ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра комп'ютерних наук

Пояснювальна записка
до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**ПРОГНОЗУВАННЯ РОЗВИТКУ ЦУКРОВОГО ДІАБЕТУ ЗА ДОПОМОГОЮ МЕТОДІВ МАШИННОГО НАВЧАННЯ НА МОВІ ПРОГРАМУВАННЯ PYTHON**»

Виконала: студентка 4 курсу, групи КНД–41
спеціальності
122 Комп'ютерні науки
(шифр і назва спеціальності)
Карпік К.О.
(прізвище та ініціали)
Керівник Жебка В.В.
(прізвище та ініціали)
Рецензент_____
(прізвище та ініціали)

Київ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Комп'ютерний наук

Ступінь вищої освіти - «Бакалавр»

Спеціальність підготовки – 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Комп'ютерних наук

В.В. Вишневський

"____" _____2021 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТЦІ

КАРПІК КАТЕРИНІ ОЛЕГІВНІ

(прізвище, ім'я, по батькові)

1. Тема роботи: «Прогнозування розвитку цукрового діабету за допомогою методів машинного навчання на мові програмування Python»

Керівник роботи: Жебка Вікторія Вікторівна, к.т.н., доцент кафедри ІПЗ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «12»березня 2021року №65.

2. Строк подання студентом роботи «20» травня 2021 року

3. Вхідні дані до роботи

Алгоритми ансамблевих методів;

Науково-технічна література з питань, пов'язаних з програмним забезпеченням щодо прогнозування;

4. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити).

4.1 Аналіз алгоритмів машинного навчання.

4.2 Вибір найбільш ефективного алгоритму.

4.3 Створення моделі програми.

4.4 Розробка програми.

4.5 Тестування програми.

5. Перелік демонстраційного матеріалу (назва основних слайдів)

1. Актуальність роботи

2. Класифікація машинного навчання

3. Порівняння деяких методів машинного навчання

4. Алгоритм роботи програми

5. Інтерфейс розробленої програми

6. Висновки

7. Апробація

6. Дата видачі завдання «12» березня 2021року

КАЛЕДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	12.03 – 17.03	
2	Аналіз алгоритмів машинного навчання	18.03 – 28.03	
3	Створення моделі програми	29.03 – 04.03	
4	Розробка програми	05.03 – 22.04	
5	Тестування програми	23.04 – 06.05	
6	Вступ, висновки, реферат	07.05 – 12.05	
7	Розробка обов‘язкових демонстраційних креслень	13.05 – 19.05	
8	Подання роботи в деканат	20.05	

Студент _____ Карпік К.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Жебка В.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 82 с., 3 табл., 72 рис., 2 дод., 33 джерела.

МАШИННЕ НАВЧАННЯ, АНСАМБЛЕВІ МЕТОДИ, МЕТОД XGBOOST, PYTHON, ПРОГНОЗУВАННЯ, ЦУКРОВИЙ ДІАБЕТ

Об'єкт дослідження – прогнозування ймовірності діагностики цукрового діабету на ранніх стадіях за допомогою ансамблевого методу машинного навчання – XGBoost реалізованого на мові програмування Phyton.

Предмет роботи – програма для прогнозування поставлення діагнозу цукровий діабет на ранніх стадіях за допомогою методу машинного навчання.

Мета роботи – створення моделі для прогнозування ймовірності діагностики цукрового діабету на ранніх стадіях для досягнення швидкої та недорогої діагностики захворювання.

Завдання роботи – розробка програми для поставлення діагнозу цукровий діабет на ранніх стадіях за допомогою методу машинного навчання.

У роботі було проаналізовано алгоритми машинного навчання. Після їх порівняння було обрано ансамблевий метод – XGBoost для дослідження прогнозування цукрового діабету. Цей метод більш швидкий алгоритм порівняно з іншими алгоритмами завдяки його паралельним та розподіленим обчисленням.

Для кращого розуміння предмету роботи було створено модель програми для поставлення діагнозу цукровий діабет на ранніх стадіях.

На основі результатів виконаних досліджень та створеної моделі, було створено програму на мові програмування Phyton, використовуючи вже готові бібліотеки.

Дана програма може бути використана у всіх сферах діяльності, де необхідна попередня діагностика пацієнта з підозрою на цукровий діабет.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми та її актуальність: з розвитком рівня життя цукровий діабет стає все більш поширеним у повсякденному житті людей. У медицині діагноз цукрового діабету визначається відповідно до рівня глюкози в крові натще, толерантності до глюкози. Чим раніше буде поставлений діагноз, тим набагато легше контролювати та лікувати це захворювання.

Для пришвидшення постановки діагнозу на ранніх стадіях цукрового діабету необхідно розробити модель. Яка за допомогою методів машинного навчання буде прогнозувати діагноз на основі наданої інформації про користувача.

Ступінь вивчення проблеми: машинне навчання може допомогти людям скласти попереднє бачення про наявність цукрового діабету відповідно до їх щоденних даних фізичного обстеження. Для методу машинного навчання найважливішими проблемами є вибір дійсних ознак та правильний класифікатор.

Останнім часом для прогнозування діабету використовуються численні алгоритми, включаючи традиційний метод машинного навчання, такий як машина векторної підтримки (SVM), дерево рішень (DT), логістична регресія тощо. Polat і Günes(2007) відрізняли діабет використовуючи аналіз основних компонентів (PCA) та нейро-нечіткий висновок. Для того, щоб мати справу з наборами даних з великими розмірами, Razavian et al. (2015) побудували моделі прогнозування на основі логістичної регресії для різних видів прогнозування діабету 2 типу. Georga et al.(2013) зосередився на глюкозі та використовував регресію опорного вектора (SVR) для прогнозування діабету, що є проблемою багатоваріантної регресії. Методи машинного навчання широко використовуються для прогнозування діабету, і вони дають бажані результати.

Об'єктом дослідження є прогнозування ймовірності діагностики цукрового діабету на ранніх стадіях за допомогою ансамблевого методу машинного навчання – XGBoost реалізованого на мові програмування Phyton.

Предметом роботи є програма для прогнозування поставлення діагнозу цукровий діабет на ранніх стадіях за допомогою методу машинного навчання.

Метою роботи є створення моделі для прогнозування ймовірності діагностики цукрового діабету на ранніх стадіях для досягнення швидкої та недорогої діагностики захворювання.

Завданням роботи є розробка програми для поставлення діагнозу цукровий діабет на ранніх стадіях за допомогою методу машинного навчання.

Методика дослідження: перш за все, потрібно було ознайомитися з методами машинного навчання. Після їх порівняння вибрати метод, який найбільше підходить. Найкращим методом для вирішення даного завдання є ансамблевий метод машинного навчання – XGBoost. Створити модель програми для поставлення діагнозу цукровий діабет на ранніх стадіях, для кращого розуміння предмету роботи. Далі необхідно було створити програму згідно побудованої моделі на мові програмування Phyton, використовуючи вже готові бібліотеки.

Для користувача програма має виглядати, як тест. Після проходження якого, буде показано результат поставлення діагнозу цукровий діабет на ранніх стадіях.

Наукова новизна роботи полягає у створення програми для прогнозування поставлення діагнозу цукровий діабет на ранніх стадіях за допомогою ансамблевого методу машинного навчання – XGBoost реалізованого на мові програмування Phyton.

Практична значущість результатів: дана програма може бути використана у всіх сферах діяльності, де необхідна попередня діагностика пацієнта з підозрою на цукровий діабет.

1 МЕТОДИ МАШИННОГО НАВЧАННЯ

1.1 Машинне навчання

При народженні Штучного інтелекту (ШІ) в 1950-х рр., він визначався що будь-яка машина, здатна виконувати завдання, яке, як правило, вимагає людського інтелекту.

Системи ШІ, як правило, демонструють принаймні деякі з наступних рис: планування, навчання, міркування, вирішення проблем, подання знань, сприйняття, рух та маніпуляції та, меншою мірою, соціальний інтелект та креативність.

Поряд з машинним навчанням існують різні інші підходи, що використовуються для побудови систем ШІ, включаючи еволюційні обчислення, де алгоритми зазнають випадкових мутацій та комбінацій між поколіннями, намагаючись "розвинути" оптимальні рішення, та експертні системи, де комп'ютери програмуються з правилами, що дозволяють імітувати поведінку експерта-людини у певній галузі, наприклад, система автопілота, що летить на літаку.

Машинне навчання дозволяє комп’ютерам вирішувати завдання, які до цього часу виконували лише люди. Від водіння автомобілів до перекладу мови, машинне навчання викликає вибух у можливостях штучного інтелекту - допомагаючи програмному забезпеченню зрозуміти безладний та непередбачуваний реальний світ.

Машинне навчання – це процес навчання комп’ютерної системи, як робити точні прогнози, подаючи дані.

Ці прогнози можуть бути відповідями на запитання: чи є шматочок фрукта на фотографії бананом чи яблуком, виявлення людей, які перетинають дорогу перед безпілотним автомобілем, чи є електронний лист спамом або розпізнавання мови досить точно, щоб створити титри для відео YouTube.

Ключова відмінність від традиційного комп'ютерного програмного забезпечення полягає в тому, що людина не писала код, який вказує системі, як відрізнити банан від яблука.

Натомість модель машинного навчання навчили як надійно розрізняти плоди, навчаючи великому обсягу даних, в цьому випадку, ймовірно, величезна кількість позначених зображень, які містять банан або яблуко.

Отримуючи нові дані, ці програми навчаються, ростуть, змінюються та розвиваються самі по собі. Іншими словами, завдяки машинному навчанню комп’ютери знаходять глибоку інформацію, не кажучи, де шукати. Натомість вони роблять це, використовуючи алгоритми, які навчаються на основі даних в ітераційному процесі.

Хоча концепція машинного навчання існує довгий час, здатність автоматизувати застосування складних математичних обчислень до великих даних набирає обертів за останні кілька років.

На високому рівні машинне навчання – це здатність адаптуватися до нових даних самостійно та за допомогою ітерацій. В основному, додатки вчаться на попередніх обчисленнях і транзакціях і використовують “розпізнавання шаблонів” для отримання надійних та обґрунтованих результатів.

Машинне навчання, безсумнівно, є одним із найбільш захоплюючих підмножин штучного інтелекту. Він виконує завдання вивчення даних із конкретними входами до машини. Важливо зрозуміти, що змушує працювати машинне навчання, а отже, як його можна використовувати в майбутньому.

1.1.1 Процес роботи машинного навчання

Процес машинного навчання починається з введення навчальних даних у вибраний алгоритм. Навчальні дані – відомі або невідомі дані для розробки остаточного алгоритму машинного навчання. Тип введення навчальних даних впливає на алгоритм, тому важливим рішенням при навчанні моделі машинного навчання є те, на яких даних тренувати модель.

Після завершення навчання моделі модель оцінюється з використанням решти даних, які не використовувались під час навчання, допомагаючи оцінити її реальні показники.

Під час навчання моделі машинного навчання зазвичай для навчання використовується близько 60% набору даних. Ще 20% даних використовується для перевірки передбачень, зроблених моделлю, та коригування додаткових параметрів, що оптимізують результати роботи моделі. Ця точна настройка розроблена для підвищення точності прогнозування моделі при поданні з новими даними. Якщо передбачення не відповідає очікуваному, алгоритм повторно тренується кілька разів, поки не буде знайдено бажаний результат. Останні 20% набору даних потім використовуються для тестування результатів навченої та налаштованої моделі, щоб перевірити, чи прогнози моделі залишаються точними при поданні нових даних.

Це дозволяє алгоритму машинного навчання постійно вчитися самостійно і давати найбільш оптимальну відповідь, яка з часом поступово збільшується в точності.

1.1.2 Використання машинного навчання

Системи машинного навчання використовуються для того, щоб рекомендувати, який продукт ви, можливо, захочете придбати далі на Amazon або яке відео ви хочете переглянути на Netflix.

Кожен пошук Google використовує безліч систем машинного навчання, щоб зрозуміти мову вашого запиту до персоналізації результатів. Так само в системах розпізнавання спаму та фішингу Gmail використовуються навчальні моделі машинного навчання, щоб уникнути поштової скриньки від шкідливих повідомлень.

Одними з найбільш очевидних демонстрацій потужності машинного навчання є віртуальні помічники, такі як Apple Siri, Google Assistant та Microsoft Cortana.

Кожен з них значною мірою покладається на машинне навчання, щоб підтримати розпізнавання голосу та здатність розуміти природну мову.

Але крім цих дуже помітних проявів машинного навчання, системи починають знаходити застосування майже у кожній галузі. Ці експлуатації включають: комп'ютерний зір для безпілотних автомобілів, дронів та роботів-доставників; розпізнавання та синтез мови та мови для чат-ботів та робочих служб; розпізнавання обличчя для спостереження в таких країнах, як Китай; допомога рентгенологам у виявленні пухлин на рентгенівських променях, допомога дослідникам у виявленні генетичних послідовностей, пов’язаних із захворюваннями, та виявленні молекул, які можуть призвести до більш ефективних ліків у галузі охорони здоров’я; забезпечення прогнозного обслуговування в інфраструктурі шляхом аналізу даних датчиків IoT; підкріплення комп'ютерного бачення, яке робить можливим безготівковий супермаркет Amazon Go, пропонуючи досить точну транскрипцію та переклад мови для ділових зустрічей - список можна продовжувати і продовжувати.

1.2 Методи машинного навчання

Машинне навчання (Machine learning, ML) залежить від алгоритмів – рис. 1.1, щоб повідомити, які дії вживаються, а потім виробляють виведену функцію. Мета машинного навчання - працювати без допомоги людини, певною мірою ця допомога є необхідною. Якщо висловити це простою мовою, ви повинні навчити свій алгоритм, як він повинен працювати і що він повинен шукати. У майбутньому ми можемо побачити, як машини досягають справжньої самосвідомості та працюють незалежно від людських введених даних. Але наразі люди та дані будуть і надалі відігравати найважливішу роль у формуванні машинних прогнозів.

Алгоритми машинного навчання використовують обчислювальні методи для «вивчення» інформації безпосередньо з наявних даних. Ось чому важливо

вводити стільки релевантних даних, скільки їх доступно. Зі збільшенням кількості зразків алгоритм ML працює все ефективніше.

Існує два основних методи керування моделлю машинного навчання (рис. 1.2) – навчання з вчителем і без вчителя. Залежно від наявних даних та запитання, алгоритм буде навчений формувати результат за допомогою одного з цих методів. Різниця між ними полягає в тому, що під час навчання з вчителем використовується повний набір маркованих даних. При навчанні без вчителя набір даних надається без чітких вказівок щодо того, що з ним робити.

Рисунок 1.1 – Методи машинного навчання

Рисунок 1.2 – Класичні види машинного навчання

1.2.1 Навчання з вчителем

Техніка навчання з вчителем частіше використовується в машинному навчанні, оскільки вона має справу з простими завданнями і проста у реалізації. Введення даних позначаються відповіддю, яку повинен отримати алгоритм, що допомагає машині вибирати шаблони в майбутньому, краще диференціювати дані або робити прогнози.

Навчання з вчителем класифікується на дві категорії алгоритмів і ідеально підходить для задач, де є доступні контрольні точки.

- Класифікація: визначається як процес розпізнавання, розуміння та групування об'єктів та ідей за попередньо встановленими категоріями, тобто "підгрупи". За допомогою цих попередньо класифікованих наборів навчальних даних класифікація в програмах машинного навчання використовує широкий спектр алгоритмів для класифікації майбутніх наборів даних по відповідних категоріях. Алгоритми класифікації, що використовуються в машинному навчанні, використовують вхідні навчальні дані з метою прогнозування

ймовірності того, що наступні дані потраплять в одну з заздалегідь визначених категорій. Одне з найпоширеніших застосувань класифікації – це фільтрація електронних листів за “спамом” або “не спамом”, як використовується сучасними провідними постачальниками послуг електронної пошти. Проблема класифікації існує, коли вихідна змінна є певною категорією. Вона оцінює ймовірність настання події на основі одного або декількох даних.

