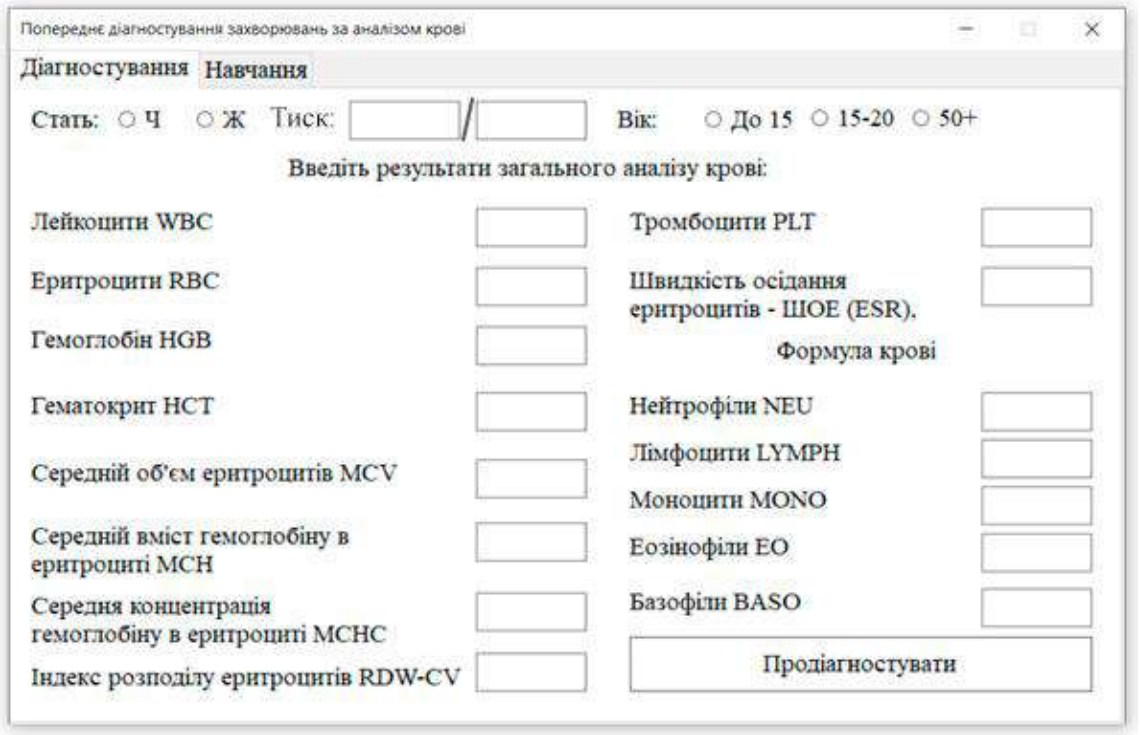


Рисунок 3.2 - Початкове вікно програмного модуля

Якщо нейронна мережа ще не пройшла процес навчання, то потрібно обрати вкладку «Навчання», після чого з’явиться вікно, представлене на рис. 3.3. Там можна обрати одну із 2 опцій:

- завантажити навчену нейронну мережу із файла;

- провести навчання нейромережі на нових даних.

У обох випадках з’явиться додаткове вікно навігатора для вказання місця знаходження потрібного файла на диску, як показано на рис. 3.4.



Рисунок 3.3 - Вкладка «Навчання» програмного модуля

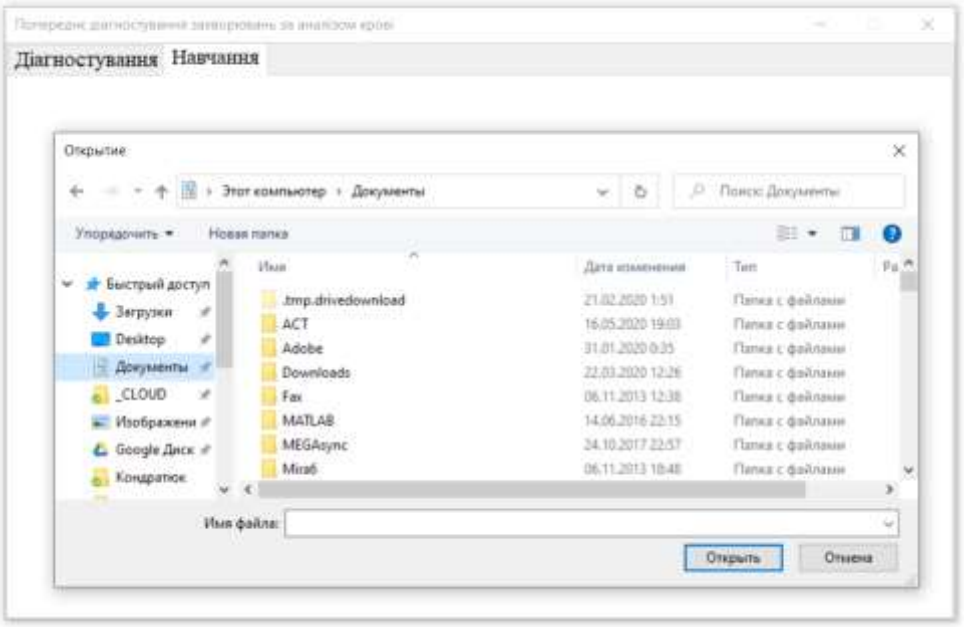


Рисунок 3.4 - Додаткове вікно навігатора для вказання місця знаходження потрібного файла на диску

Інформація про навчену нейромережу Кохонена зберігається у вигляді .txt файлу, який представляє собою масив чисел (матриця ваг нейронів мережі) розміром $m \times n$, де m=19 (де 17 - кількість входів нейромережі, на які подаються 15 параметрів аналізу крові, значення статі і віку пацієнта, а також значення артеріального тиску), а n - кількість нейронів у мережі Кохонена.

Інформація про навчальну вибірку також зберігається у вигляді .txt файлу, який являє собою масив чисел розміром $V \times (m+2)$, де до m додається 2 параметри: ідентифікатор відомого пацієнта та мітка класу хвороб, а V - обсяг навчальної вибірки, тобто кількість пацієнтів з відомими групами хвороб, на прикладах яких навчається мережа Кохонена.

Після проведення навчання мережа Кохонена готова до використання для попереднього діагностування пацієнтів і тому можна перейти на вкладку «Діагностування» та заповнити всі поля параметрами пацієнта (стать, вік) та його аналізу крові (як це показано на рис. 3.5).