Коротше кажучи, класифікація є формою "розпізнавання зразків". Тут алгоритми класифікації, застосовані до навчальних даних, знаходять однакову закономірність (подібні послідовності чисел, слова чи настрої тощо) у майбутніх наборах даних.

- Регресія: це метод навчання з учителем, який допомагає знайти кореляцію між змінними і дозволяє нам прогнозувати безперервну вихідну змінну на основі однієї або кількох змінних-предикторів. Він в основному використовується для прогнозування, моделювання часових рядів і визначення причинно-наслідкового зв'язку між змінними. У регресії ми будуємо графік між змінними, які найкраще відповідають заданим точкам даних, використовуючи цей графік, модель машинного навчання може робити прогнози щодо даних.

Простіше кажучи, регресія показує лінію або криву, яка проходить через всі крапки даних на графі цільової-предиктор таким чином, щоб вертикальна відстань між точками даних і лінією регресії було мінімальним. Відстань між точками даних і лінією говорить про те, чи зафіксувала модель сильну зв'язок чи ні. Проблема регресії існує, коли вихідна змінна є реальним значенням, яке коливається (тобто долари, вага, вимірювання). Вона допомагає передбачити (або пояснити) певне значення на основі набору попередніх даних.

У реальному світі існують різні сценарії, в яких нам потрібні деякі прогнози на майбутнє, такі як погодні умови, прогноз продажів, маркетингові тенденції та інші. У такому випадку нам потрібна якась технологія, яка може робити прогнози більш точно. Тому для такого випадку нам знадобиться регресійний аналіз, який є статистичним методом і використовується в машинному навчанні та науці про дані. Регресія оцінює взаємозв'язок між метою та незалежної змінної. Виконуючи

регресію, ми можемо впевнено визначити найбільш важливий фактор, найменш важливий фактор і те, як кожен фактор впливає на інші фактори.

Відмінність між регресією та класифікацією полягає в тому, що класифікація є завданням прогнозування дискретної мітки класу, тоді як регресія передбачає кількість.

Ми регулярно використовуємо навчання з вчителем, щоб навчити себе чи когось іншого новому завданню. Це як би тестування з ключем відповіді. Після того, як засвоїти завдання, цей прийом можна застосувати до подібних процесів та інформації.

1.2.2 Навчання без вчителя

У навчанні без вчителем модель машинного навчання навчається органічно, а не отримує набір даних з явними інструкціями. Потім вона намагається автоматично знайти структуру в вихідних даних за допомогою аналізу та інтерпретації. Хоча навчання з вчителем є найпростішим, ми не завжди маємо доступ до повних, ідеально позначених наборів даних для навчання алгоритму. Там, де навчання з вчителем має „правильну” відповідь, навчання без вчителя корисно в ситуаціях, коли аналітики (або насправді хто-небудь) задають питання, а алгоритм не має відповіді, або є більше, ніж одна відповідь.

Модель навчання без вчителя класифікується на три різні категорії алгоритмів, які групують дані на основі подібності або взаємозв'язку між змінними:

- Кластеризація: один із найпоширеніших методів дослідницького аналізу даних, який використовується для отримання уявлення про структуру даних. Це можна визначити як завдання ідентифікації підгруп у даних таким чином, що точки даних в одній і тій же підгрупі (кластері) дуже схожі, тоді як точки даних у різних кластерах дуже різні. Іншими словами, ми намагаємось знайти в даних однорідні підгрупи, щоб точки даних у кожному кластері були якомога подібнішими за мірою подібності. Рішення, який засіб схожості використовувати, залежить від конкретної програми. Кластерний аналіз може

бути здійснений на основі ознак, де ми намагаємось знайти підгрупи зразків на основі ознак, або на основі зразків, де ми намагаємось знайти підгрупи ознак на основі зразків. Кластеризація використовується при сегментації ринку; де ми намагаємось знайти споживачів, схожих один на одного, будь то з точки зору поведінки чи атрибутів, сегментації / стиснення зображень; де ми намагаємося згрупувати схожі регіони, кластеризувати документи за темами тощо. Найпопулярнішим методом кластеризації є метод K-means, де “K” представляє кількість кластерів, які дослідник даних вирішив створити. Кластеризація K-means має на меті розділити n спостережень на k кластерів, в яких кожне спостереження належить до кластера з найближчим середнім значенням, слугуючи прототипом кластера. Як правило, метод K-means призначає кожну точку даних найближчому з випадково створених центрів і повторно обчислює центр кожного кластера .

- Зменшення розмірності: Коли йдеться про дані з великими розмірами, часто корисно зменшити розмірність, проектуючи дані в підпростір нижчого розміру, який фіксує “сутність” даних. Це називається зменшенням розмірності. Висока розмірність може означати сотні, тисячі або навіть мільйони вхідних змінних. Менша кількість вхідних розмірів часто означає відповідно меншу кількість параметрів або простішу структуру в моделі машинного навчання, яка називається ступенем свободи. Модель із занадто великим ступенем свободи, швидше за все, перевершить набір навчальних даних, і тому може не мати ефективної роботи з новими даними. Бажано мати прості моделі, які добре узагальнюють, і, в свою чергу, вхідні дані з невеликою кількістю вхідних змінних. Це особливо справедливо для лінійних моделей, де кількість входів та ступені свободи моделі часто тісно пов'язані. Зменшення розмірності – це техніка підготовки даних, що виконується на даних до моделювання. Це може бути виконано після очищення даних та масштабування даних, а також перед навчанням прогностичної моделі. Це процес можна розділити на вибір об’єкта та вилучення об’єкта. Іншими словами, цей метод використовується для усунення найменш важливої інформації з набору даних, наприклад, непотрібних або зайвих

стовпців, рядків та пікселів, які не є важливими для аналізу. Чим більша кількість функцій чи інших фрагментів даних, тим важче стає працювати над конкретним питанням. Ось чому вчені-дослідники намагаються видалити недоречну або надлишкову інформацію з набору даних.

- Пошук правил асоціації: правила асоціації – це твердження "якщо-тоді", які допомагають показати ймовірність взаємозв'язку між елементами даних у великих наборах даних у різних типах баз даних. Видобуток правил асоціацій має ряд застосувань і широко використовується для виявлення кореляції продажів у даних про транзакції або в наборах медичних даних. У науці про дані правила асоціації використовуються для пошуку кореляцій та співіснувань між наборами даних. Вони ідеально використовуються для пояснення закономірностей у даних із, здавалося б, незалежних сховищ інформації, таких як реляційні бази даних та транзакційні бази даних. Акт використання правил асоціації іноді називають "видобутком правил асоціацій" або "майнінговими асоціаціями".

Це найпоширеніші методи машинного навчання. Однак із розвитком машинного навчання в дію вступають нові методи.

1.3 Нові техніки та методи машинного навчання

Ці методи машинного навчання набагато вдосконаленіші та складніші. Їх розробка є багатообіцяючою, оскільки все більше нових додатків стає можливим.

1.3.1 Глибоке навчання та нейронні мережі (Deep learning and Neural networks)

Глибоке навчання - це підмножина машинного навчання в галузі штучного інтелекту, яка може імітувати функцію обробки даних мозку людини і створювати подібні моделі, які мозок використовує для прийняття рішень. На відміну від алгоритмів, що базуються на завданнях, системи глибокого навчання вчаться на поданих даних - вони можуть вчитися на неструктурованих або немаркованих даних. Архітектури глибокого навчання, такі як глибокі нейронні мережі, мережі

вірувань та періодичні нейронні мережі, та згорткові нейронні мережі знайшли застосування в галузі комп'ютерного зору, розпізнавання аудіо / мовлення, машинного перекладу, фільтрації соціальних мереж, біоінформатики, дизайну ліків та багато іншого.

Завдання нейронних мереж та глибокого навчання полягає у захопленні нелінійних моделей даних шляхом додавання шарів параметрів до моделі. Іншими словами, можна думати про глибоке навчання як про вдосконалення традиційного машинного навчання, що складається з більшої кількості шарів, що дозволяють вищі рівні абстракції та покращених прогнозів з вхідних даних. Алгоритми глибокого навчання використовують нейронні мережі для пошуку асоціацій між набором входів і виходів. Отже, нейронні мережі складаються з вхідного, прихованого та вихідного шарів. Наприклад, зробимо дві фотографії, на одній із зображень кота та на собаці. Це наш внесок. Ця інформація передається між кількома мережевими рівнями через певну математичну функцію. Коли ця частина закінчена, ви отримуєте результат. Наприклад, інформація, що малюнок 1 містить собаку.

Незважаючи на це, методи глибокого навчання вимагають багато даних та великих обчислювальних потужностей. Для того, щоб ефективно запускати алгоритми глибокого навчання, нам потрібні дуже потужні комп'ютери, розширені графічними процесорами. Передбачається, що метод глибокого навчання буде рости дуже швидко, але до цих пір було встановлено, що, наприклад, він чудово працює з аналізом зображень та розпізнаванням обличь.

1.3.2 Навчання з підкріпленням (Reinforcement learning, RL)

Навчання з підкріпленням - це навчання моделей машинного навчання для прийняття послідовності рішень. Агент вчиться досягати мети в невизначеному, потенційно складному середовищі. При навчанні з підкріпленням штучний інтелект стикається з ігровою ситуацією. Комп’ютер використовує методи спроб і помилок, щоб знайти рішення проблеми. Щоб примусити машину робити те, що

хоче програміст, штучний інтелект отримує або винагороду, або покарання за вчинені ними дії. Його мета – максимізувати загальну винагороду.

Незважаючи на те, що програміст встановлює політику винагороди – тобто правила гри - він не дає моделі жодних підказок чи пропозицій щодо вирішення гри. Від моделі залежить, як виконати завдання, щоб максимізувати винагороду, починаючи від абсолютно випадкових випробувань і закінчуючи вишуканою тактикою та надлюдськими навичками. Використовуючи потужність пошуку та багато випробувань, навчання з підкріплення в даний час є найефективнішим способом натякнути на креативність машини. На відміну від людських істот, штучний інтелект може накопичувати досвід з тисяч паралельних ігор, якщо алгоритм навчання з підкріпленням працює на достатньо потужній комп’ютерній інфраструктурі. Навчання моделям, які керують автономними автомобілями, є чудовим прикладом можливого застосування навчання з підкріплення. У ідеальній ситуації комп’ютер не повинен отримувати інструкцій з керування автомобілем. Програміст уникав би підключення будь-чого, що пов'язано із завданням, і дозволяв машині вчитися на власних помилках.

Навчання з підкріпленням – це, без сумніву, передова технологія, яка має потенціал для перетворення нашого світу. Однак його потрібно використовувати не в кожному випадку. Тим не менше, навчання з підкріплення, здається, є найбільш вірогідним способом зробити машину творчою – оскільки пошук нових, інноваційних способів виконання її завдань насправді є творчістю.

Алгоритми навчання з підкріпленням охоче використовуються в іграх, наприклад, шахи або GO. Ми бачили одне з найбільш впізнаваних додатків RL ще в 2017 році, коли комп’ютерна програма Google під назвою AlphaGo обіграла найкращого гравця світу в GO, гру, яку багато хто вважає найскладнішою настільною грою у світі. AlphaGo має самовикладаючий Штучний інтелект та просто підсилює прогрес та потужність штучного інтелекту для вирішення дуже складних завдань гри GO. Як показує результат поєдинку - він працює належним чином.

1.3.3 Ансамблеві методи

У машинному навчанні, незалежно від того, стикаємось ми з класифікацією чи проблемою регресії, вибір моделі надзвичайно важливий, щоб мати шанс отримати хороші результати. Цей вибір може залежати від багатьох змінних задачі: кількості даних, розмірності простору, гіпотези розподілу та інші.

Низьке зміщення та низька дисперсія, хоча вони найчастіше змінюються в протилежних напрямках, є двома найбільш фундаментальними ознаками, які очікуються для моделі. Дійсно, щоб мати можливість "вирішити" проблему, ми хочемо, щоб наша модель мала достатньо ступенів свободи, щоб вирішити основну складність даних, з якими ми працюємо, але ми також хочемо, щоб вона мала не надто багато ступенів свободи, щоб уникнути високої дисперсії і бути більш надійним. Це добре відомий компроміс між зміщенням і дисперсією .

В ансамблевій теорії навчання ми називаємо моделі слабких учнів (або базові моделі), які можуть бути використані як будівельний матеріал для проектування більш складних моделей шляхом поєднання декількох з них. Здебільшого ці базові моделі працюють не так добре самі по собі, тому що вони мають високий зсув (наприклад, моделі з низьким ступенем свободи), або тому, що вони мають занадто велику дисперсію, щоб бути надійними (наприклад, моделі із високим ступенем свободи). Потім ідея ансамблевих методів полягає в тому, щоб спробувати зменшити зміщення та/або дисперсію таких слабких учнів, комбінуючи кілька з них разом, щоб створити сильного учня (або модель ансамблю), який досягає кращих результатів.

Для того, щоб створити ансамблевий метод навчання, нам спочатку потрібно вибрати наші базові моделі, які будуть агреговані. Більшу частину часу (у тому числі у добре відомих методах Bagging and Boosting) використовується єдиний базовий алгоритм навчання, щоб ми мали однорідних слабких учнів, які навчались різними способами. Отриману нами модель ансамблю називають «однорідною». Однак існують також деякі методи, що використовують різні типи

базових алгоритмів навчання: деякі неоднорідні слабкі учні потім об’єднуються в «модель неоднорідних ансамблів».

Важливим моментом є те, що наш вибір слабких учнів повинен узгоджуватися з тим, як ми узагальнюємо ці моделі. Якщо ми вибираємо базові моделі з низьким зміщенням, але великою дисперсією, це повинно бути з методом агрегування, який має тенденцію до зменшення дисперсії, тоді як якщо ми вибираємо базові моделі з низькою дисперсією, але з великим зміщенням, це повинен бути метод агрегування, який має тенденцію до зменшення зміщення.

Є три основні види мета-алгоритмів, які спрямовані на поєднання слабких учнів:

- bagging , який часто розглядає однорідних слабких учнів, навчає їх незалежно один від одного паралельно і поєднує, виконуючи якийсь детермінований процес усереднення;
- boosting, який часто розглядає однорідних слабких учнів, навчає їх послідовно дуже адаптивним способом (базова модель залежить від попередніх) та поєднує їх, дотримуючись детермінованої стратегії;
- stacking, який часто розглядає різнорідних слабких учнів, навчає їх паралельно та поєднує, навчаючи мета-моделі для виведення прогнозу на основі різних прогнозів слабких моделей.

1.3.4 Bagging

Паралельно використовуючи різні методи, ми підбираємо різних учнів, які розглядаються, незалежно один від одного, і, отже, можна навчати їх одночасно. Найвідомішим таким підходом є «bagging» (що означає «bootstrap агрегат», рис. 1.3), метою якого є створення моделі ансамблю, яка є більш надійною, ніж окремі моделі, що її складають.