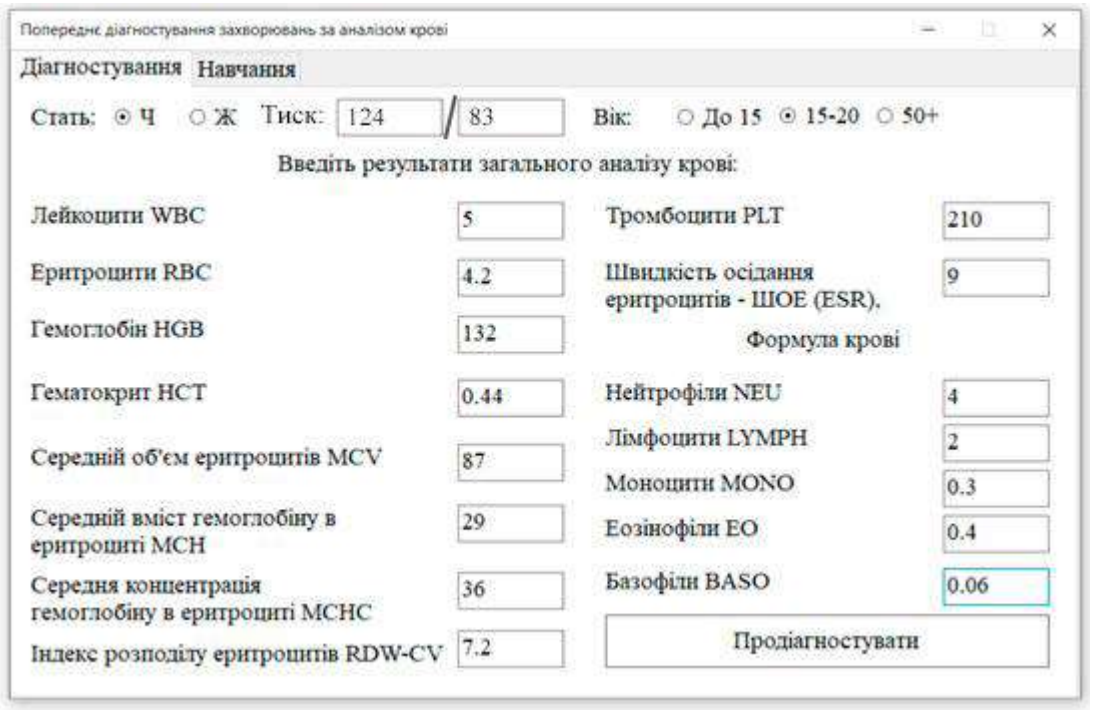


Рисунок 3.5 - Заповнення на вкладці «Діагностування» всіх полів параметрів пацієнта (стать, вік), його аналізу крові та параметрів тиску

Далі потрібно натиснути на кнопку «Продіагностувати» і з’явиться вікно з результатом попереднього діагностування пацієнта (див. рис. 3.6).

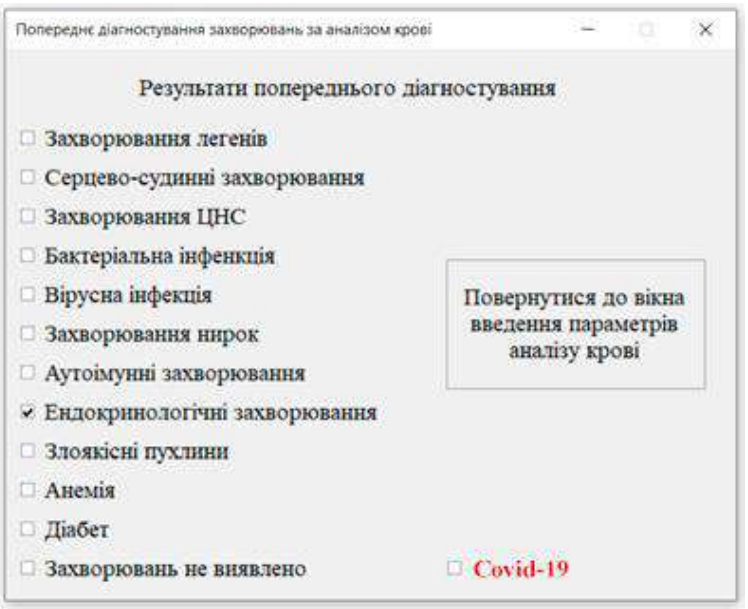


Рисунок 3.6 - Вікно модуля з результатом попереднього діагностування пацієнта

Із рис. 3.6 видно, що пацієнт має хворобу із групи «Ендокринологічні захворювання». Із цього вікна можна повернутись до вікна по рис. 3.5,

натиснувши кнопку «Повернутися до вікна введення параметрів аналізу крові».

Тестування розробленого програмного модуля показало його надійну роботу в різних режимах та повну відповідність поставленому завданню.

Проведемо експериментальні дослідження програмного модуля з метою перевірки його коректного функціонування та визначення достовірності діагностики у порівнянні з програмами-аналогами. Для проведення експериментів була використані на безкоштовні загальнодоступні бази даних показників аналізів крові та відповідних хвороб [14].

Проведемо тестування надійності програмного модуля попередньої діагностики пацієнта на основі нейронної мережі Кохонена проти аналогового програмного забезпечення. Для порівняння оберемо найпопулярнішу систему онлайн діагностування YourDiagnosis [14], яка базується на медичних алгоритмах. Для діагностування візьмемо 100 наборів вхідних значень та визначимо чи правильно вони були розпізнані. Результати тестування представлені у табл. 3.1.

Таблиця 3.1 – Результати тестування розробленого програмного модуля та програми-аналога YourDiagnosis

Програмний засіб	Кількість пацієнтів у тестовій вибірці	Кількість правильно діагностованих пацієнтів	Кількість невірно діагностованих пацієнтів	Достовір-ність діагностування пацієнтів
YourDiagnosis	100	84	16	84 %
Розроблений програмний модуль	100	92	8	92 %

Із табл. 3.1 видно, що розроблений програмний модуль має вищу достовірність попереднього діагностування пацієнтів (92%), ніж аналогічна програма (84%), а значить достовірність попереднього діагностування пацієнтів покращена на 8%, тобто мета роботи досягнута.

Тестування розробленого програмного модуля попередньої діагностики пацієнтів на основі нейронної мережі Кохонена показало його надійну працездатність. Програмний модуль повністю виконує поставлене завдання.

ВИСНОВКИ

У кваліфікаційній роботі магістра досліджений процес попереднього медичного діагностування пацієнтів за аналізом крові та артеріального тиску, розроблено програмний модуль попереднього діагностування пацієнтів на основі нейронної мережі Кохонена.

У першому розділі кваліфікаційної роботи було проведено аналіз предметної області для створення програмного модуля. У ході аналізу предметної області були проаналізовані сучасні процеси та методи діагностування пацієнтів, запропонований метод медичного діагностування, що використовує значення показників загального аналізу крові та артеріального тиску для визначення групи захворювань, до яких відноситься хвороба пацієнта.

У другому розділі кваліфікаційної роботи магістра було проведено аналіз процесу кластеризації, також обґрунтовано вибір методу кластеризації. Розроблено структуру, математичну модель, порядок функціонування нейронної мережі Кохонена та алгоритм функціонування програмного модуля попереднього діагностування.

У третьому розділі кваліфікаційної роботи обґрунтовано вибір мови та середовища програмування для розробки програмного модуля та створено діаграму класів для його функціонуванння, реалізовані всі основні модулі програми.

Було протестовано можливості програмного модуля попереднього діагностування пацієнтів на основі нейронної мережі Кохонена, а саме: завантаження даних та навчання мережі на їх основі, проведення тестового діагностування з вводом даних та отримання результатів, щодо груп хвороб тестового пацієнта. Всі функції працюють коректно. Також проведено тестування розробленого програмного модуля та програми-аналога YourDiagnosis. Розроблений програмний модуль має вищу достовірність попереднього діагностування пацієнтів (92%), ніж аналогічна програма (84%), а значить достовірність попереднього діагностування пацієнтів покращена на 8%.

Програмний модуль попереднього діагностування пацієнтів на основі

нейронної мережі Кохонена повністю придатний до використання і може використовуватися у медичній сфері з метою попередньої діагностики пацієнтів та економії часу лікарів і пацієнтів. Програмний модуль може бути вдосконалений при потребі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. В.О. Кондратюк, Л.А. Савицька, Н.В. Добровольська. Програмний модуль попереднього діагностування пацієнтів на основі нейронної мережі Кохонена // Інформаційні технології та комп`ютерна інженерія. м. Вінниця - 2023. №1, том 56.