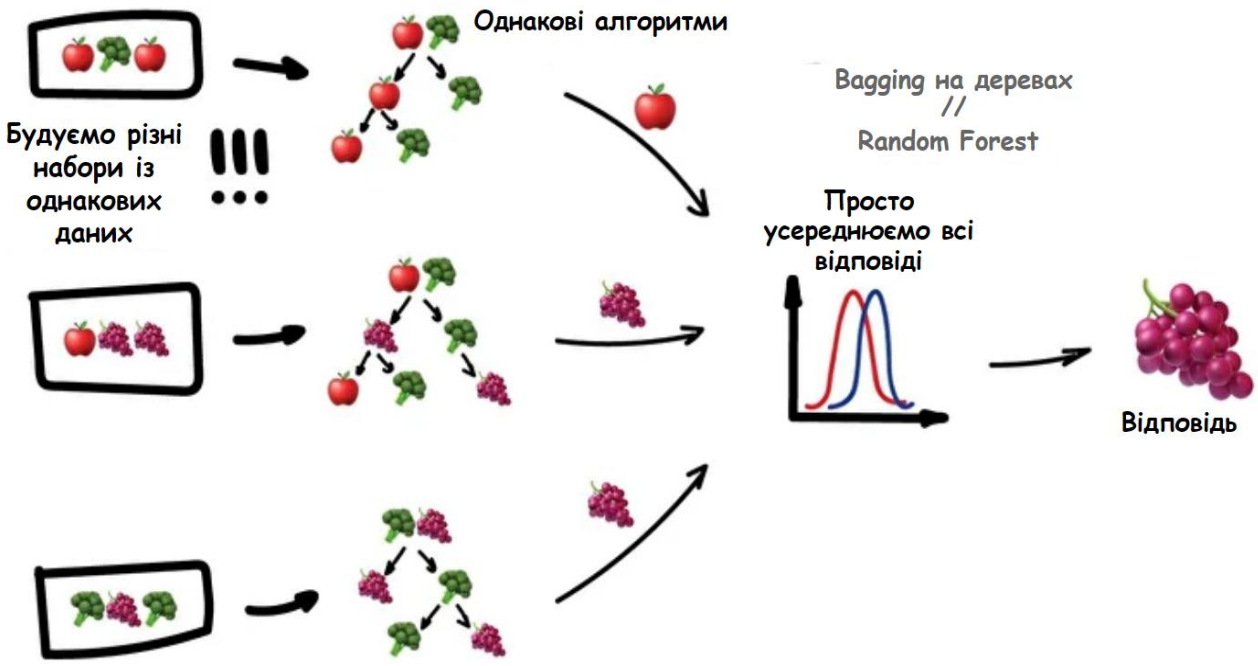


Рисунок 1.3 – Bagging

Почнемо з визначення bootstrapping. Цей статистичний прийом полягає у формуванні зразків розміру B (так званий bootstrap samples) з початкового набору даних розміром N шляхом випадкового малювання із заміною B спостережень.

За деякими припущеннями, ці зразки мають досить хороші статистичні властивості: у першому наближенні їх можна розглядати як отримані безпосередньо із справжнього базового (і часто невідомого) розподілу даних, так і незалежно один від одного. Отже, їх можна розглядати як репрезентативні та незалежні зразки справжнього розподілу даних. Гіпотеза, яку потрібно перевірити, щоб зробити це наближення дійсним, є подвійною. По-перше, розмір N початкового набору даних повинен бути достатньо великим, щоб охопити більшу частину складності базового розподілу, щоб вибірка з набору даних була гарним наближенням вибірки з реального розподілу (репрезентативність). По-друге, розмір N набору даних повинен бути достатньо великим порівняно з розміром B у bootstrap samples, щоб вибірки не були занадто співвіднесені (незалежність).

Bootstrap samples часто використовуються, наприклад, для оцінки дисперсії або довірчих інтервалів статистичних оцінювачів. За визначенням, статистичний оцінювач є функцією деяких спостережень і, отже, випадковою величиною з

дисперсією, яка походить від цих спостережень. Для того, щоб оцінити дисперсію такого оцінювача, нам потрібно оцінити його на декількох незалежних вибірках, взятих із розподілу інтересу. У більшості випадків розгляд справді незалежних зразків вимагав би занадто багато даних порівняно з наявною кількістю.

Під час навчання моделі, незалежно від того, маємо справу з класифікацією чи проблемою регресії, ми отримуємо функцію, яка приймає вхідні дані, повертає вихідні дані і яка визначається щодо набору даних навчання. Через теоретичну дисперсію навчального набору даних (ми нагадуємо, що набір даних - це спостережувана вибірка, що походить із справжнього невідомого базового розподілу), вбудована модель також зазнає змін: якщо б спостерігався інший набір даних, ми отримали б іншу модель .

Ідея bagging тоді проста: ми хочемо підібрати кілька незалежних моделей та визначити «усереднення» їх прогнозу, щоб отримати модель з меншою дисперсією. Однак на практиці ми не можемо підібрати повністю незалежні моделі, оскільки для цього потрібно занадто багато даних. Отже, ми покладаємось на добрі «приблизні властивості» bootstrap samples (репрезентативність та незалежність), щоб підібрати моделі, які є майже незалежними.

По-перше, ми створюємо декілька bootstrap samples, щоб кожен новий bootstrap sample діяв як ще один (майже) незалежний набір даних, отриманий із справжнього розподілу. Тоді ми можемо підібрати слабкого учня для кожного з цих зразків і, нарешті, згрупувати їх таким чином, що ми начебто «усереднимо» їх результати і, отже, отримаємо модель ансамблю з меншою дисперсією, ніж її компоненти. Грубо кажучи, оскільки bootstrap samples є приблизно незалежними та однаково розподіленими, так це і є вивчені базові моделі.

Існує кілька можливих способів об'єднання кількох моделей, паралельно встановлених. Для задачі регресії результати окремих моделей можуть бути буквально усереднені, щоб отримати результати моделі ансамблю. Що стосується проблеми класифікації, клас, виведений кожною моделлю, можна розглядати як голосування, а клас, який отримав більшість голосів, повертається моделлю ансамблю (це називається жорстким голосуванням). Ще для класифікаційної

задачі ми також можемо розглянути ймовірності кожного класу, поверненого всіма моделями, усереднити ці ймовірності та зберегти клас із найбільшою середньою ймовірністю (це називається м'яким голосуванням). Середні значення або голоси можуть бути простими або зваженими, якщо можна використовувати відповідні ваги.

Нарешті, ми можемо сказати, що однією з найбільших переваг bagging є те, що їх можна паралельно розподілити. Оскільки різні моделі встановлюються незалежно одна від одної, при необхідності можна використовувати інтенсивні методи паралелізації.

Random forests

Навчальні дерева – це дуже популярні базові моделі для ансамблевих методів. Сильних учнів, що складаються з декількох дерев, можна назвати «лісами». Дерева, що складають ліс, можна вибрати як мілкі (з невеликою глибиною), так і глибокі (безліч глибин, якщо вони не повністю виросли). Неглибокі дерева мають меншу дисперсію, але більший зсув і тоді буде кращим вибором для послідовних методів. Глибокі дерева, з іншого боку, мають низький зсув, але велику дисперсію, і тому є важливим вибором для методу bagging, який головним чином зосереджений на зменшенні дисперсії.

Random forest – підхід являє собою метод упаковки в bagging, де глибокі дерева, встановлені на bootstrap samples, об'єднується для отримання вихідного сигналу з більш низькою дисперсією.

1.3.5 Boosting

У послідовних методах різні комбіновані слабкі моделі більше не встановлюються незалежно одна від одної. Ідея полягає в тому, щоб підбирати моделі ітеративно так, щоб навчання моделі на даному етапі залежало від моделей, встановлених на попередніх етапах. «Boosting» є найвідомішим із цих підходів, і він створює модель ансамблю (рис. 1.4), яка, як правило, менш упереджена, ніж слабкі учні, які її складають.

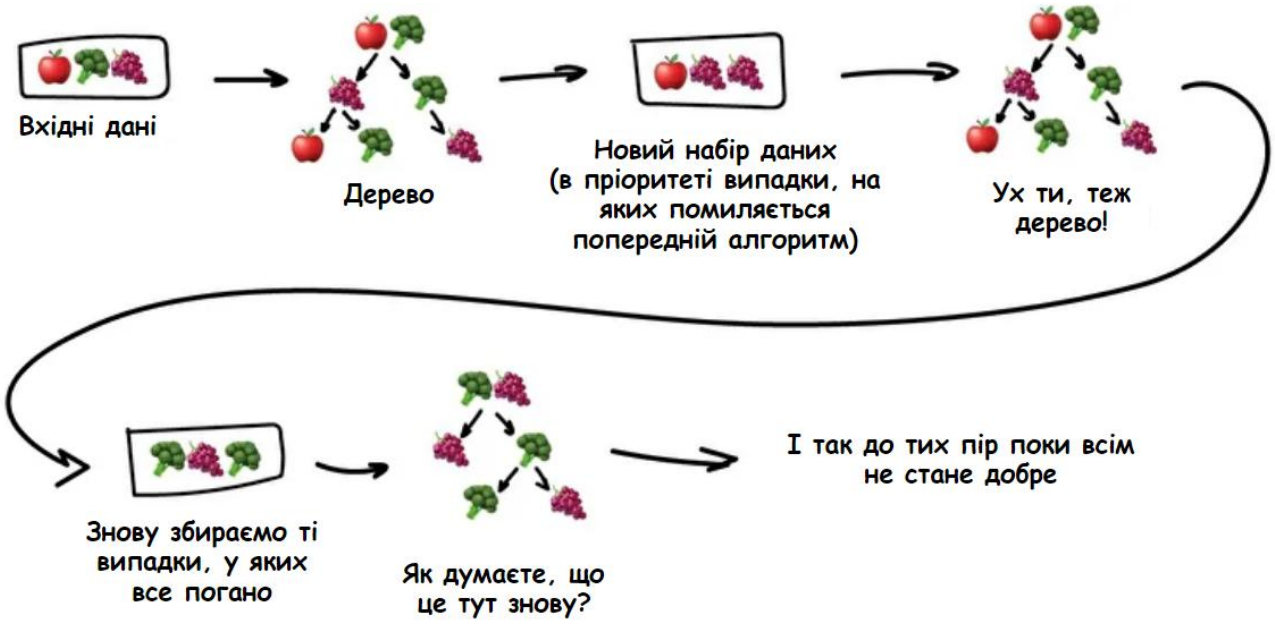


Рисунок 1.4 – Boosting

Методи boosting працюють у тому ж дусі, що і методи bagging: ми створюємо сімейство моделей, які об'єднуються, щоб отримати сильного учня, який має кращі результати. Однак, на відміну від bagging, які головним чином спрямовані на зменшення розбіжностей, boosting – це техніка, яка полягає у підгонці послідовно кількох слабких учнів дуже адаптивно: кожна модель у послідовності оснащена, надаючи більше значення спостереженням у наборі даних, які були погано оброблені попередні моделі в послідовності. Інтуїтивно кожна нова модель зосереджує свої зусилля на найскладніших спостереженнях щоб відповідати дотепер, так що в кінці процесу ми отримаємо сильного учня з нижчим зміщенням. Boosting, подібно до bagging, може використовуватися як для регресії, так і для проблем класифікації.

Будучи в основному зосереджено на зниження зміщення, базові моделі, які часто розглядаються для підвищення моделі з низькою дисперсією , але з високим зсувом. Наприклад, якщо ми хочемо використовувати дерева в якості базових моделей, ми більшу частину часу вибиратимемо неглибокі дерева рішень лише з кількома глибинами. Ще однією важливою причиною, яка мотивує використання моделей з низькою дисперсією, але з великим зміщенням як слабких учнів для підвищення, є те, що ці моделі, як правило, менш обчислювально придатні для

встановлення. Дійсно, оскільки обчислення, що відповідають різним моделям, неможливо робити паралельно (на відміну від bagging), може стати надто дорогим, щоб послідовно вмістити кілька складних моделей.

1.3.6 Stacking

Stacking включає комбінування прогнозів з декількох моделей машинного навчання на одному наборі даних, таких як bagging і boosting.

На відміну від bagging, у stacking моделі зазвичай відрізняються (наприклад, не всі дерева рішень) і вміщуються в одному наборі даних (наприклад, замість зразків навчального набору даних).

На відміну від boosting, у stacking одна модель використовується для того, щоб дізнатися, як найкраще поєднувати прогнози з внесених моделей (наприклад, замість послідовності моделей, що коригують прогнози попередніх моделей).

Архітектура моделі stacking (рис. 1.5) включає дві або більше базових моделей, які часто називають моделями рівня 0, і мета-моделі, яка поєднує в собі прогнози базових моделей, іменованих як модель рівня 1.

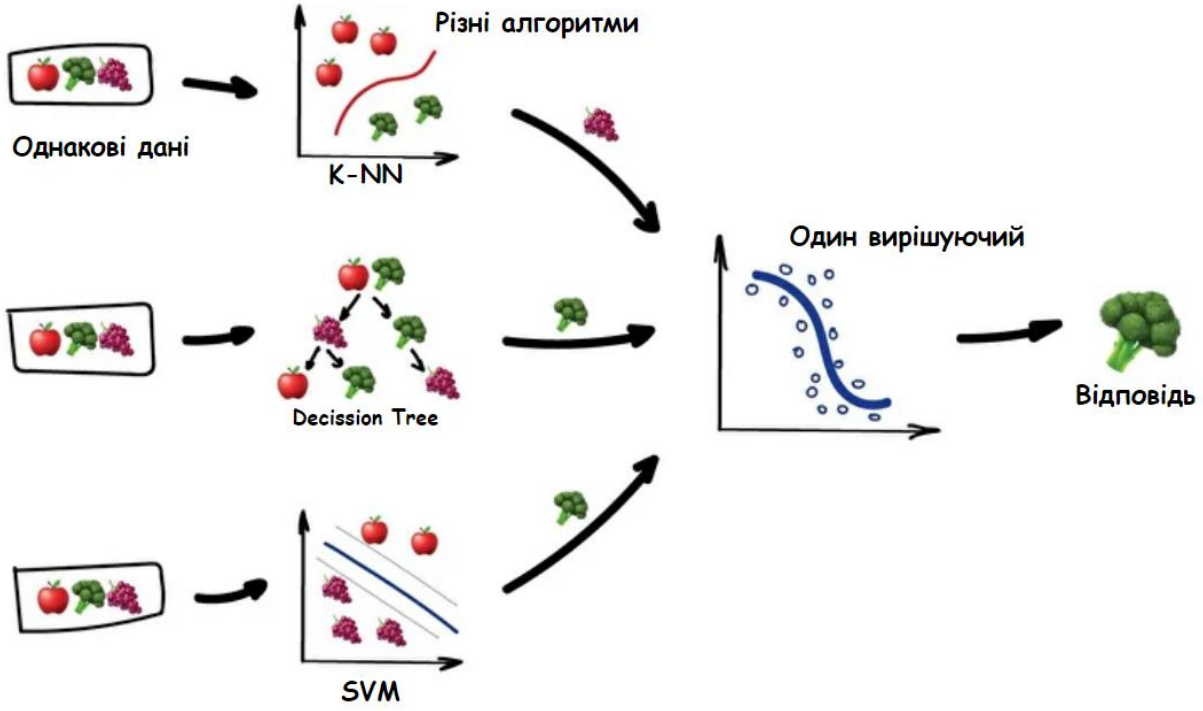


Рисунок 1.5 – Stacking

Моделі рівня 0 (базові моделі): моделі відповідають навчальним даним та передбачають прогнози.

Модель 1-го рівня (мета-модель): модель, яка дізнається, як найкраще поєднувати прогнози базових моделей.

Мета-модель навчається на основі прогнозів, зроблених базовими моделями на даних поза вибіркою. Тобто дані, що не використовуються для підготовки базових моделей, подаються до базових моделей, робляться прогнози, і ці прогнози, разом із очікуваними результатами, забезпечують вхідні та вихідні пари навчального набору даних, що використовуються для відповідності мета-моделі.

Вихідні дані базових моделей, що використовуються як вхідні дані для мета-моделі, можуть бути реальним значенням у разі регресії та значеннями ймовірності або мітками класів у разі класифікації.

Найпоширеніший підхід до підготовки набору навчальних даних для мета-моделі - це перехресне перевірка k-кратного переліку базових моделей, де прогнози використовуються як основа для навчального набору даних для мета-моделі.

Навчальні дані для мета-моделі можуть також включати вхідні дані до базових моделей, наприклад вхідні елементи навчальних даних. Це може забезпечити додатковий контекст для мета-моделі щодо того, як найкраще поєднувати прогнози з мета-моделі.

Після того, як навчальний набір даних підготовлений до мета-моделі, мета-модель може тренуватися ізольовано на цьому наборі даних, а базові моделі можна тренувати на всьому оригінальному навчальному наборі даних.