2. Автоматизація медичних установ [Електронний ресурс]. Режим доступу: http://www.aksimed.com/clients/

3. Телемедицина [Електронний ресурс]. Режим доступу: http://uk.wikipedia.org/wikiТелемедицина - Назва з екрану

4. Алгоритми діагностування пацієнтів [Електронний ресурс]. Режим доступу: http://www.vidal.by/vracham/Informatsiya-dlya-spetsialistov/Nevrologiya-psihiatriya/Algoritmy-diagnostiki-i-vedeniya-patsientov/ - Назва з екрану

5. Аналізи крові [Електронний ресурс]. Режим доступу: http://medportal.ua/ - Назва з екрану

6. Загальний аналіз крові [Електронний ресурс]. Режим доступу: http://ukrhealth.net/zagalnij-analiz-krovi/ - Назва з екрану

7. Загальний аналіз крові: розшифровка і норми [Електронний ресурс]. Режим доступу: http://rheumatology.org.ua/blog/articles/1076/ - Назва з екрану

8. Сімахін В.О. Кластеризація медичних даних с помощью нейросетей / В.О. Сімахін, А.В. Кашінцев [Електронний ресурс]. Режим доступу: http://neurocomp.ua/

9. Автоматизація загального аналізу крові [Електронний ресурс]. Режим доступу: http://www.myshared.com/slide/114542/ - Назва з екрану

10. Автоматизація - хімічний аналіз [Електронний ресурс]. Режим доступу: http://www.ngpedia.com/id226550p1.html - Назва з екрану

11. Your Diagnosis. Your Free Online Medical Diagnosis & Symptoms Analysis Tool [Електронний ресурс]. Режим доступу: http://www.yourdiagnosis.com/ - Назва з екрану.

12. Diagnoz.ua. Іноваційний медичний центр [Електронний ресурс]. Режим доступу: http://www.diagnoz.ua/ - Назва з екрану.

13. Руденко О.Г. Штучні нейронні мережі / О.Г. Руденко, Є.В.

Бодянський – Харків, 2005. – 407 с.

14. Dostal P. Cluster analysis and neural network / P. Dostal, P.Pokorny [Електронний ресурс]. Режим доступу: http://dsp.vscht.cz/konference_matlab/ MATLAB08/ prispevky/025_dostal.pdf

15. Нейронна м е р е ж а Кохонена [Електронний ресурс]. Режим доступу: http://www.machinelearning.com/index.php?title=Нейронна_мережа_Кохонена – Назва з екрану.

16. Дмитрієва Т.В. Самоорганізуючі карти Кохонена / Т.В. Дмитрієва [Електронний ресурс]. Режим доступу: http://storage.library. opu.ua/online/conference/tez_45/1-8.pdf

17. Нейронні мережі на C#. Мережа Кохонена [Електронний ресурс]. Режим доступу: http://www.pvsm.com/net/7339 - Назва з екрану.

18. Нейронні мережі на C#. Мережа Кохонена [Електронний ресурс]. Режим доступу: http://habrahabr.com/post/143668/ – Назва з екрану.

19. Скільки в Україні медиків, лікарень та ФАПів URL: https://www.slovoidilo.ua/2021/05/25/infografika/suspilstvo/skilky-ukrayini-medykiv-likaren-ta-fapiv

20. Amazon Web Services Cookbook, Official AWS documentation, 2018 (https://docs.aws.amazon.com/en_us/opsworks/latest/userguide/cookbooks-101-basics.html)

21. Kubernetes Cookbook, Michael Hausenblas, Sébastien Goasguen, March 2018, (https://www.oreilly.com/library/view/kubernetes-cookbook/9781491979679/)

22. Kubernetes cluster-operator documentation, Google\Cloud Native Foundation, 2018 (https://kubernetes.io/docs/home/?path=users&persona=cluster-operator)

23. Google`s Site Reliability Engineering Workbook, O`Reilly (Betsy Beyer, Chris Jones, Jennifer Petoff, Niall Murphy), 2016 (https://landing.google.com/sre/sre-book/toc/index.html)

24. HERO Technical specifications, 2016 (http://www.mediaexcel.com/resources/brochures/Media_Excel_Datasheet.pdf)

25. Kogan J., Nicholas C., Teboulle M. Clustering Large and High Dimensional

Data (http://www/csee/umbc/edu/nicolas/clustering/tutorial.pdf)

26. Unsupervised learning Clustering, MIT October 2014 (http://www.mit.edu/~9.54/fall14/slides/Class13.pdf)

27. Jain A.K., Murty M.N., Flynn P.J. Data Clustering: A Review (http://www/csee/umbc/edu/nicolas/clustering/p264-jain.pdf)

28. Docker book, Conteinerization is new virtualization, James Turnbull, 2014 (https://www.amazon.com/Docker-Book-Containerization-new-virtualization-ebook/dp/B00LRROTI4)

29. Neural-network based clustering, Basel Abu-Jamous, Asoke K. Nandi, 2015 (https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118906545.ch14)

30. Neural-network toolbox for MATLAB, Howard Demuth Mark Beale, 1992 (http://cda.psych.uiuc.edu/matlab_pdf/nnet.pdf)

31. НЕЙРОННІ МЕРЕЖІ І ГЕНЕТИЧНІ АЛГОРИТМИ, Кононюк, Київ 2008 (http://fs.onu.edu.ua/clients/client11/web11/metod/imem/neyron.pdf)

32. Kohonen`s neural-networks, 2015 (https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_kohonen_selforganizing_feature_maps.htm)

33. Kohonen network, 2008 (http://www.scholarpedia.org/article/Kohonen_network)

34. Introduction to neural networks, MIT, 2008 (https://ocw.mit.edu/courses/brain-and-cognitive-sciences/9-641j-introduction-to-neural-networks-spring-2005/)

35. How neural networks think, MIT, 2017 (http://news.mit.edu/2017/how-neural-networks-think-0908)

36. Artificial neural networks, Lecture MIT, 2003 (https://ocw.mit.edu/courses/sloan-school-of-management/15-062-data-mining-spring-2003/lecture-notes/NeuralNet2002.pdf)

37. Штучні нейронні мережі, Руденко О.Г., Бодянський Є.В. 2002 (http://www.library.univ.kiev.ua/ukr/elcat/new/detail.php3?doc_id=1092072)

38. Клітково-нейронне моделювання, Обчислювальні мережі, Дніпро 2014, (https://www.researchgate.net/publication/322529917_Klitkovo-

nejronne_modeluvanna_Obcisluvalni_merezi_nejroniv)

39. Нейронні мережі, Тимощук В.П., Львів, 2011 (http://vlp.com.ua/node/6698)

40. Data Mining: Concepts and Techniques, Jiawei Han, 2000 (http://myweb.sabanciuniv.edu/rdehkharghani/files/2016/02/The-Morgan-Kaufmann-Series-in-Data-Management-Systems-Jiawei-Han-Micheline-Kamber-Jian-Pei-Data-Mining.-Concepts-and-Techniques-3rd-Edition-Morgan-Kaufmann-2011.pdf)

41. Data Mining: Practical Machine Learning Tools and Techniques, 1999 (ftp://ftp.ingv.it/pub/manuela.sbarra/Data%20Mining%20Practical%20Machine%20Learning%20Tools%20and%20Techniques%20-%20WEKA.pdf)

42. Data Mining and Analysis: Fundamental Concepts and Algorithms, 2014 (https://repo.palkeo.com/algo/information-retrieval/Data%20mining%20and%20analysis.pdf)

43. Data Mining: The Textbook Charu C. Aggarwal, 2015 (https://doc.lagout.org/Others/Data%20Mining/Data%20Mining_%20The%20Textbook%20%5BAggarwal%202015-04-14%5D.pdf)

44. Matlab Neural-network examples, Primoz Potocnik, 2012 (http://lab.fs.uni-lj.si/lasin/wp/IMIT_files/neural/NN-examples.pdf)

45. Neural networks in Matlab, Kian Rayaneh Sabz, 2008 (https://www.researchgate.net/publication/264742064_Neural_networks_in_Matlab)

46. Matlab deepdive in neural-networks, (https://www.mathworks.com/help/deeplearning/ref/learnk.html)

ДОДАТОК А

Загальний алгоритм роботи програмного модуля попереднього діагностування пацієнтів на основі нейронної мережі Кохонена

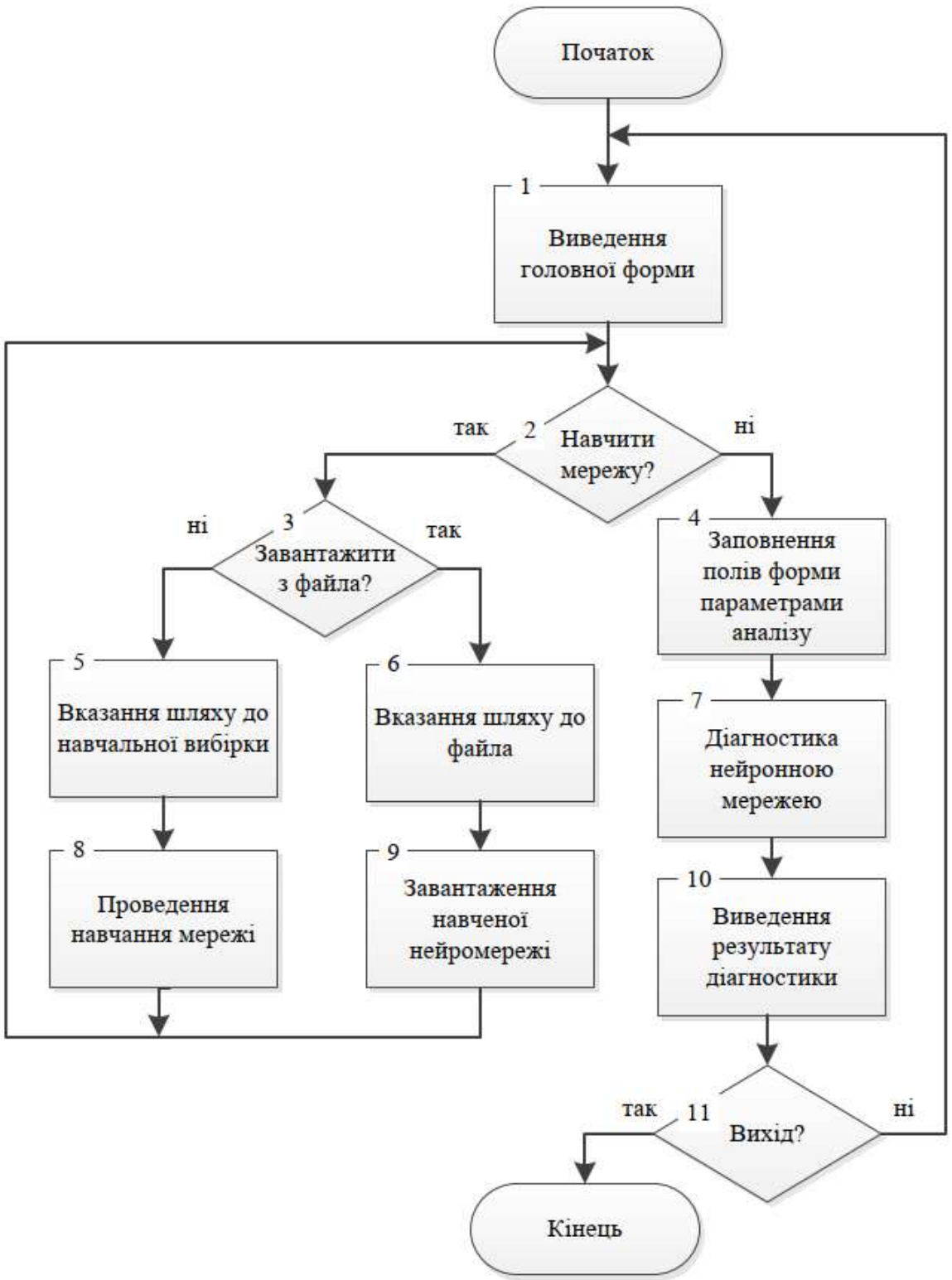


Рисунок А.1 - Загальний алгоритм роботи програмного модуля попереднього діагностування пацієнтів на основі нейронної мережі Кохонена