Stacking доречно застосовувати, коли кілька різних моделей машинного навчання мають навички роботи з набором даних, але володіють навичками по-різному. Іншим способом сказати це те, що прогнози, зроблені моделями, або помилки, передбачені моделями, є некорельованими або мають низьку кореляцію.

Stacking призначений для поліпшення ефективності моделювання, хоча не гарантовано призведе до покращення у всіх випадках.

Досягнення поліпшення результативності залежить від складності проблеми та того, чи достатньо вона представлена навчальними даними та достатньо складною, щоб можна було дізнатись більше, поєднуючи прогнози. Це також залежить від вибору базових моделей та того, чи достатньо вони вмілі та недостатньо некорельовані у своїх прогнозах (або помилках). Якщо базова модель працює так само добре або краще, ніж ансамбль stacking, замість неї слід використовувати базову модель, враховуючи її меншу складність (наприклад, простіше описати, навчити та підтримувати).

1.3.7 Порівняння Bagging та Boosting

Щоб використовувати Bagging або Boosting, потрібно вибрати базовий алгоритм для учнів. Наприклад, якщо ми виберемо дерево класифікації, Bagging and Boosting буде складатися з набору дерев, розмірів яких багато. Вони отримують N учнів, що генерують додаткові дані на етапі навчання. N нових наборів навчальних даних отримують шляхом випадкової вибірки із заміною вихідного набору. Шляхом вибірки із заміною деякі спостереження можуть повторюватися у кожному новому наборі навчальних даних.

У випадку Bagging, будь-який елемент має однакову ймовірність появи в новому наборі даних. Однак для Boosting спостереження зважуються, і тому деякі з них будуть брати участь у нових наборах частіше. Ці множинні набори використовуються для навчання одного і того ж алгоритму учнів, і тому виробляються різні класифікатори.

Хоча етап навчання паралельний для Bagging (тобто кожна модель будується незалежно), Boosting будує нового учня послідовно. В алгоритмах Boosting кожен класифікатор навчається на даних, беручи до уваги успіх попередніх класифікаторів. Після кожного тренувального кроку ваги перерозподіляються. Неправильно класифіковані дані збільшують свою вагу, щоб підкреслити найскладніші випадки. Таким чином, наступні учні будуть зосереджуватись на них під час навчання.

Для прогнозування класу нових даних нам потрібно лише застосувати N учнів, які навчаються до нових спостережень. У програмі Bagging результат отримується шляхом усереднення відповідей учнів, що навчаються (або більшості голосів). Однак Boosting призначає другий набір ваг, цього разу для N класифікаторів, щоб взяти середньозважене їх оцінок.

На етапі підготовки Boosting алгоритм розподіляє ваги для кожної отриманої моделі. Той, хто навчається з хорошим результатом класифікації за даними тренувань, отримуватиме більшу вагу, ніж поганий. Отже, оцінюючи нового учня, Boosting також повинен відстежувати помилки учнів. Деякі з методів Boosting включають додаткові умови утримувати або відкидати одного учня. Наприклад, в AdaBoost, найвідомішій, для підтримки моделі потрібна помилка менше 50%; в іншому випадку ітерація повторюється до досягнення учнем кращого, ніж випадкове відгадування.

Немає прямого переможця; це залежить від даних, моделювання та обставин. Bagging and Boosting зменшують дисперсію вашої єдиної оцінки, оскільки вони поєднують кілька оцінок з різних моделей. Отже, результатом може бути модель з більш високою стабільністю.

Якщо проблема полягає в тому, що одна модель отримує дуже низьку продуктивність, Bagging рідко отримує кращі упередження. Однак Boosting може створити комбіновану модель з меншими помилками, оскільки вона оптимізує переваги та зменшує підводні камені однієї моделі. Навпаки, якщо складність однієї моделі надмірно пристосована, то найкращим варіантом є Bagging. Boosting зі свого боку не допомагає уникнути надмірного припасування; насправді ця методика стикається з цією проблемою сама. З цієї причини функція Bagging діє частіше, ніж Boosting. Наглядно побачити порівняння можна в таблиці 1.1

Таблиця 1.1 – Порівняння Bagging та Boosting

Bagging	**Boosting**
Окремі дерева / моделі не залежать одне від одного	Окремі дерева не незалежні одне від одного

Продовження таблиця 1.1 – Порівняння Bagging та Boosting

Bagging	**Boosting**
Немає поняття вчитися один у одного в bagging. Кожна окрема модель / дерево отримуватиме зразки ознак / стовпців з усього навчального набору разом із вибіркою спостережень / рядків для цих ознак	Під час підсилення кожне з дерев буде вчитися на помилках попереднього дерева і намагатиметься мінімізувати залишкову помилку, оскільки воно продовжує рухатися вперед послідовно. Наприклад, у мене є boosting алгоритм з 3 послідовними моделями M1, M2 та M3. M2 намагається зменшити залишкову помилку, породжену M1, а M3 намагатиметься збити залишкову помилку, породжену M2, близько до нуля, забезпечуючи тим самим найкращу точність.
Швидкість навчання не використовується як гіперпараметр у bagging методах, оскільки дерева не залежать одне від одного	Швидкість навчання використовується як гіперпараметр у методах Boosting, оскільки кожне з дерев навчається на попередніх ітераціях
Приклади: Random Forest, Extra Tree algorithms.	Приклади: Gradient Boosting, ADABoosting, XGBoosting
Допомагає зменшити дисперсію	Допомагає зменшити як упередженість, так і дисперсію

1.4 XGBoost

XGBoost означає eXtreme Gradient Boosting. Він став популярним останнім часом і є домінуючим прикладним машинним навчанням та змаганнями Kaggle для структурованих даних через його масштабованість.

XGBoost - це розширення для дерев рішень з посиленим градієнтом (GBM) і спеціально розроблене для підвищення швидкості та продуктивності (Рисунок 1.6).

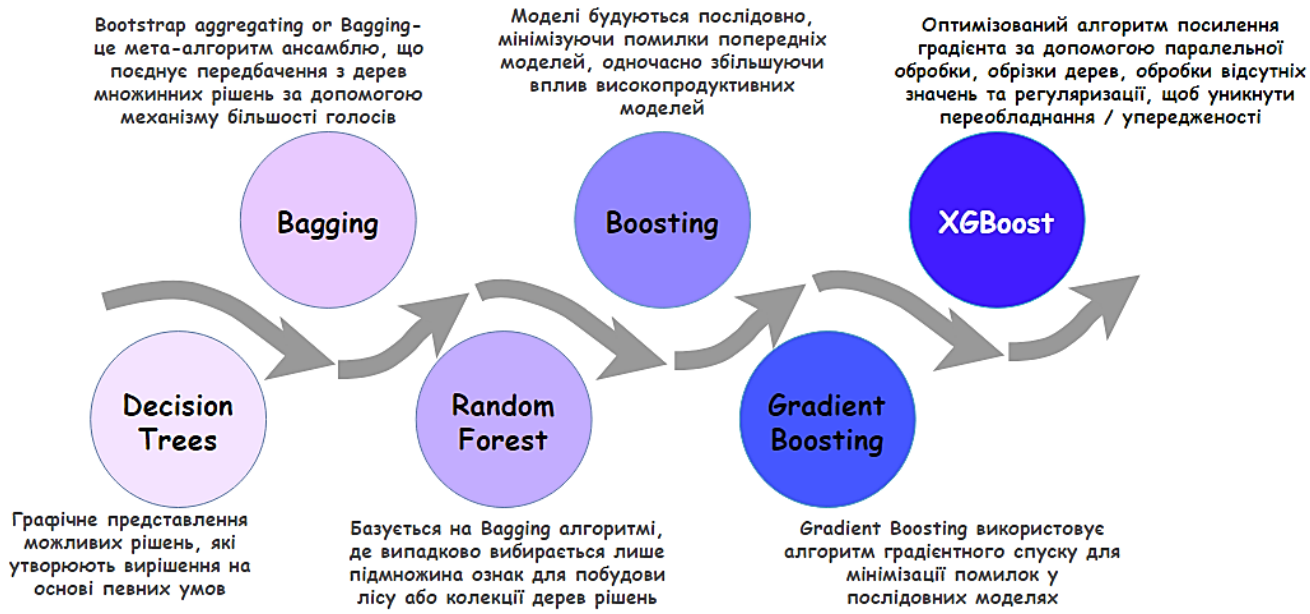


Рисунок 1.6 – Еволюція XGBoost алгоритму від Decision Trees

Особливості XGBoost:

а) Регулярне навчання: Термін регуляризації допомагає згладити остаточні вивчені ваги, щоб уникнути надмірного припасування. Регуляризована мета прагне вибрати модель із простими та передбачуваними функціями;

б) Підсилення градієнтного дерева: модель ансамблю дерева не може бути оптимізована за допомогою традиційних методів оптимізації в евклідовому просторі. Натомість модель навчається адитивним способом;

в) Зменшення вибірки для усадки та стовпців: Окрім регульованої мети, використовуються два додаткові методи для подальшого запобігання переобладнанню. Перша техніка – усадка, запроваджена Фрідманом. Усадка масштабує щойно додані ваги у коефіцієнт η після кожного кроку нарощування дерева. Подібно швидкості навчання при стохастичній оптимізації, усадка зменшує вплив кожного дерева і залишає простір для майбутніх дерев для вдосконалення моделі.

Особливості системи:

- Паралелізація побудови дерева, використовуючи всі ваші ядра процесора під час навчання. Збір статистичних даних для кожного стовпця може бути розпаралельований, що дає нам паралельний алгоритм для розбитого пошуку;
- Кешований доступ: XGBoost був розроблений для оптимального використання апаратного забезпечення. Це робиться шляхом виділення внутрішніх буферів у кожному потоці, де можна зберігати статистику градієнта;
- Блоки для позаядерних обчислень для дуже великих наборів даних, які не поміщаються в пам’ять;
- Розподілені обчислення для навчання дуже великих моделей з використанням кластера машин;
- Блок стовпців для паралельного навчання. Найбільш трудомісткою частиною навчання дерева є отримання даних у відсортованому порядку. З метою зменшення витрат на сортування дані зберігаються в блоках стовпців у відсортованому порядку у стислому форматі.

Цілі XGBoost:

а) Швидкість виконання: XGBoost майже завжди був швидшим за інші тестові реалізації від R, Python Spark, і це дійсно швидше порівняно з іншими алгоритмами;

б) Ефективність моделі: XGBoost домінує у структурованих або табличних наборах даних щодо проблем класифікаційного та регресійного моделювання.

Потужність XGBoost

Краса цього потужного алгоритму полягає у його масштабованості, яка забезпечує швидке навчання за допомогою паралельних та розподілених обчислень та забезпечує ефективне використання пам'яті.

XGBoost - це популярна реалізація посилення градієнта. Ось деякі особливості XGBoost, які роблять його таким цікавим:

- Регуляризація: XGBoost має можливість штрафувати складні моделі за допомогою регуляризації як L1, так і L2. Регуляризація допомагає запобігти переобладнанню;
- Обробка розріджених даних: Відсутні значення або кроки обробки даних, такі як одноразове кодування, роблять дані розрідженими. XGBoost включає алгоритм роздільного пошуку з урахуванням розрідженості для обробки різних типів розрідженості в даних;
- Зважений ескіз квантилів: Більшість існуючих алгоритмів, заснованих на дереві, можуть знайти точки розділення, коли точки даних мають однакову вагу (за допомогою алгоритму квантильного ескізу). Однак вони не обладнані для обробки зважених даних. XGBoost має алгоритм розподіленого зваженого квантильного ескізу для ефективної обробки зважених даних;
- Структура блоків для паралельного навчання: для швидшого обчислення XGBoost може використовувати кілька ядер на центральному процесорі. Це можливо завдяки структурі блоків у його системі проектування. Дані сортуються та зберігаються в одиницях пам'яті, які називаються блоками. На відміну від інших алгоритмів, це дозволяє повторно використовувати макет даних шляхом подальших ітерацій, замість того, щоб обчислювати їх знову;
- Поінформованість про кеш: У XGBoost для отримання статистики градієнта за індексом рядків потрібен неперервний доступ до пам'яті. Отже,

XGBoost був розроблений для оптимального використання апаратного забезпечення. Це робиться шляхом виділення внутрішніх буферів у кожному потоці, де можна зберігати статистику градієнта;

- Неосновні обчислення: Ця функція оптимізує доступний простір на диску та максимізує його використання при обробці величезних наборів даних, які не поміщаються в пам'ять.

1.5 Постановка бакалаврського дослідження

Таблиця 1.2 – Переваги та недоліки методів машинного навчання

Алгоритм	Концепція	Переваги	Недоліки
Навчання з вчителем			
Logistic regression	Якщо ви хочете класифікувати такі дані, як виявлення належності об'єкта до категорії або якщо ви знаєте, яка ймовірність події станеться, використовуйте логістичну регресію.	Швидко тренуватися і прогнозувати. Добре для невеликих проблем з класифікаційними даними. Легко зрозуміти	Не дуже точно. Не використовуйте для нелінійних даних. Не гнучкий для адаптації до складних даних.
Навчання без вчителя			
K-Means	Алгоритм K- Means означає кластеризацію даних шляхом поділу даних на групи однакової дисперсії, мінімізуючи критерій, відомий як інерція або сума кластерів	Швидше, ніж ієрархічна кластеризація. Якщо вибрано оптимальне значення k, тоді алгоритм працює	Не працює з категоріальними значеннями. Не працює добре, коли кластери перекриваються. Не працює добре

	у кластері. Цей алгоритм вимагає вказати кількість кластерів.	дуже добре. K-кластери можна обчислювати партіями для поліпшення продуктивності.	з нерегулярними даними, наприклад кластери різної щільності або коли точки даних розподілені.

Продовження таблиці 1.2 – Переваги та недоліки методів машинного навчання

Алгоритм	**Концепція**	**Переваги**	**Недоліки**
PCA	Якщо ви використовуєте дані з великою кількістю функцій (розмірів), тоді ви можете використовувати PCA для зменшення можливостей для спрощення даних.	Може пришвидшити обчислення. Допомагає нам у візуалізації даних із великими розмірами. Простий для розуміння та використання.	Споживає пам’ять і вимагає завантаження всіх даних у пам’ять. Потрібен великий набір матриці кореляції для зберігання кореляцій.
Навчання з підкріпленням			
Q-learning	Алгоритм Q-навчання оцінює дію, яка допомагає агенту у виборі наступної дії. Коли алгоритм Q-lerning використовує дію, він використовує цільову політику для вибору	Спроби знайти оптимальну політику. Гнучкий і адаптується, коли приймаються різні рішення. На	Може потрапити в пастку в місцевих мінімумах. Може бути повільнішим за SARSA. Якщо

	найкращої (оптимальної) дії. Q-навчання приймає швидкість навчання (від 0 до 1), що вказує, наскільки швидко ми хочемо, щоб алгоритм навчався,	відміну від методів Монте-Карло, йому не потрібно чекати до кінця, щоб оновити функцію корисності.	доступна велика винагорода, то Q-навчання викликає винагороду, навіть якщо вона дорога.

Продовження таблиці 1.2 – Переваги та недоліки методів машинного навчання

Алгоритм	**Концепція**	**Переваги**	**Недоліки**
Q-learning	Коефіцієнт знижки (від 0 до 1), що вказує на важливість майбутніх винагород. Q-learning оцінює коефіцієнт знижки та винагороду.		У реальному житті можна було б віддати перевагу консервативному підходу.
Державна дія - винагорода - державна дія (SARSA)	В алгоритмі агент (машина) починається з перебування в стані. Він виконує дію і отримує винагороду. Потім агент переходить у наступний стан, виконує дію, а потім оновлює попереднє значення дії. Коли SARSA вибирає наступну дію приймає швидкість навчання (від 0 до 1), вказуючи,	Покращує поточну політику, в якій приймаються рішення. Спроби знайти найкращу політику. Алгоритм SARSA консервативний і уникає небезпечного оптимального шляху.	Не змінює політику та отримує нагороди під час навчання, що може виявитись повільним процесом.

	наскільки швидко ми хочемо, щоб алгоритм навчався, коефіцієнт знижки (від 0 до 1), що вказує на важливість майбутніх винагород та початкових умов.		

Продовження таблиці 1.2 – Переваги та недоліки методів машинного навчання

Алгоритм	Концепція	Переваги	Недоліки
Глибоке навчання та нейронні мереж			
Neural Network	Якщо ви хочете побудувати розпізнавання мови, безпілотний автомобіль, тоді ви можете використовувати алгоритм нейронної мережі. Він натхненний біологічною нейронною мережею мозку.	Дуже точний. Велика кількість параметрів для підвищення передбачуваності. Може вирішувати складні нелінійні, класифікаційні та глибокі проблеми навчання.	Дуже повільно тренуватися та прогнозувати. Потрібен великий обсяг даних. Можна розглядати як чорний ящик. Вони обчислювально дорогі.
Ансамблеві методи			
Random Forest	Якщо у вас великий набір даних, і прогнозування даних базується на кількох рішеннях, тоді використовуйте	Висока точність. Хороша відправна точка для вирішення проблеми. Гнучкий і може	Повільний на тренуванні. Переобладнання. Не підходить для невеликих зразків. Невеликі

	випадковий ліс.	добре відповідати різноманітним даним. Швидке виконання. Простий у використанні.	зміни в моделі змін даних про навчання. Іноді занадто просто для дуже складних проблем.

Продовження таблиці 1.2 – Переваги та недоліки методів машинного навчання

Алгоритм	**Концепція**	**Переваги**	**Недоліки**
Random Forest	За допомогою випадкового лісу ви можете розділити дані, передати їх багатьом деревам рішень, об'єднати кілька дерев у ліс і використати більшість голосів, щоб знайти найкраще можливе рішення. Прикладом може бути пошук найкращого продавця телевізійної марки на наступний рік на основі різних категорій, таких як ціна, телевізор, проданий минулого року, гарантія, розмір екрану тощо.	Корисно для проблем регресії та класифікації. Може моделювати відсутні значення. Він має високі показники.	

Провівши аналіз та дослідження роботи методів машинного навчання, для подальшого тестування було обрано наступні методи: Logistic Regression, Random Forest, Gradient Boosting, XGBoost. Як можна побачити на Рисунку 1.7 метод XGBoost має найкращим поєднанням «Prediction Power – Training Time», порівняно з іншими алгоритмами. Тому для подальшого дослідження було обрано метод XGBoost.

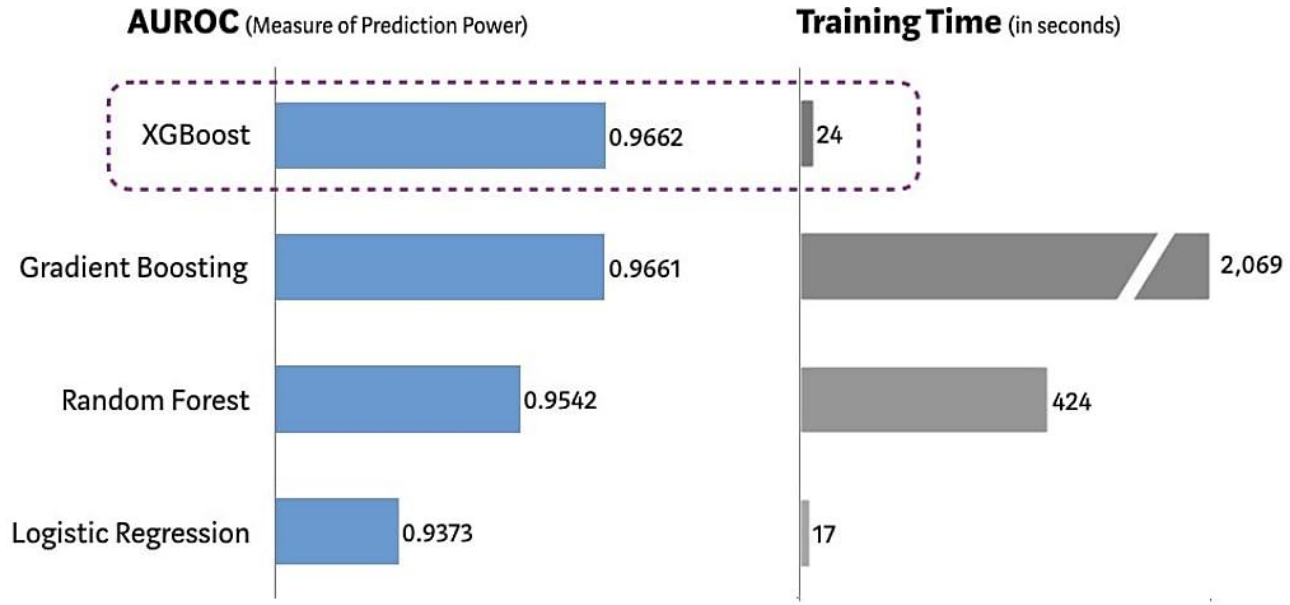


Рисунок 1.7 – Порівняння деяких методів машинного навчання

XGBoost – це більш швидкий алгоритм порівняно з іншими алгоритмами завдяки його паралельним та розподіленим обчисленням. XGBoost розроблений як з глибокими міркуваннями з точки зору оптимізації систем, так і принципів машинного навчання. Метою цієї бібліотеки є розширення граничних обчислювальних значень машин, щоб забезпечити масштабовану, портативну та точну бібліотеку.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Цукровий діабет

Цукровий діабет (ЦД) – одне з найчастіше діагностованих захворювань у світі. За даними Міжнародної федерації діабету, у 2030 році більше 439 мільйонів людей матимуть діагноз – цукровий діабет. Приблизно 2-5 мільйонів пацієнтів щороку втрачають життя через ЦД. Незважаючи на високу поширеність захворювання, досі не запропоновано жодного ефективного методу зменшення його частоти, хоча в даний час для лікування та контролю захворювання застосовуються різні методи.

Майже всі продукти які ми з'їдаємо, розщеплюється на цукор (так званий глюкозою) і викидається в кров. Якщо рівень глюкози в крові підвищується, це сигналізує про те, що підшлункова залоза виділяє інсулін. Він діє як ключ для проникнення цукру в кров у клітини вашого організму для використання в якості енергії.

Якщо у вас ЦД, то це означає що інсулін у вашому організмі виробляється у не достатній кількості або він не використовується організмом, так, як слід. Коли недостатньо інсуліну або клітини перестають реагувати на інсулін , занадто багато цукру в крові залишається в крові.

Люди, які страждають на цукровий діабет, мають високу ймовірність таких захворювань, як хвороби серця, нирок, інсульт, проблеми з очима, пошкодження нервів тощо. Як повідомляється, він є четвертою причиною смерті у більшості людських суспільств.

2.1.1 Типи цукрового діабету

Існує три основних типи ЦД: I тип, II тип та гестаційний діабет (ЦД під час вагітності).

Вважається, що ЦД I типу викликаний аутоімунною реакцією (організм атакує себе помилково), яка зупиняє ваше тіло від вироблення інсуліну. Приблизно 5-10% людей, які страждають на ЦД, страждають на тип I. Симптоми ЦД I типу часто розвиваються швидко. Зазвичай діагностується у дітей та молоді. Якщо у вас ЦД I типу, вам потрібно кожного дня приймати інсулін, щоб вижити. В даний час ніхто не знає, як попередити ЦД I типу.

При ЦД II типу ваше тіло погано вживає інсулін і не може підтримувати рівень цукру в крові на нормальному рівні. Близько 90-95% людей з ЦД страждають на тип II. Він розвивається протягом багатьох років і зазвичай діагностується у дорослих. Ви можете не помітити жодних симптомів, тому важливо пройти тест на рівень цукру в крові, якщо ви ризикуєте . ЦД II типу можна запобігти або відкласти за допомогою здорового способу життя, такого як схуднення, вживання здорової їжі та активність.

Гестаційний діабет розвивається у вагітних жінок, які ніколи не хворіли на ЦД. Якщо у вас гестаційний діабет, ваша дитина може мати більший ризик проблем зі здоров'ям. Гестаційний діабет зазвичай зникає після народження дитини, але збільшує ризик розвитку ЦД II типу в подальшому житті. Ваша дитина частіше страждає ожирінням як дитина, так і підліток, і більш схильний до розвитку ЦД II типу в подальшому житті.

2.1.2 Фактори, які впливають на появу цукрового діабету

Фактори ризику ЦД I типу не такі чіткі, як переддіабету та ЦД II типу. Відомі фактори ризику включають:

- Сімейна історія : Присутність батьків, брата або сестри з діабетом II типу.
- Вік: Ви можете захворіти на ЦД I типу в будь-якому віці, але частіше він розвинеться, коли ви дитина, підліток або молодий дорослий.

В даний час ніхто не знає, як попередити ЦД I типу.

Ви ризикуєте розвинути ЦД II типу, якщо:

- маєте переддіабет;

- маєте надмірну вагу;
- вік від 45 років;
- маєте батьків, брата або сестру, хворих на ЦД II типу;
- фізично активні менше 3 разів на тиждень;
- мали коли-небудь гестаційний діабет.

Ви можете запобігти або затримати ЦД II типу за допомогою простих, перевірених змін способу життя, таких як схуднення, якщо у вас надмірна вага, раціональне харчування та постійні заняття спортом.

2.1.3 Симптоми цукрового діабету

Якщо у вас є будь-який із наведених нижче симптомів ЦД, зверніться до лікаря щодо тестування рівня цукру в крові:

- сеча багато, часто вночі;
- відчуваєте сильну спрагу;
- схуднули, не намагаючись;
- постійно відчуваєте сильний голод;
- маєте розмитість зору;
- у вас німіють або поколюють руки або ноги;
- відчуваєте себе дуже втомленими;
- маєте дуже суху шкіру;
- маєте виразки, які заживають повільно;
- маєте більше інфекцій, ніж зазвичай.

Ще немає ліків від ЦД, але схуднення, вживання здорової їжі, активність та рання діагностика ЦД дійсно можуть допомогти.

2.2 Важливість машинного навчання в охороні здоров‘я

Охорона здоров’я – це завжди велике питання для будь-якої нації, і це завжди складне завдання. Найкраще показує охорону здоров'я нації - це стан

мешканців, які там проживають. Поліпшення системи охорони здоров’я може безпосередньо призвести до економічного зростання, оскільки здорова людина може виявитись великим надбанням для нації та може ефективно вести діяльність серед робочої сили, ніж будь-яка нездорова людина. Охорона здоров’я - це об’єднання та інтеграція всіх заходів, які можна вжити для вдосконалення системи охорони здоров’я. Охорона здоров’я - це профілактика, діагностика та лікування. Поліпшення охорони здоров’я повинно бути першочерговим завданням, над яким слід роздумати. Використання технологій для вдосконалення охорони здоров’я виявилось дуже вигідним.

Сучасна практика в лікарні полягає у зборі необхідної інформації для діагностики цукрового діабету за допомогою різних тестів, і на основі діагнозу надається відповідне лікування. Big Data Analytics відіграє значну роль у галузях охорони здоров’я. В галузі охорони здоров’я мають великі обсяги баз даних. Використовуючи аналітику великих даних, можна вивчати величезні масиви даних та знаходити приховану інформацію, приховані закономірності, щоб виявляти знання з даних та відповідно прогнозувати результати. Машинне навчання може допомогти запобігти, виявити та лікувати різні захворювання.

Технології машинного навчання та обробки даних є найкращими джерелами для вдосконалення системи охорони здоров’я. Ручне виявлення або діагностика, проведена лікарями, займає багато часу і не має точності. Машини, створені для того, щоб навчитися виявляти хвороби за допомогою машинного навчання та інтелектуального аналізу даних, можуть краще діагностувати проблему, і це теж з високою точністю. Машинне навчання може не тільки виявитися корисним для діагностики та прогнозування захворювання, але також може бути корисним при персоналізованому лікуванні та модифікації поведінки, виробництві ліків та виявленні нових моделей, що призводять до нових ліків та методів лікування, дослідженнях клінічних випробувань. За допомогою машинного навчання ми можемо робити все це і кращим чином. Машинне навчання вважається однією з найважливіших функцій штучного інтелекту, що підтримує розробку комп'ютерних систем, що мають можливість отримувати знання з минулого

досвіду без необхідності програмування для кожного випадку. Машинне навчання вважається гострою потребою сьогоднішньої ситуації з метою усунення зусиль людини шляхом підтримки автоматизації з мінімальними вадами. Існуючий метод виявлення цукрового діабету – використання лабораторних тестів, таких як глюкоза в крові натще і толерантність до перорального прийому глюкози. Однак цей спосіб трудомісткий.

Ми живемо в епоху, коли дані генеруються експоненційно, що призводить до підсумовування величезних даних. Особливо в галузі охорони здоров'я доступність даних є великою, але потреба у витягуванні знань з них також велика, інакше збір великих даних про охорону здоров'я є марним, якщо їх не використовувати. Інтелектуальний аналіз даних та машинне навчання допомагає знайти корисну інформацію, яку можна надалі широко використовувати. У галузі охорони здоров’я для покращення догляду за пацієнтами можна використовувати видобування даних та машинне навчання, найкращі практики, ефективне лікування пацієнта, виявлення шахрайства та доступніші медичні послуги. Видобуток даних також може бути використаний для виявлення спалаху чуми раніше часу (прогнозування) шляхом спостереження за тенденціями розвитку симптомів / скарг пацієнтів.

Різні моделі прогнозування були розроблені та впроваджені різними дослідниками з використанням варіантів методів видобутку даних, алгоритмів машинного навчання або також поєднання цих методів. Dr Saravana Kumar N M, Eswari, Sampath P і Lavanya S (2015) впровадили систему з використанням методів Hadoop та Map Reduce для аналізу даних діабетиків. Ця система передбачає тип діабету, а також пов'язані з ним ризики. Система базується на Hadoop і є економічною для будь-якої організації охорони здоров’я. Aiswarya Iyer (2015) використовувала методику класифікації для вивчення прихованих закономірностей у наборі даних цукрового діабету. У цій моделі використовувались наївні Байєс та дерева рішень. Було проведено порівняння ефективності обох алгоритмів, і в результаті було продемонстровано ефективність обох алгоритмів. К. Раджеш та В. Сангета (2012) використовували техніку

класифікації. Вони використовували алгоритм дерева рішень C4.5, щоб знайти приховані шаблони з набору даних для ефективної класифікації. Humar Kahramanli and Novruz Allahverdi (2008) використовували штучну нейронну мережу (ANN) у поєднанні з нечіткою логікою для прогнозування діабету. B.M. Patil, R.C. Joshi та Durga Toshniwal (2010) запропонували Гібридну модель прогнозування, яка включає простий алгоритм кластеризації K-засобів, з подальшим застосуванням алгоритму класифікації до результату, отриманого з алгоритму кластеризації. Для побудови класифікаторів використовується алгоритм дерева рішень C4.5. Mani Butwall і Shraddha Kumar (2015) запропонували модель із використанням Класифікатора випадкових лісів для прогнозування поведінки цукрового діабету. Nawaz Mohamudally1 та Dost Muhammad (2011) використовували алгоритм дерева рішень C4.5, нейронну мережу, алгоритм кластеризації K-засобів та візуалізацію для прогнозування цукрового діабету.

Я пропоную розглянути прогнозування ймовірності діагностики цукрового діабету на ранніх стадіях за допомогою ансамблевого методу машинного навчання – XGBoost реалізованого на мові програмування Phyton.

2.3 Мова програмування – Phyton

Python – це потужна, самовпевнена та своєрідна сценарійна мова, яку люблять (і ненавидять) програмісти у всьому світі завдяки своєму стилю, синтаксису та увазі до пробілів. Її функції можна виконувати за допомогою більш простих команд і менше тексту, ніж у більшості конкуруючих мов. І це може пояснити, чому популярність стрімко зростає серед розробників, студентів-програмістів та технологічних компаній.