ДОДАТОК Б

Структура нейронної мережі Кохонена

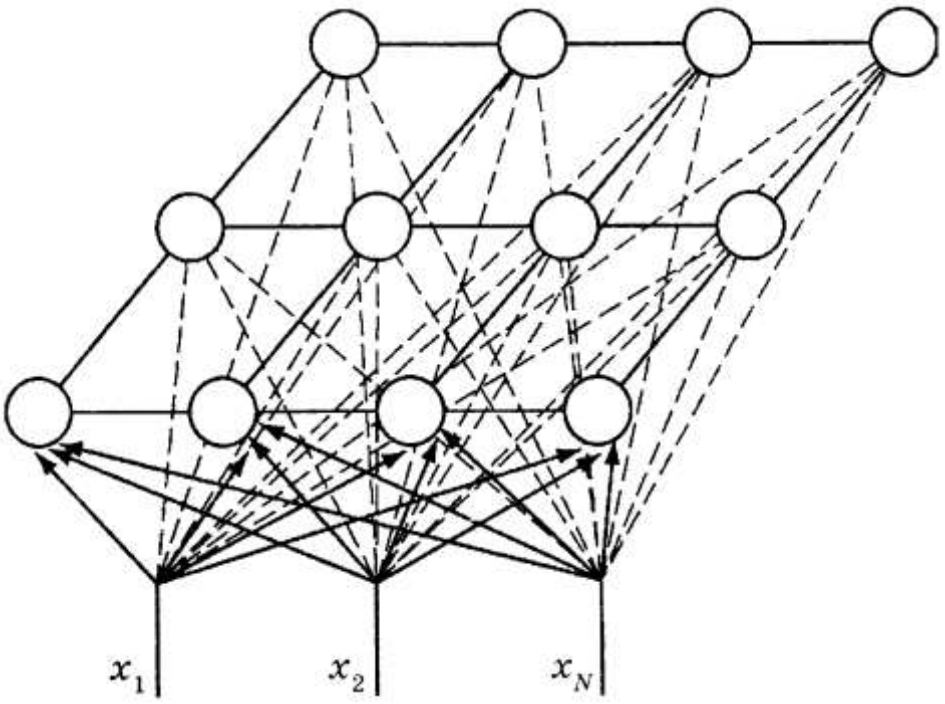


Рисунок Б.1 - Структура нейронної мережі Кохонена

ДОДАТОК В

Діаграма класів програмного модуля

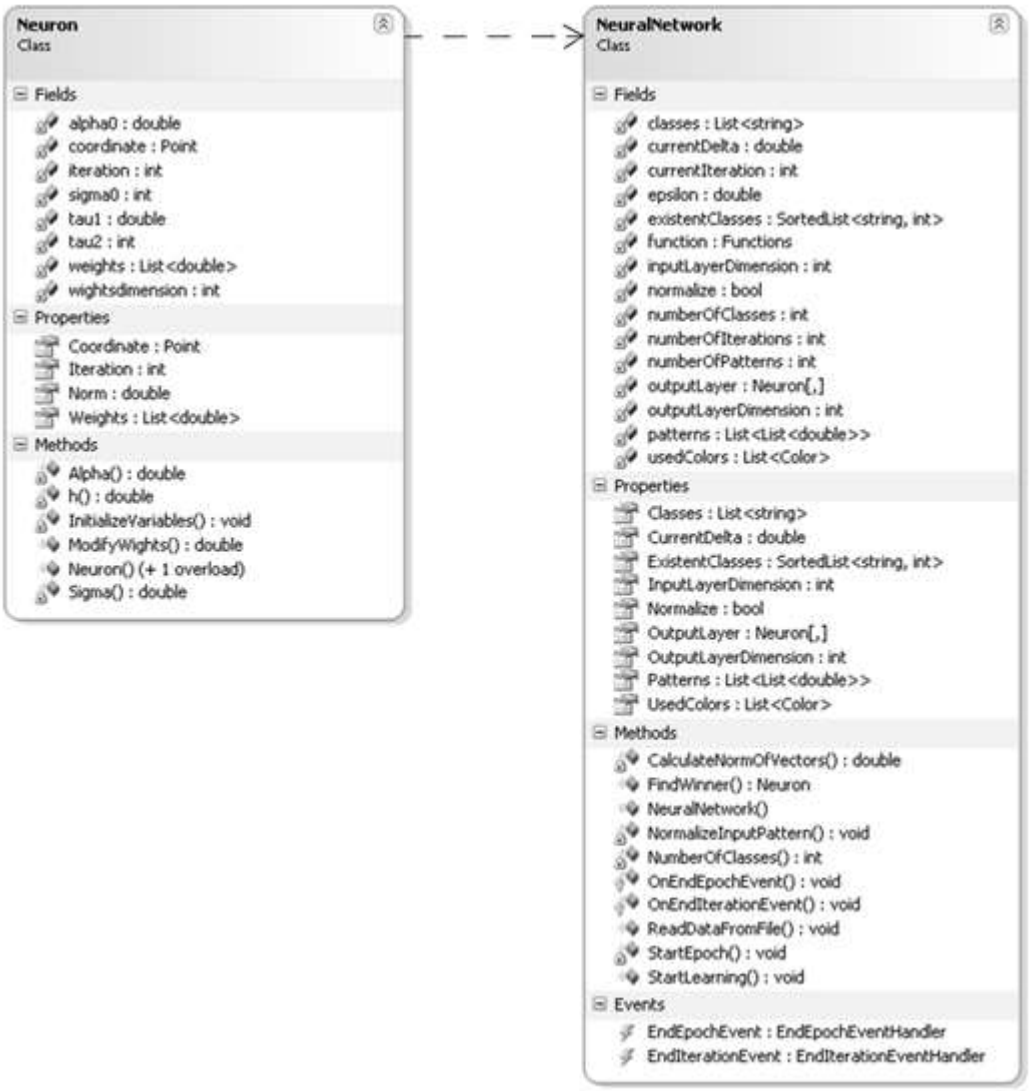


Рисунок В.1 - Діаграма класів програмного модуля

ДОДАТОК Г

Початкове вікно програмного модуля

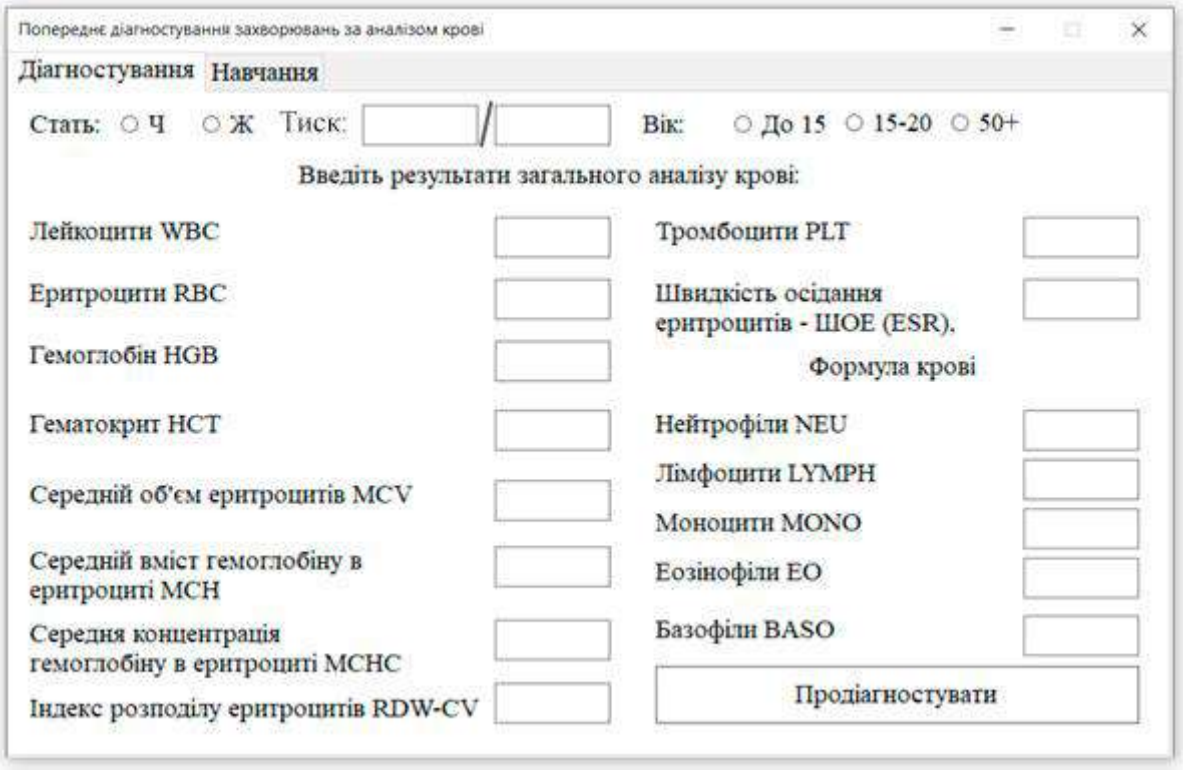


Рисунок Г.1 - Початкове вікно програмного модуля

ДОДАТОК Д

Результати роботи програми

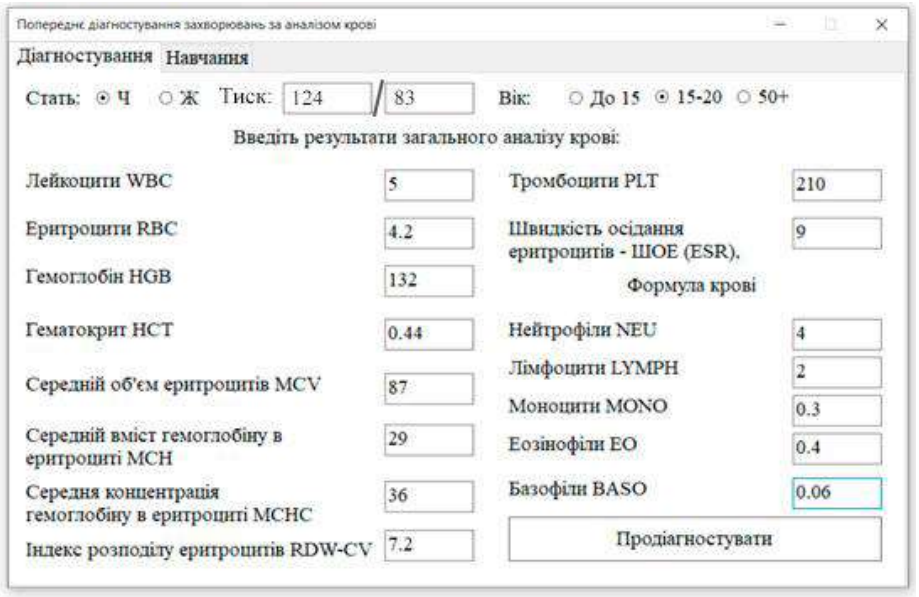


Рисунок Д.1 - Заповнення на вкладці «Діагностування» всіх полів параметрів пацієнта (стать, вік), його аналізу крові та параметрів тиску

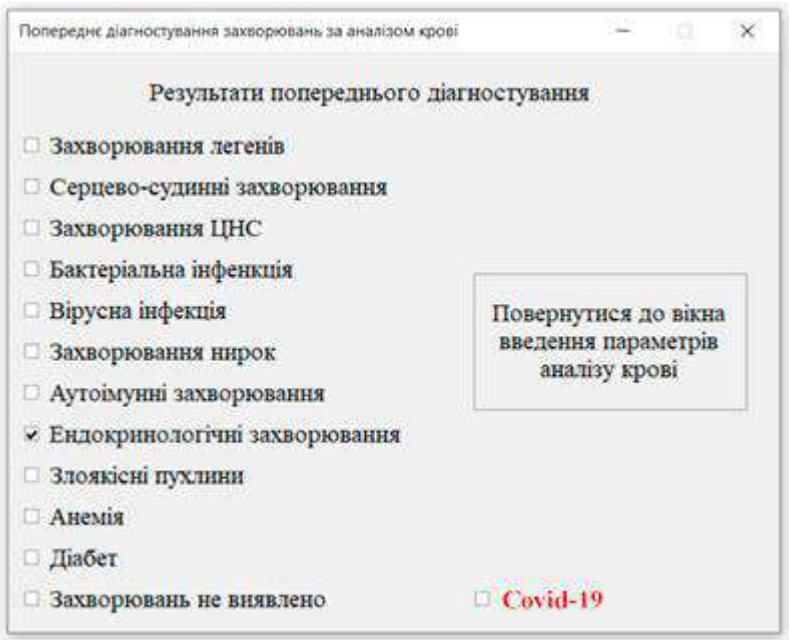


Рисунок Д.2 - Вікно модуля з результатом попереднього діагностування пацієнта

ДОДАТОК Е

Лістинг програми

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
namespace SOFMTest
{
  static class Program
  {
  /// <summary>
  /// The main  entry point for the application.
  /// </summary>
  [STAThread]
  static void Main()
  {
  Application.EnableVisualStyles();
  Application.SetCompatibleTextRenderingDefault(false);
  Application.Run(new  frmMain());
   }
  }
}
<?xml  version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
  <sectionGroup name="applicationSettings"
type="System.Configuration.ApplicationSettingsGroup,  System, Version=2.0.0.0,
Culture=neutral,  PublicKeyToken=b77a5c561934e089"  >
  <section name="SOFMTest.Properties.Settings"
type="System.Configuration.ClientSettingsSection,  System, Version=2.0.0.0,
Culture=neutral,  PublicKeyToken=b77a5c561934e089"  requirePermission="false" />
  </sectionGroup>
  <sectionGroup name="userSettings"
type="System.Configuration.UserSettingsGroup, System, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089"  >
  <section name="SOFMTest.Properties.Settings"
type="System.Configuration.ClientSettingsSection,  System, Version=2.0.0.0,
Culture=neutral,  PublicKeyToken=b77a5c561934e089"
allowExeDefinition="MachineToLocalUser"  requirePermission="false"  />

   </sectionGroup>
  </configSections>
  <applicationSettings>
  <SOFMTest.Properties.Settings>
                  <setting name="MaximalNumberOfNeurons"  serializeAs="String">
  <value>1000</value>
```

```
  </setting>
                  <setting name="MaximalNumberOfIterations"  serializeAs="String">