Python був створений в 1991 році голландським програмістом Guido Van Rossum. Це інтерпретована мова. Це означає, що у нього є інтерпретатор для безпосереднього запуску програми, на відміну від більш складних машинних мов. Насправді, Van Rossum хоче, щоб Python з часом став таким же зрозумілим і

доступним, як і звичайна англійська. Він також зробив мову відкритою, що означає, що кожен може внести до неї вклад, і він сподівається, що вона стане такою ж потужною, як і мови-конкуренти.

“Читаність” є ключовим фактором у філософії Python . Як такий, він спрямований на обмеження блоків коду (блоків тексту вихідного коду) і замість цього має пробіли для більш чіткого, менш зайнятого вигляду. Це універсальна мова, яка працює на багатьох системах.

Не дивно, що з огляду на свою доступну та універсальну природу, Python входить до п’ятірки найпопулярніших мов у світі.

Python використовується Вікіпедією, Google (де раніше працював Van Rossum), Yahoo !, CERN та NASA, та багато інших організацій.

Він часто використовується як "мова сценаріїв" для веб-додатків. Це означає, що він може автоматизувати певний ряд завдань, роблячи його більш ефективним. Отже, Python (і подібні йому мови) часто використовується в програмних додатках, на сторінках веб-браузера, оболонках операційних систем та деяких іграх.

Як і інші мови кодування, Python – це один з небачених елементів, який ми використовуємо, не знаючи цього. YouTube та Instagram – одні серед незліченних сайтів, які використовують Python. Значною частиною коду Dropbox є Python (там, де зараз працює Van Rossum).

Мова використовується в науково-математичних обчисленнях і навіть у проектах ШІ.

Не буде перебільшенням сказати, що Python відіграє незначну частину всього нашого життя. Це одна з тих невидимих сил, яка присутня в наших мобільних пристроях, веб-пошуку та іграх (і не тільки). Отже, це був очевидний вибір для нашого дослідження ансамблевого методу машинного навчання – XGBoost.

2.4 Алгоритм роботи програми

Алгоритм – це набір інструкцій, призначених для виконання конкретного завдання. Тому було складено алгоритм роботи програми в нотації діаграми діяльності (рис. 2.1).

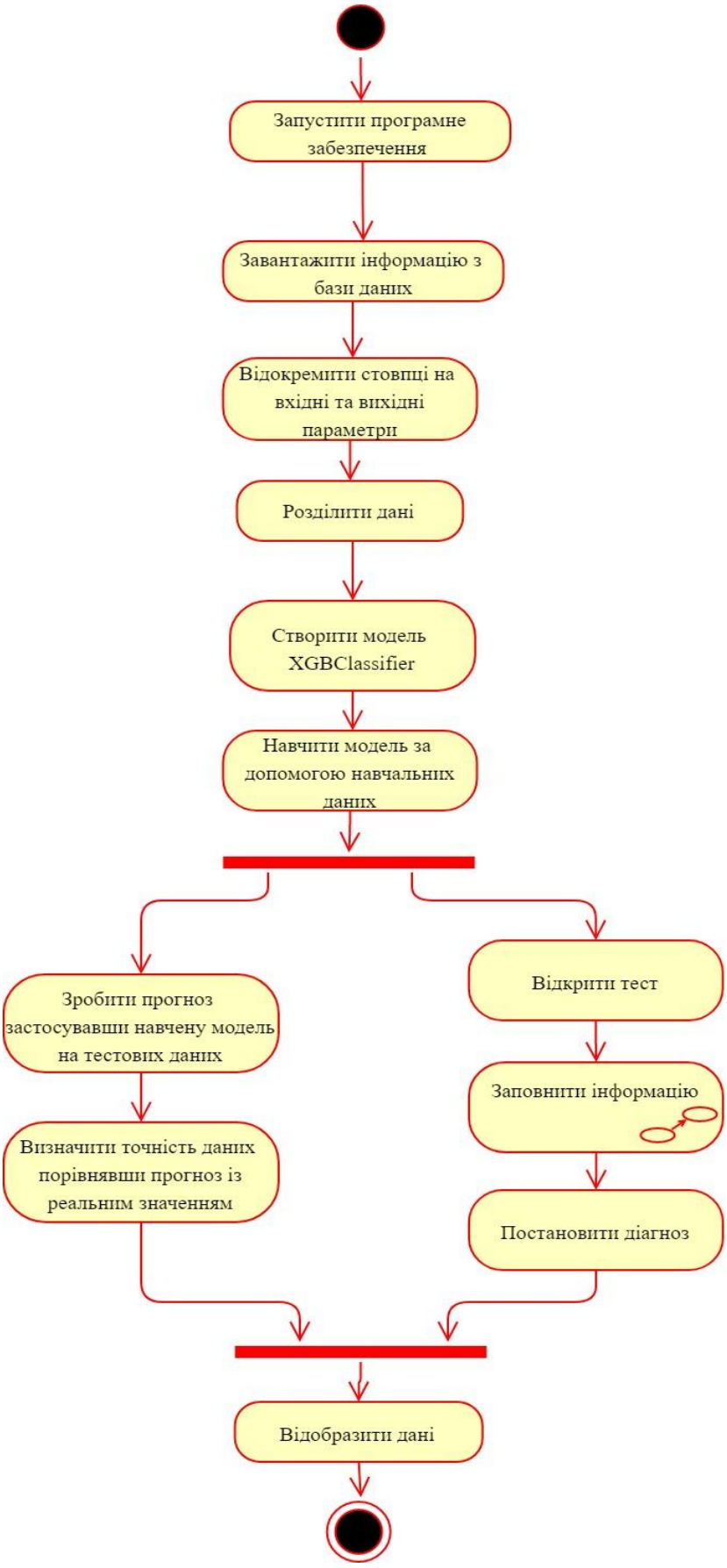


Рисунок 2.1 Діаграма діяльності алгоритму роботи програми

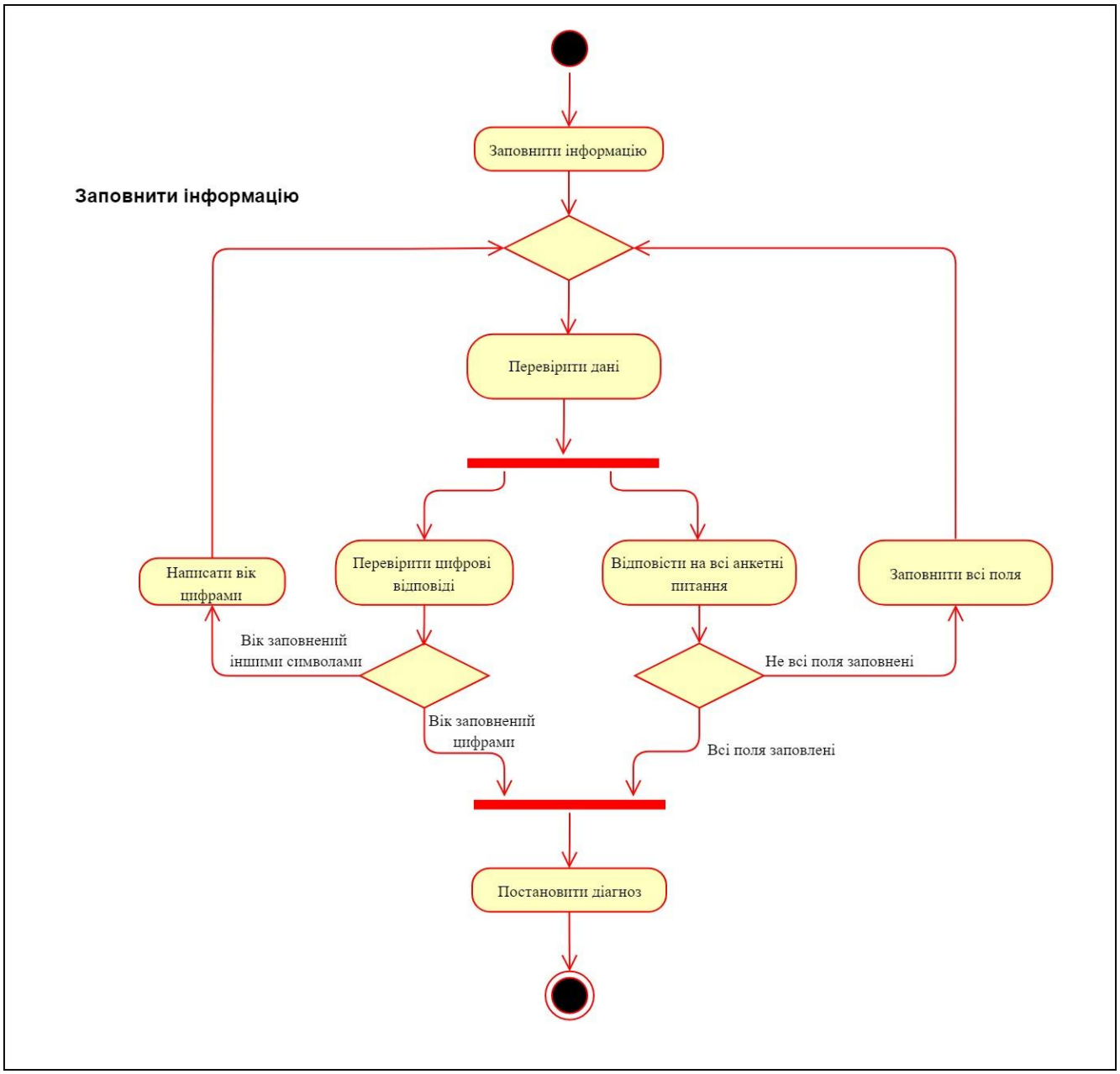


Рисунок 2.2 Декомпозиція діаграми діяльності

За даною методикою, для того щоб провести попереднє діагностування цукрового діабету у людини на ранніх стадіях, нам необхідно запустити програмне забезпечення. Після чого починається робота з самою програмою, йде процес завантаження інформації з бази даних. Програма аналізує інформацію у базі даних, відокремлюючи стовпці на вхідні (X – атрибути) та вихідні (Y – діагноз) параметри. Далі йде процес розділення даних на навчальну та тестову вибірку. Потім ми створюємо модель XGBClassifier та навчаємо її за допомогою навчальних даних. Тепер ми готові використовувати навчену модель для того щоб зробити прогноз із застосуванням наших тестових даних. Для того щоб,

визначити точність даних ми порівняємо наш прогноз із реальними значеннями. Паралельно з цим процесом ми проводимо тестування. Після відкриття тесту ми заповнюємо всю інформацію необхідну для постановки діагнозу. Цю дію більш детальніше можна побачити на рис. 2.2. Після заповнення інформації йде процес перевірки даних. Ми перевіряємо чи є відповіді у всіх анкетних питаннях та чи вірно введені цифрові відповіді. Якщо ж не всі поля заповнені, то ми заповнюємо всі поля або вік заповнений іншими символами (вік – єдине цифрове поле у нашій програмі), то ми пишемо вік цифрами та повертаємося до процесу перевірки даних. Якщо перевірка даних пройшла успішно, тобто ми заповнили всі поля та вік заповнений цифрами, наша навчена програма ставить діагноз. В кінці на екрані відображається діагноз та точність даних нашої програми. На цьому методика завершена.

3 РОБОТА З ПРОГРАМОЮ

3.1 Бібліотеки

3.1.1 Бібліотека XGBoost

Для початку роботи з нашою програмою нам необхідно встановити бібліотеку XGBoost.

XGBoost оптимізована розподілена gradient boosting бібліотека розроблена , щоб бути дуже ефективною, гнучкою і портативною. Реалізує алгоритми машинного навчання в рамках Gradient Boosting . XGBoost забезпечує паралельне підсилення дерев (також відоме як GBDT, GBM), яке швидко та точно вирішує багато проблем науки про дані.

XGBoost – це інструмент, мотивований формальним принципом. Що більш важливо, він розробляється як з глибоким розглядом з точки зору оптимізації систем, так і принципів машинного навчання . Метою цієї бібліотеки є розширення граничних обчислювальних значень машин, щоб забезпечити масштабовану , портативну та точну бібліотеку. Більшість розробників машинного навчання намагаються використовувати цю бібліотеку, щоб отримати більш точну модель.

Дану бібліотеку було застосовано у цьому програмному забезпечені для прогнозування діагнозу цукрового діабету.

Кроки встановлення бібліотеки XGBoost на комп’ютері:

1. Завантажити Python та встановити його на комп’ютер.
2. Встановити pip , якщо він відсутній на вашому комп'ютері.

Pip є менеджером пакетів для Python. Це означає, що це інструмент, який дозволяє встановлювати та керувати додатковими бібліотеками та залежностями , які не розповсюджуються як частина стандартної бібліотеки.

Щоб перевірити чи встановлено pip потрібно: перейти у командний рядок до розташування каталогу скриптів Python і ввести наступне - pip –version (Рисунок 3.1).

```
C:\Users\Your Name\AppData\Local\Programs\Python\Python36-32\Scripts>pip --version
```

Рисунок 3.1 – Перевірка версії pip

Якщо у не встановлено PIP, можна його завантажити та встановити за допомогою команди python get-pip.py.

Тепер pip встановлено на комп’ютері.

3. Завантажити файл Python XGBoost в інтернеті, при цьому переконатися що файл буде відповідати версії python та архітектурі системи.
4. Встановити завантажений файл XGBoost Python, використовуючи команду pip install. Це все. Тепер XGBoost успішно встановлено на комп’ютері.
5. Імпортувати пакет XGBoost в нашу програму (Рисунок 3.2).

```
import xgboost
import numpy
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from tkinter import *
from tkinter import messagebox as mb
```

Рисунок 3.2 Імпортовані бібілотеки та функції в програмі

3.1.2 Бібліотека NumPy

NumPy – це безкоштовна бібліотека Python, яка оснащена набором складних математичних розрахунків, придатних для обробки статистичних даних. Ця бібліотека, яка забезпечує багатовимірний об’єкт масиву, різні похідні об’єкти (наприклад, масковані масиви та матриці) та асортимент підпрограм для швидких операцій над масивами, включаючи математичні, логічні, маніпуляції з фігурами, сортування, виділення, введення / виведення , дискретні перетворення Фур'є, основна лінійна алгебра, основні статистичні операції, випадкове моделювання та багато іншого. Маніпулювання даними є ключовим компонентом у науках даних та машинному навчанні. У випадку машинного навчання система бере величезну

кількість даних і готується робити точні прогнози. Звичайно, йому потрібно було б виконувати арифметичні дії над числовими даними, щоб отримати щось значуще. Тут в гру вступає бібліотека NumPy. Основною одиницею даних в цій бібліотеці є масив NumPy. Всі дані зберігаються в масивах. Бібілотека забезпечує широкий репертуар математичних та статистичних операцій, які можна виконувати з цими масивами. Тому наступним імпортується пакет NumPy(Рисунок 3.2).

3.1.3 Бібліотека Sklearn

Sklearn (або Scikit-learn) – це бібліотека Python, яка пропонує різні функції для обробки даних, які можна використовувати для класифікації, кластеризації та вибору моделі.

Model_selection – це метод встановлення проекту для аналізу даних, а потім його використання для вимірювання нових даних. Вибір належної моделі дозволяє отримувати точні результати під час прогнозування. Для цього потрібно навчити модель , використовуючи певний набір даних. Потім перевірити модель на інший набір даних.

train_test_split – це функція у виборі моделі Sklearn для розділення масивів даних на два підмножини : для навчальних даних та для тестування даних. За допомогою цієї функції не потрібно ділити набір даних вручну. Тому наступною функцією потрібно додати до програми train_test_split із метода model_selection бібліотеки Sklearn (Рисунок 3.2).

За замовчуванням Sklearn train_test_split робить випадкові розділи для двох підмножин. Однак також можа вказати випадковий стан для операції. Використання одного і того ж набору даних як для навчання, так і для тестування залишає простір для прорахунків , що збільшує шанси неточних прогнозів .

Sklearn train_test_split має кілька параметрів (Рисунок 3.3).

```
train_test_split(X, y, train_size=0.*,test_size=0.*, random_state=*)
```

Рисунок 3.3 Приклад синтаксису train_test_split

Параметри train_test_split:

− x, y. Перший параметр - це набір даних, який вибирається для використання.

− train_size. Цей параметр встановлює розмір навчального набору даних . Є три варіанти: None, який за замовчуванням, Int, який вимагає точної кількості зразків, і float, який коливається від 0,1 до 1,0.

− test_size. Цей параметр визначає розмір набору даних для тестування . Стан за замовчуванням відповідає розміру навчання. Буде встановлено 0,25, якщо розмір тренування встановлений за замовчуванням.

− random_state. Режим за замовчуванням виконує випадковий розподіл за допомогою np.random. Крім того, можна додати ціле число, використовуючи точне число.

Використання одного і того ж набору даних як для навчання, так і для тестування залишає простір для прорахунків , що збільшує шанси неточних прогнозів .

train_test_split - функція дозволяє розбити набір даних з легкістю, переслідуючи в ідеальну модель . Також потрібно мати на увазі, що модель не повинна бути надмірно чи недоопрацьованою .

Надмірно опрацьована - це ситуація, коли модель показує майже ідеальну точність при обробці навчальних даних. Така ситуація трапляється, коли модель має складний набір правил . Коли модель надмірно опрацьована, вона може бути неточною при обробці нових даних.

Недоопрацьована коли модель не підходить навчальні даних з - за зведення правил , які занадто прості . Не можна покластися на модель недостатнього опрацьовану, щоб зробити точний прогноз.

Metrics – цей модуль реалізує функції, що оцінюють помилку передбачення для конкретних цілей.

accuracy_score - функція обчислює точність , або фракції (по замовчуванню) або кількість (normalize=False) правильних передбачень. У

класифікації з багатьма мітками функція повертає точність підмножини. Якщо весь набір передбачуваних міток для вибірки суворо збігається з справжнім набором міток, то точність підмножини становить 1,0; інакше 0,0.

Ця функція має кілька параметрів (Рисунок 3.4).

```
sklearn.metrics.accuracy_score(y_true, y_pred, *, normalize=True, sample_weight=None)
```

Рисунок 3.4 Приклад синтаксису accuracy_score

Параметри accuracy_score:

- y_true. Основні правдиві (правильні) мітки.
- y_pred. Передбачені мітки, повернені класифікатором.
- normalize. Якщо False, повертається кількість правильно класифікованих міток. В іншому випадку повертається частка правильно класифікованих міток.
- sample_weight. Ваги міток, за замовчуванням = None.

Для того щоб оцінити точність прогнозування необхідно імпортувати функцію accuracy_score із методу metrics з бібліотеки Sklearn (Рисунок 3.2).

3.1.4 Бібліотека Tkinter

Більшість пише код і запускає його в терміналі командного рядка або IDE (інтегрованому середовищі розробки), і код видає результат на основі того, що очікується від нього, або на терміналі, або в самій IDE. Однак що, якщо програміст хочете, щоб система мала вигадливий користувальницький інтерфейс потрібно відобразити результат за допомогою графічного інтерфейсу.

Графічний інтерфейс - це не що інше, як настільна програма, яка надає інтерфейс, який допомагає взаємодіяти з комп’ютерами та збагачує досвід передачі команди (введення з командного рядка) коду. Вона використовуються для виконання різних завдань на робочих столах, ноутбуках та інших електронних пристроях тощо.

Tkinter (Tk) - це спосіб створення в Python графічних інтерфейсів користувача (GUI) і включений у всі стандартні розподіли Python. Набір інструментів Tk - це міжплатформна колекція «графічних елементів управління», тобто віджетів, для побудови інтерфейсів додатків.

Цей фреймворк надає користувачам Python простий спосіб створення елементів графічного інтерфейсу за допомогою віджетів, знайдених у наборі інструментів Tk. Віджети Tk можна використовувати для побудови кнопок, меню, полів даних тощо у програмі Python. Одного разу створені, ці графічні елементи можуть бути пов’язані з функціями, функціональними методами даних чи навіть іншими віджетами або взаємодіяти з ними.

Tk вже давно є невід'ємною частиною Python. Він надає надійний і незалежний від платформи набір інструментів для вікон, який доступний програмістам Python, використовуючи tkinter пакет, та його розширення, tkinter.tixi tkinter.ttk модулі.

Tkinter пакет являє собою тонкий об'єктно-орієнтована шар поверх Tk. Для використання tkinter не потрібно писати код Tcl, але доведеться звернутися до документації Tk, а іноді і до документації Tcl. tkinter- це набір обгорток, які реалізують віджети Tk як класи Python. Крім того, внутрішній модуль _tkinter забезпечує захищений від потоків механізм, який дозволяє взаємодіяти Python і Tcl.

Головними достоїнствами є те, що це швидко, і що воно зазвичай постачається в комплекті з Python. Хоча його стандартна документація є слабкою, доступний хороший матеріал, який включає: посилання, навчальні посібники, книгу та інші. Tkinter також славиться своїм застарілим зовнішнім виглядом, який значно покращений у Tk 8.5.

Здебільшого tkinter це все, що вам дійсно потрібно, але доступна також низка додаткових модулів. Інтерфейс Tk розташований у двійковому модулі з іменем _tkinter. Цей модуль містить низькорівневий інтерфейс до Tk, і ніколи не повинен використовуватися безпосередньо програмістами програм. Зазвичай це

спільна бібліотека (або DLL), але в деяких випадках може статично пов’язана з інтерпретатором Python.

На додаток до інтерфейсного модуля Tk, tkinter включає ряд модулів Python, які tkinter.constants є одними з найважливіших. Імпорт tkinter автоматично імпортується tkinter.constants, тому, як правило, для використання Tkinter все, що вам потрібно, це проста заява про імпорт:

Отож, для кращого сприйняття інформації при проходженні тесту потрібно імпортувати бібліотеку Tkinter (Рисунок 3.2).

3.2 Завантаження та підготовка даних

У проекті прогнозного моделювання алгоритми машинного навчання вивчають відображення від вхідних змінних до цільової змінної.

Найпоширеніша форма проекту прогнозного моделювання включає так звані структуровані дані або табличні дані. Це дані, які мають вигляд в електронній таблиці або матриці, з рядками прикладів і стовпцями об’єктів для кожного прикладу.

Не можна вмістити та оцінювати алгоритми машинного навчання на вихідних даних; натомість людина повинна трансформувати дані відповідно до вимог окремих алгоритмів машинного навчання. Більше того, вона повинна вибрати подання для даних, яке найкраще викриває невідому основну структуру проблеми прогнозування для алгоритмів навчання, щоб отримати найкращу ефективність, враховуючи наявні ресурси в проекті прогнозного моделювання.

Враховуючи те, що є стандартні реалізації високопараметризованих алгоритмів машинного навчання у бібліотеках з відкритим кодом, моделі підгонки стали звичним явищем. Таким чином, найскладнішою частиною кожного проекту прогнозного моделювання є те, як підготувати єдине, що є унікальним для проекту: дані, що використовуються для моделювання.

Дані, використані у цьому дослідженні, були отримані від 520 пацієнтів Sylhet Diabetes Hospital у Сілхеті, Бангладеш, шляхом прямих опитувальних

листів пацієнтів у 2020 році та схвалено лікарем. Для дослідження було обрано 17 змінних, 16 з яких вважаються вхідними змінними, а одна служить змінною відповіді. Атрибути в цій базі даних є цілими чи дійсними. Всі дані відповідають симптомам на ранніх стадіях цукрового діабету, були підібрані таким чином, щоб була можливість поставити діагноз, як найточніше.

Інформація про атрибути:

1. Вік, було обрано людей від 20 до 65 років.
2. Стать: 1 – Чоловік, 2 – Жінка.
3. Поліурія (часте сечовипускання): 1 –Так, 0 – Ні.
4. Полідипсія (відчуття постійної спраги): 1 – Так, 0 – Ні.
5. Раптова втрата ваги: 1 – Так, 0 – Ні.
6. Слабкість: 1 – Так, 0 – Ні.
7. Поліфагія (відчуття постійного голоду): 1 – Так, 0 – Ні.
8. Генітальна молочниця (інфекційна хвороба): 1 – Так, 0 – Ні.
9. Погіршення зору: 1 – Так, 0 – Ні.
10. Свербіж: 1 – Так, 0 – Ні.
11. Дратівливість: 1 – Так, 0 – Ні.
12. Довге загоєння ран: 1 – Так, 0 – Ні.
13. Часткове зниження м’язової сили: 1 – Так, 0 – Ні.
14. М’язова напруженість: 1 – Так, 0 – Ні.
15. Алопеція (патологічне випадіння волосся): 1 – Так, 0 – Ні.
16. Ожиріння (надмірна вага): 1 – Так, 0 – Ні.
17. Діагноз: 1 – Позитивний, 0 – Негативний.

Це дуже гарний набір даних для нашого дослідження, тому що всі атрибути є числовими змінними. Вхідні дані можуть мати багато форм, таких як зображення, часові ряди, текст, відео тощо. Найпоширеніший тип вхідних даних зазвичай називають табличними даними або структурованими даними. Це дані, які можна побачити в електронній таблиці, у базі даних або у файлі змінної, відокремленому комами, тому весь набір було збережено у форматі CSV, для кращого сприймання програмою.

Для того щоб, завантажити файл CSV у вигляді масиву NumPy в дану програму, потрібно використати функцію loadtxt() – Рисунок 3.5

```
# load data
dataset = numpy.loadtxt('diabetes.csv', delimiter=",")
```

Рисунок 3.5 Завантаження даних у програму

Далі необхідно відокремити стовпці набору даних вхідних атрибутів (X) та вихідні атрибути (Y). Це можна зробити дуже легко, дізнавшись вказані індекси стовпців у форматі NumPy масиву (Рисунок 3.6).

```
# split data into X and y
X1 = dataset[:,0:16]
Y1 = dataset[:,16]
```

Рисунок 3.6 – Відокремлення даних на X та Y

І нарешті потрібно розділити дані X та Y на навчальний та тестовий набори даних. Навчальний набір буде використовуватися для створення моделі XGBoost, а тестовий набір даних буде використовуватися, для того щоб зробити прогноз, за допомогою якого можна буде оцінити якість моделі. Для цього було використано функцію train_test_split() із бібліотеки scikit-learn (Рисунок 3.7).

```
# split data into train and test sets
seed = 17
test_size = 0.33
X_train, X_test, y_train, y_test = train_test_split(X1, Y1, test_size=test_size, random_state=seed)
```

Рисунок 3.7 – Розділення даних на тестовий та навчальний набір

Тепер все готово для того щоб навчити модель XGBoost.

3.3 Навчання моделі XGBoost та прогнозування діагнозу за її допомогою

Модель XGBoost для класифікації називається XGBClassifier. Можна створити цю модель та навчити її за допомогою навчального набору даних. Дана модель має функцію fit() для навчання моделі (Рисунок 3.8).

Параметри для навчання моделі можуть бути передані моделі в конструкторі.

```
# fit model no training data
model = xgboost.XGBClassifier()
model.fit(X_train, y_train)
```

Рисунок 3.8 – Навчання моделі XGBClassifier

Тепер можна зробити прогнози, застосувавши навчену модель на тестовому наборі даних. Для цього, щоб робити прогнози ми використовуємо scikit-learn функцію model.predict () (Рисунок 3.9) .

За замовчуванням, передбачення, зроблені XGBoost є можливостями. Оскільки це бінарна задача класифікації, кожний прогноз є ймовірністю приналежності до першого класу. Тому можна легко перетворити їх в значення двоїчних класів шляхом округлення до 0 або 1.

```
# make predictions for test data
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]
```

Рисунок 3.9 – Прогноз за допомогою XGBoost

Тепер, коли було використано навчену модель, щоб зробити прогнози за новими даними, можна оцінити якість прогнозів, порівнюючи їх з реальними значеннями. Для цього було використано вбудовану в scikit-learn функцію accuracy_score () (Рисунок 3.10).

```
# evaluate predictions
accuracy = accuracy_score(y_test, predictions)*100.0
```

Рисунок 3.10 – Визначення точності прогнозування

3.4 Створення інтерфейсу користувача

Tkinter надає різноманітні загальні елементи графічного інтерфейсу, які можна використовувати для побудови інтерфейсу - такі як кнопки, меню та різні види полів введення та області відображення. Називаються ці елементи віджетами. Класи віджетів надають багато функціональних можливостей за замовчуванням. У них є методи для налаштування зовнішнього вигляду

графічного інтерфейсу - наприклад, упорядкування елементів за певним макетом - і для обробки різних видів керованих користувачем подій.

3.4.1 Класи віджетів

Вбудовано багато різних класів віджетів tkinter:

- Frame – це віджет-контейнер, який розміщується всередині вікна, яке може мати власну межу та фон - воно використовується для групування пов’язаних віджетів разом у макеті програми.
- Canvas – це віджет для малювання графіки. У розширеному використанні його також можна використовувати для створення власних віджетів - можна намалювати все, що подобається, і зробити його інтерактивним.
- Text – відображає відформатований текст, який можна редагувати та мати вбудовані зображення.
- Button – зазвичай відображає безпосередньо на дії користувача - коли користувач натискає на кнопку, що - щось має статися.
- Label – це простий віджет, який відображає короткий фрагмент тексту або зображення, але зазвичай не є інтерактивним.
- Message – схожий на a Label, але призначений для довших текстів, які потрібно обернути.
- Scrollbar – дозволяє користувачеві прокручувати вміст, який занадто великий, щоб бути видимим відразу.
- Checkbutton, Radiobutton, Listbox – різні види віджетів введення - вони дозволяють користувачеві вводити інформацію в програму.
- Menu та Menubutton – використовуються для створення випадаючих меню.

3.4.2 Створення кореневого вікна

Tk - це клас, який було використано для створення кореневого вікна - головного вікна цієї програми. За допомогою методу geometry() присвоюємо вікну його розміри. А за допомогою методу title() заголовок програми (Рисунок 3.11)

```
root = Tk()
root.geometry('750x650+10+10')
root.title("Predicting Diabetes Mellitus With Machine Learning Techniques")
```

Рисунок 3.11 Створення кореневого вікна

Віджет Frame дуже важливий для процесу групування та упорядкування інших віджетів. Він працює як контейнер, який відповідає за організацію положення інших віджетів. Він використовує прямокутні області на екрані для впорядкування макета та для заповнення цих віджетів. Frame також можна використовувати як базовий клас для реалізації складних віджетів.
Ось простий синтаксис для створення цього віджета (Рисунок 3.12)

```
w = Frame ( master, option, ... )
```

Рисунок 3.12 – Синтаксис віджета Frame

Параметри:

- master - представляє батьківське вікно.
- options - параметри цього віджета. Ці параметри можна використовувати як пари ключ-значення, розділені комами.

Тому для створення контейнера для всіх інших віджетів було створено віджет Frame (Рисунок 3.13)

```
body=Frame(root,height=400)
```

Рисунок 3.13 – Створення віджета Frame

Для того щоб віджети відображалися у вікні, було застосовано менеджер геометрії – pack.

pack - це найпростіший менеджер віджетів, яким можна кодувати в Tkinter. Замість того, щоб оголосити точне розташування віджета, метод pack ( ) оголошує позицію віджетів по відношенню один до одного. Однак точність pack () обмежена у порівнянні з place() та grid( ), які мають абсолютне позиціонування. Для простого розташування віджетів вертикально або горизонтально по відношенню один до одного, pack () є вибраним менеджером макета.

pack () організовує віджети в горизонтальних та вертикальних полях, які обмежені лівим, правим, верхнім, нижнім положеннями, зміщеними та відносно один одного в межах кадру.