  <value>10000</value>
  </setting>
   </SOFMTest.Properties.Settings>
  </applicationSettings>
  <userSettings>
  <SOFMTest.Properties.Settings>
  <setting name="MinimalEpsilon"  serializeAs="String">
  <value>1E-09</value>
  </setting>
  <setting name="MaximalEpsilon"  serializeAs="String">
  <value>0.1</value>
  </setting>
   </SOFMTest.Properties.Settings>
  </userSettings>
</configuration>
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using SOFM;

          Application.DoEvents();
        }
        private void nn_EndIterationEvent(object sender, EventArgs e) {
          if (pbStatus.Value < pbStatus.Maximum) pbStatus.Value++;
        }
        private void lbPatterns_SelectedIndexChanged(object sender, EventArgs e) {
          inputDataChart.Series[1].Clear();
          outputDataChart.Series[1].Clear();

  inputDataChart.Series[1].Add(nn.Patterns[lbPatterns.SelectedIndex][0],
nn.Patterns[lbPatterns.SelectedIndex][1]);
  Neuron Winner = nn.FindWinner(nn.Patterns[lbPatterns.SelectedIndex]);
  outputDataChart.Series[1].Add(Winner.Weights[0], Winner.Weights[1]);
  sofmVisualizer.LightUpThePixel(Winner.Coordinate.X,
Winner.Coordinate.Y);
  }

        private void lblIONTLink_Click(object sender, EventArgs e) {
          System.Diagnostics.Process.Start("http://www.iont.ru");
        }
```

```
        private void chbVisualization_CheckedChanged(object sender, EventArgs e) {
          this.nn_EndEpochEvent(this, new EndEpochEventArgs());
        }

        private void AddPatternsToListBox()
        {
          lbPatterns.Items.Clear();
          string patternString;
          for (int i = 0; i < nn.Patterns.Count; i++)
          {
          patternString = "";
          patternString += nn.Classes[i]+" ";
          for (int j = 0; j < nn.InputLayerDimension; j++)
           patternString += nn.Patterns[i][j].ToString("g2") +" ";
          lbPatterns.Items.Add(patternString);
          }
        }

        private void SetError(TextBox tb, string s) {
          errorProvider.SetError(tb, s);

            bool errorOnOtherTextBoxControl = false;
            foreach (Control c in this.gbInputParams.Controls) {
              if (c is TextBox)
              {
                  if (errorProvider.GetError(c).Length!=0)
                    errorOnOtherTextBoxControl = true;
              }

          }
          if (!errorOnOtherTextBoxControl)
           btnLoadDataAndCreateNetwork.Enabled = true;
          else
              btnLoadDataAndCreateNetwork.Enabled = false;
        }

private void tb_Validating(object sender, CancelEventArgs e)
{
TextBox tb = (TextBox)sender;
if (tb.Text.Length != 0)
{
bool notValidSymbol = false;
char[] cArray = tb.Text.ToCharArray();
foreach (char c in cArray)
{
if (tb.Name == "tbEpsilon")
{
```

```
  if (!Char.IsDigit(c)  && c != '.') notValidSymbol  = true;
 }
 else
 {
 if (!Char.IsDigit(c))  notValidSymbol  = true;
 }
 }
 if (!notValidSymbol)
 {
 switch (tb.Name)
 {
 case "tbNumberOfCards":
 {
 if (Convert.ToInt32(tb.Text) <=
Properties.Settings.Default.MaximalNumberOfNeurons)  SetError(tb,"");
 else SetError(tb, "Maximal  number of neurons is - " +
Properties.Settings.Default.MaximalNumberOfNeurons.ToString());
 break;
 }
 case "tbIterationsNumber":
 {
 if (Convert.ToInt32(tb.Text) <=
Properties.Settings.Default.MaximalNumberOfIterations)  SetError(tb, "");
                                 else SetError(tb, "Maximal  number of iterations  - " +
Properties.Settings.Default.MaximalNumberOfIterations.ToString());
 break;

 }
 case "tbEpsilon":
 {
                                 double epsilon = Convert.ToDouble(tb.Text);
 if (epsilon <= Properties.Settings.Default.MaximalEpsilon  &&
epsilon >= Properties.Settings.Default.MinimalEpsilon) SetError(tb, "");
                                else SetError(tb, "Epsilon must lie  between " +
Properties.Settings.Default.MinimalEpsilon.ToString()  + "and " +
Properties.Settings.Default.MaximalEpsilon.ToString());
 break;
 }
 }
 }
 else
 {
  SetError(tb,"Incorrect input. Please try again.");
 }
 }

        }
```

```
namespace SOFM
{
  public enum Functions
  {
  Discrete,
  Gaus,
  MexicanHat,
  FrenchHat
  }
}
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
namespace SOFM
{
  public delegate  void EndEpochEventHandler(object  sender, EndEpochEventArgs e);
           public delegate  void EndIterationEventHandler(object  sender, EventArgs  e);

                public class
              NeuralNetwork
                              {
                    private  Neuron[,]
                          outputLayer;
          public Neuron[,] OutputLayer
          {
            get { return outputLayer;  }
            set { outputLayer = value;  }
          }
          private  int inputLayerDimension;
          private  int outputLayerDimension;
          private  int numberOfPatterns;
          private  List<List<double>> patterns;
          private  List<string> classes;
          private  SortedList<string, int> existentClasses;
          private  List<System.Drawing.Color>  usedColors;
          private  bool normalize;
          private  int numberOfIterations;
          private  int currentIteration;

          private  Functions
          function; private  double
          epsilon;
                        private  double
                        currentEpsilon;
  private  double CalculateNormOfVectors(List<double>  vector1, List<double> vector2)
  {
```

```
double value = 0;
for(int i=0; i<vector1.Count; i++)
value += Math.Pow((vector1[i] - vector2[i]), 2);
value = Math.Sqrt(value);
return value;
}

        private void NormalizeInputPattern(List<double> pattern) {
          double nn = 0;
          for (int i = 0; i < inputLayerDimension; i++)
          {
           nn += (pattern[i] * pattern[i]);
          }
          nn = Math.Sqrt(nn);
          for (int i = 0; i < inputLayerDimension; i++)
          {
          pattern[i] /= nn;

        }
        }
private void StartEpoch(List<double> pattern)
{
Neuron Winner = this.FindWinner(pattern);
currentEpsilon = 0;
for (int i = 0; i < outputLayerDimension; i++)
for (int j = 0; j < outputLayerDimension; j++)
{
currentEpsilon += outputLayer[i, j].ModifyWights(pattern, Winner.Coordinate,
currentIteration, function);
}
currentIteration++;
currentEpsilon = Math.Abs(currentEpsilon / (outputLayerDimension *
outputLayerDimension));
EndEpochEventArgs e = new EndEpochEventArgs();
OnEndEpochEvent(e);
}

        public bool Normalize
        {
          get { return normalize; }
          set { normalize = value;
        }
        }

        public List<List<double>> Patterns {
          get { return patterns; }
        }
```

```
public List<string>
Classes {
        get { return
          classes; }
}

public int InputLayerDimension
{
  get { return inputLayerDimension;
}
}

public int OutputLayerDimension
{
   get { return outputLayerDimension;
                                          }
}

public double CurrentDelta
{
  get { return currentEpsilon;
}
}

public SortedList<string, int>
ExistentClasses {
  get { return existentClasses; }
}

public List<System.Drawing.Color>  UsedColors
{
  get { return usedColors; }
}

private  int NumberOfClasses()
{
  existentClasses = new SortedList<string, int>();
  existentClasses.Add(classes[0], 1);
  int k = 0;
  int d = 2;
  for (int  i = 1; i < classes.Count; i++)
  {
  k=0;
  for (int  j = 0; j < existentClasses.Count; j++)
              if (existentClasses.IndexOfKey(classes[i])!=-1)  k++;
  if (k == 0)
  {
```

```
            existentClasses.Add(classes[i],d);
            d++;
            }
            }
            return existentClasses.Count;
          }

  public System.Drawing.Color[,] ColorSOFM()
  {
  System.Drawing.Color[,] colorMatrix = new
System.Drawing.Color[outputLayerDimension, outputLayerDimension];
  int numOfClasses = NumberOfClasses();
  List<System.Drawing.Color> goodColors = new
List<System.Drawing.Color>();

  goodColors.Add(System.Drawing.Color.Black);
  goodColors.Add(System.Drawing.Color.Red);
  goodColors.Add(System.Drawing.Color.Navy);
  goodColors.Add(System.Drawing.Color.Green);
  goodColors.Add(System.Drawing.Color.Yellow);
  usedColors = new List<System.Drawing.Color>(numOfClasses);
  usedColors.Add(goodColors[0]);
  int k = 0;
  int randomColor = 0;
  Random r = new Random();
  while (usedColors.Count != numOfClasses)
  {
  k = 0;
  randomColor = r.Next(goodColors.Count);
  foreach (System.Drawing.Color cl in usedColors)
  if (cl == goodColors[randomColor]) k++;
  if (k == 0) usedColors.Add(goodColors[randomColor]);
  }
  for (int i = 0; i < outputLayerDimension; i++)
  for (int j = 0; j < outputLayerDimension; j++)
  colorMatrix[i, j] =
System.Drawing.Color.FromKnownColor(System.Drawing.KnownColor.ButtonFa ce);

  for (int i = 0; i < patterns.Count; i++)
  {
  Neuron n = FindWinner(patterns[i]);
   colorMatrix[n.Coordinate.X,n.Coordinate.Y] =
usedColors[existentClasses[classes[i]]-1];
  }
  return colorMatrix;
  }
```

```
public NeuralNetwork(int m, int numberOfIterations, double epsilon, Functions f)
{
outputLayerDimension = m;
currentIteration = 1;
this.numberOfIterations = numberOfIterations;
function = f;
this.epsilon = epsilon;
currentEpsilon = 100;
}

public Neuron FindWinner(List<double> pattern)
{
List<double> norms = new List<double>(outputLayerDimension *
outputLayerDimension);
double D = 0;
Neuron Winner = outputLayer[0, 0];
double min = CalculateNormOfVectors(pattern, outputLayer[0,
0].Weights);
for (int i = 0; i < outputLayerDimension; i++)
for (int j = 0; j < outputLayerDimension; j++)
{
D = CalculateNormOfVectors(pattern, outputLayer[i, j].Weights);
if (D < min)
{
min = D;
Winner = outputLayer[i, j];
}
}
return Winner;
}

public void StartLearning()
{
int iterations = 0;
while (iterations<=numberOfIterations && currentEpsilon > epsilon)
{
List<List<double>> patternsToLearn = new List<List<double>>(numberOfPatterns);
foreach (List<double> pArray in patterns)
patternsToLearn.Add(pArray);
Random randomPattern = new Random();
List<double> pattern = new List<double>(inputLayerDimension);
for (int i = 0; i < numberOfPatterns; i++)
{
pattern = patternsToLearn[randomPattern.Next(numberOfPatterns -
i)];

                    StartEpoch(pattern);
```

```
                patternsToLearn.Remove(pattern);
              }
              iterations++;
              OnEndIterationEvent(new  EventArgs());
            }
public void ReadDataFromFile(string  inputDataFileName)
{
StreamReader sr = new StreamReader(inputDataFileName);
string line  = sr.ReadLine();
int k = 0;
for (int  i = 0; i < line.Length; i++)
{
if (line[i]  == ' ') k++;
}

inputLayerDimension = k;
int sigma0  = outputLayerDimension;

outputLayer = new Neuron[outputLayerDimension, outputLayerDimension];
Random r = new Random();
for (int  i = 0; i < outputLayerDimension;  i++)
for (int  j = 0; j < outputLayerDimension;  j++)
{
outputLayer[i,  j] = new Neuron(i,  j, sigma0);
outputLayer[i,  j].Weights  = new List<double>(inputLayerDimension);
for (k = 0; k < inputLayerDimension;  k++)
{
 outputLayer[i,  j].Weights.Add(r.NextDouble());
}
}

            k = 0;
            while  (line  != null)
            {
              line  = sr.ReadLine();
              k++;
            }
            patterns = new List<List<double>>(k);
            classes = new List<string>(k);
            numberOfPatterns = k;

                  List<double> pattern;

            sr = new StreamReader(inputDataFileName); line  = sr.ReadLine();

while  (line  != null)
{
```

```
 int startPos = 0;
 int endPos = 0;
 int j = 0;
 pattern = new List<double>(inputLayerDimension);
 for (int ind = 0; ind < line.Length; ind++)
 {
 if (line[ind] == ' ' && j != inputLayerDimension)
 {
 endPos = ind;
 pattern.Add(Convert.ToDouble(line.Substring(startPos, endPos - startPos)));
 startPos = ind + 1;
 j++;
 }
 if (j > inputLayerDimension) throw new
InvalidDataException("Wrong file format. Check input data file, and try again");
 }
 if (normalize) this.NormalizeInputPattern(pattern);
 patterns.Add(pattern);
 startPos = line.LastIndexOf(' ');
 classes.Add(line.Substring(startPos));
 line = sr.ReadLine();
  }
 }

        public event EndEpochEventHandler EndEpochEvent;
        public event EndIterationEventHandler EndIterationEvent;

        protected virtual void OnEndEpochEvent(EndEpochEventArgs e) {
           if (EndEpochEvent != null)
              EndEpochEvent(this, e);
        }

 protected virtual void OnEndIterationEvent(EventArgs e)
 {
     if (EndIterationEvent != null)
        EndIterationEvent(this, e);
  }
 }
} using System;
using System.Collections.Generic;

using System.Text;
using System.Drawing;

namespace SOFM
{
 public class Neuron
```

```
 {
 private List<double> weights;
 private Point coordinate;
 private int iteration;
 private int wightsdimension;
 private int sigma0;
 private double alpha0 = 0.1;
 private double tau1;
 private int tau2 = 1000;

 private double h(Point winnerCoordinate, Functions f)
 {
 double result = 0;
 double distance = 0;
 switch (f)
 {
 case Functions.Discrete:
 {
 distance = Math.Abs(this.Coordinate.X - winnerCoordinate.X) +
Math.Abs(this.Coordinate.Y - winnerCoordinate.Y);
 switch ((int)distance)
 {
 case 0: result = 1; break;
 case 1: result = 0.5f; break;
 case 2: result = 0.25f; break;
 case 3: result = 0.125f; break;
 }
 break;
 }
 case Functions.Gaus:

 {
 distance = Math.Sqrt(Math.Pow((winnerCoordinate.X -
coordinate.X), 2) + Math.Pow((winnerCoordinate.Y - coordinate.Y), 2));
 result = Math.Exp(-(distance * distance) /
(Math.Pow(Sigma(iteration), 2)));
 break;
 }
 case Functions.MexicanHat:
 {
 distance = Math.Sqrt(Math.Pow((winnerCoordinate.X -
coordinate.X), 2) + Math.Pow((winnerCoordinate.Y - coordinate.Y), 2));
 result = Math.Exp(-(distance * distance) /
Math.Pow(Sigma(iteration), 2)) * (1 - (2 / Math.Pow(Sigma(iteration), 2)) * (distance *
distance));
 break;
 }
```

```
case Functions.FrenchHat:
{
int a = 2;
distance = Math.Abs(this.Coordinate.X - winnerCoordinate.X) +
Math.Abs(this.Coordinate.Y - winnerCoordinate.Y);
if (distance <= a) result = 1;
else
if (distance < a && distance <= 3 * a) result = -1 / 3;
else
if (distance > 3 * a) result = 0;
break;
}
}
return result;
}

        private void InitializeVariables(int sigma0)
        {
          iteration = 1;
          this.sigma0 = sigma0;
          tau1 = 1000 / Math.Log(sigma0);
        }

        private double Alpha(int t)
        {
          double value = alpha0 * Math.Exp(-t/tau2);
          return value;
        }

        private double Sigma(int t)
        {
          double value = sigma0 * Math.Exp(-t/tau1);
          return value;
        }

        public List<double> Weights
        {
          get { return weights; }
          set
          {
          weights = value;
           wightsdimension = weights.Capacity;
          }
        }
        public Point Coordinate
        {
         get { return coordinate; }
```

```
        set { coordinate = value;  }
       }

       public int Iteration
       {
         get { return iteration;  }
       }

           public Neuron(int  x, int y, int sigma0)
       {
         coordinate.X = x;
         coordinate.Y = y;
         InitializeVariables(sigma0);
       }

       public Neuron(Point coordinate, int  sigma0)
       {
         this.coordinate = coordinate;
         InitializeVariables(sigma0);
       }
  public double ModifyWights(List<double>  pattern, Point winnerCoordinate,
int iteration,  Functions f)
  {
  double avgDelta  = 0;
  double modificationValue  =0;
  for (int  i = 0; i < wightsdimension;  i++)
  {
  modificationValue  = Alpha(iteration) * h(winnerCoordinate,  f) *
(pattern[i]  - weights[i]);
  weights[i]  += modificationValue;
  avgDelta  += modificationValue;
  }
  avgDelta  = avgDelta  / wightsdimension;
  return avgDelta;
  }

       public double Norm {
         get
           {
             double norm = 0;
            foreach (double d in weights)
             {
             norm += d;
             }
             norm = norm / this.wightsdimension;
             return norm}}}}
```