Опції заповнення pack ():

- padx , який прокладає зовні вздовж осі x;
- pady , який прокладає зовні вздовж осі y;
- ipadx , який прокладається внутрішньо вздовж осі x;
- ipady , який прокладається всередині вздовж осі y;
- anchor, використовуються для визначення місця розташування тексту відносно контрольної точки, може приймати значення N (north) – північ, S (south) – південь, W (west) – захід, E (east) – схід та їх комбінації(Рисунок 3.14).

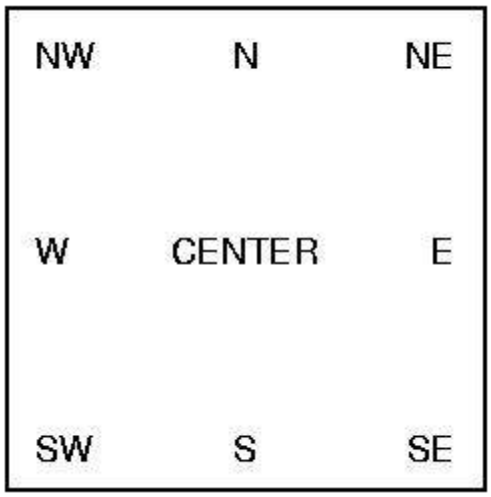


Рисунок 3.14 – Констанси anchor

Тому, необхідно застосувати даний метод для нашого контейнера, та подальшого відображення віджетів.(Рисунок 3.15)

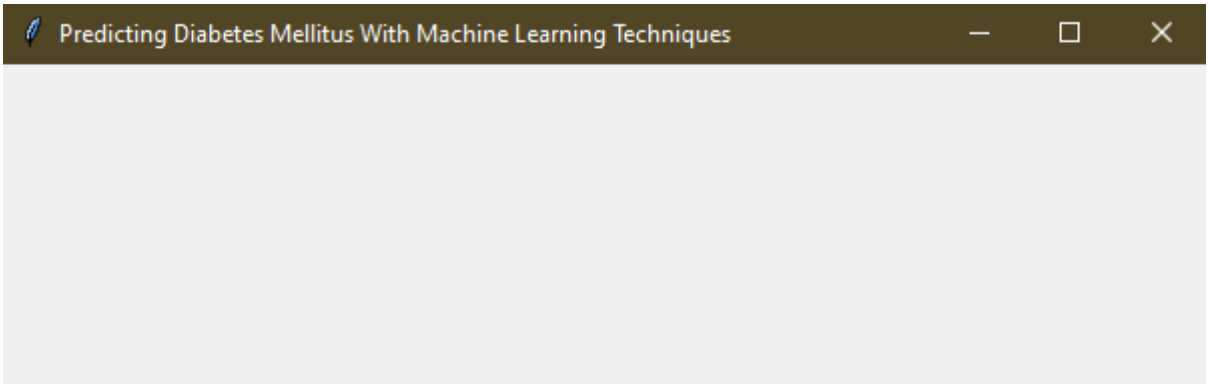


Рисунок 3.15 – Відображення кореневого вікна

3.4.3 Створення віджета Label

Для постановки діагнозу, необхідно створити питання, на які користувач буде давати відповіді під час проходження тесту. Для цього було використано віджет Label.

Tkinter Label - це віджет, який використовується для реалізації вікон, на яких можна розмістити текст або зображення. Текст, який відображається за допомогою цього віджета, розробник може змінити у будь-який час. Він також використовується для виконання таких завдань, як підкреслення частини тексту та розподіл тексту по декількох рядках. Важливо зазначити, що на Label може відображатися лише один шрифт одночасно для відображення тексту. Щоб використовувати Label, просто потрібно вказати, що в ній відображати (це може бути текст або зображення).

Синтаксис для створення цього віджета (Рисунок 3.16).

```
w = Label ( master, option, ... )
```

Рисунок 3.16 – Синтаксис віджета Lable

Параметри:

- master - представляє батьківське вікно.
- options - параметри цього віджета. Ці параметри можна використовувати як пари ключ-значення, розділені комами.

Деякі опції Label, які використовуються у програмі:

- text – щоб відобразити один або кілька рядків тексту у віджеті Label, необхідно встановити для цього параметра рядок, що містить текст. Внутрішні нові рядки ("\ n") примусять до розриву рядка:
- font – параметр вказує, яким шрифтом цей текст буде відображатися;
- fg – цей параметр визначає колір тексту.

У програмі було створено віджети Label. У опції text було записано необхідний текст, а у - font, його шрифт та розмір – Рисунок 3.17

```
l0 = Label(scrollable_body,text = "Пройдіть короткий тест на наявність цукрово діабету:", font = ("Times New Roman", 18))
l1 = Label(scrollable_body,text = "Скільки вам років?", font = ("Times New Roman", 14))
l2 = Label(scrollable_body,text = "Вкажіть вашу стать:", font = ("Times New Roman", 14))
l3 = Label(scrollable_body,text = "Ви часто ходите в туалет?", font = ("Times New Roman", 14))
l4 = Label(scrollable_body,text = "Ви часто відчуваєте спрагу?", font = ("Times New Roman", 14))
l5 = Label(scrollable_body,text = "Чи втрачали ви раптово вагу останнім часом?", font = ("Times New Roman", 14))
l6 = Label(scrollable_body,text = "Чи часто ви відчуваєте слабкість?", font = ("Times New Roman", 14))
l7 = Label(scrollable_body,text = "Чи є у вас відчуття постійного голоду?", font = ("Times New Roman", 14))
l8 = Label(scrollable_body,text = "Чи є вас свербіж, почервоніння чи рясні білі виділення на статевих органах?", font = ("Times New Roman", 14))
l9 = Label(scrollable_body,text = "Чи було у вас погіршення зору останнім часом?", font = ("Times New Roman", 14))
l10 = Label(scrollable_body,text = "Чи є у вас свербіж шкіри?", font = ("Times New Roman", 14))
l11 = Label(scrollable_body,text = "Чи часто ви дратуєтесь?", font = ("Times New Roman", 14))
l12 = Label(scrollable_body,text = "Чи довго у вас загоюються рани?", font = ("Times New Roman", 14))
l13 = Label(scrollable_body,text = "Чи є у вас часткове зниження м'язової сили?", font = ("Times New Roman", 14))
l14 = Label(scrollable_body,text = "Чи відчуваєте ви м'язову напруженість?", font = ("Times New Roman", 14))
l15 = Label(scrollable_body,text = "Чи багато волосся у вас випадає останнім часом?", font = ("Times New Roman", 14))
l16 = Label(scrollable_body,text = "У вас збільшена маса тіла?", font = ("Times New Roman", 14))
l17 = Label(scrollable_body,text = "Результат:", font = ("Times New Roman", 14), fg="red")
l18 = Label(scrollable_body,text = "Точність даних: %.2f%%" % accuracy, font = ("Times New Roman", 12), fg="blue")
```

Рисунок 3.17 – Створення віджета Lable

Для проходження тесту були написані запитання, які відповідають атрибутам бази даних, на основі якої програма навчилася ставити діагноз.

Питання для поставлення діагнозу цукрового діабету на ранніх стадіях:

1. Скільки вам років?
2. Вкажіть вашу стать.
3. Ви часто ходите в туалет?
4. Ви часто відчуваєте спрагу?
5. Чи втрачали ви раптово вагу останнім часом?
6. Чи часто ви відчуваєте слабкість?
7. Чи є у вас відчуття постійного голоду?
8. Чи є вас свербіж, почервоніння чи рясні білі виділення на статевих органах?
9. Чи було у вас погіршення зору останнім часом?
10. Чи є у вас свербіж шкіри?
11. Чи часто ви дратуєтесь?
12. Чи довго у вас загоюються рани?
13. Чи є у вас часткове зниження м'язової сили?
14. Чи відчуваєте ви м'язову напруженість?
15. Чи багато волосся у вас випадає останнім часом?
16. У вас збільшена маса тіла?
17. Результат.

В останньому віджеті Lable під назвою «l18» відображається оцінка якості прогнозів нашої програми.(Рисунок 3.18)

```
Точність даних: 95.93%
```

Рисунок 3.18 – Оцінка якості прогнозів

3.4.4 Створення віджета Button

Віджет Button використовується для додавання кнопок у програмі Python. Ці кнопки можуть відображати текст або зображення, що передають призначення кнопок. Можна приєднати функцію або метод до кнопки, яка викликається автоматично при натисканні кнопки. Button - це віджет, призначений для взаємодії з користувачем, тобто якщо кнопку натиснути клацанням миші, може бути розпочато якусь дію. Функція або метод Python можуть бути пов'язані з кнопкою. Ця функція або метод буде виконаний, якщо кнопку натиснути певним чином.

Синтаксис для створення цього віджета (Рисунок 3.19)

```
w = Button ( master, option=value, ... )
```

Рисунок 3.19 – Синтаксис віджета Button

Параметри:

- master – представляє батьківське вікно;
- options – параметри цього віджета. Ці параметри можна використовувати як пари ключ-значення, розділені комами.

Деякі опції Button, які використовуються у програмі:

- command – функція або метод, що викликаються при натисканні кнопки;
- text – відображає текст на кнопці.
- bg – звичайний колір фону;
- height – висота кнопки в текстових рядках (для текстових кнопок) або пікселях (для зображень);
- width – ширина кнопки буквами (якщо відображається текст) або пікселями (якщо відображається зображення).

У програмі створено віджет Button та додано необідні опції (Рисунок 3.20)

```
but = Button(scrollable_body,text="Результат",width=15, height=2,command=check)
but['bg']='#7FFFD4'
```

Рисунок 3.20 – Створення віджета Button

3.4.5 Створення віджета Radiobutton

Radiobutton, яку іноді називають кнопкою опцій, є графічним елементом інтерфейсу користувача Tkinter, що дозволяє користувачеві вибрати (саме) один із заздалегідь визначеного набору параметрів. Перемикачі можуть містити текст або зображення. Кнопка може відображати текст лише одним шрифтом. Функція або метод Python можуть бути пов'язані з перемикачем. Ця функція або метод буде викликаний, якщо натиснути цей перемикач.

Radiobutton названі на честь фізичних кнопок, що використовуються на старих радіостанціях для вибору діапазонів хвиль або попередньо встановлених радіостанцій. Якщо натиснути таку кнопку, інші кнопки вискакували б, залишаючи натиснуту кнопку єдиною натиснутою.

Кожна група віджетів Radiobutton повинна бути пов'язана з однією і тією ж змінною. Натискання кнопки змінює значення цієї змінної на заздалегідь визначене значення.

Синтаксис для створення цього віджета – Рисунок 3.21

```
w = Radiobutton ( master, option, ...  )
```

Рисуно 3.21 – Синтаксис віджета Radiobutton

Параметри:

- master – представляє батьківське вікно;
- options – параметри цього віджета. Ці параметри можна використовувати як пари ключ-значення, розділені комами.

Деякі опції Radiobutton, які використовуються у програмі:

- text – label, що відображається біля перемикача. Використовуйте нові рядки ("\ n") для відображення декількох рядків тексту;
- font – шрифт, що використовується для тексту;

- variable – керуюча змінна, якою ця перемикач ділиться з іншими перемикачами у групі. Це може бути як IntVar, так і StringVar;
- value - коли radiobutton увімкнена користувачем, його змінна керування встановлюється як параметр поточного значення. Якщо змінною управління є IntVar , надається кожному перемикачу в групі інший варіант цілочисельного значення. Якщо змінною керування є StringVar , надається кожному перемикачу інший варіант значення рядка.

У цій програмі було створено 15 груп віджетів для відповіді на кожне запитання, загалом це 30 віджетів Radiobutton.(Рисунок 3.22)

```
var = IntVar()
var.set(0)
b1 = Radiobutton(scrollable_body,text="Чоловік",font = ("Times New Roman", 12),
                 variable=var, value=1)
b2 = Radiobutton(scrollable_body,text="Жінка",font = ("Times New Roman", 12),
                   variable=var, value=2)

var1 = IntVar()
var1.set(0)
b3 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
                 variable=var1, value=3)
b4 = Radiobutton(scrollable_body,text="Ні",font = ("Times New Roman", 12),
                   variable=var1, value=4)

var2 = IntVar()
var2.set(0)
b5 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
                 variable=var2, value=5)
b6 = Radiobutton(scrollable_body,text="Ні",font = ("Times New Roman", 12),
                   variable=var2, value=6)

var3 = IntVar()
var3.set(0)
b7 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
                 variable=var3, value=7)
b8 = Radiobutton(scrollable_body,text="Ні",font = ("Times New Roman", 12),
                   variable=var3, value=8)

var4 = IntVar()
var4.set(0)
b9 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
                 variable=var4, value=9)
b10 = Radiobutton(scrollable_body,text="Ні",font = ("Times New Roman", 12),
                   variable=var4, value=10)

var5 = IntVar()
var5.set(0)
b11 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
                 variable=var5, value=11)
b12 = Radiobutton(scrollable_body,text="Ні",font = ("Times New Roman", 12),
                   variable=var5, value=12)

var6 = IntVar()
var6.set(0)
b13 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
                 variable=var6, value=13)
b14 = Radiobutton(scrollable_body,text="Ні",font = ("Times New Roman", 12),
                   variable=var6, value=14)
```

Рисунок 3.22 – Створення віджетів Radiobutton

3.4.6 Створення віджета Entry

Віджети Entry - це основні віджети Tkinter, які використовуються для отримання введення, тобто текстових рядків, від користувача програми. Цей віджет дозволяє користувачеві вводити один рядок тексту. Якщо користувач

вводить рядок, який перевищує доступний простір відображення віджета, вміст буде прокручуватися. Це означає, що рядок не можна побачити повністю. Клавіші зі стрілками можна використовувати для переміщення до невидимих частин струни. Якщо ви хочете ввести кілька рядків тексту, вам доведеться скористатися текстовим віджетом. Віджет введення також обмежений одним шрифтом.

Синтаксис для створення цього віджета – Рисунок 3.23.

```
w = Entry( master, option, ... )
```

Рисунок 3.23 – Синтаксис віджета Entry

Параметри:

- master – представляє батьківське вікно;
- options – параметри цього віджета. Ці параметри можна використовувати як пари ключ-значення, розділені комами.

Деякі опції Entry, які використовуються у програмі:

- width - ширина прапорця за замовчуванням визначається розміром відображуваного зображення або тексту. Можна встановити для цього параметра кількість символів, і кнопка завжди матиме місце для такої кількості символів.

Деякі методи, які використовуються у програмі:

- get() - повертає поточний текст запису у вигляді рядка.

У даній програмі створено лише один віджет Entry, який відповідє за вік користувача. (Рисуно 3.24)

```
entry = Entry(scrollable_body, width=5)
```

Рисунок 3.24 – Створення віджету Entry

Далі за допомогою методу pack () відобразимо всі створені віджети у кореневому вікні. (Рисунок 3.25)

```
l0.pack()
l1.pack(anchor=NW, padx=20,pady=5)
entry.pack(anchor=NW, padx=30)
l2.pack(anchor=NW, padx=20,pady=5)
b1.pack(anchor=NW,padx=20)
b2.pack(anchor=NW,padx=20)
l3.pack(anchor=NW, padx=20,pady=5)
b3.pack(anchor=NW,padx=20)
b4.pack(anchor=NW,padx=20)
l4.pack(anchor=NW, padx=20,pady=5)
b5.pack(anchor=NW,padx=20)
b6.pack(anchor=NW,padx=20)
l5.pack(anchor=NW, padx=20,pady=5)
b7.pack(anchor=NW,padx=20)
b8.pack(anchor=NW,padx=20)
l6.pack(anchor=NW, padx=20,pady=5)
b9.pack(anchor=NW,padx=20)
b10.pack(anchor=NW,padx=20)
l7.pack(anchor=NW, padx=20,pady=5)
b11.pack(anchor=NW,padx=20)
b12.pack(anchor=NW,padx=20)
l8.pack(anchor=NW, padx=20,pady=5)
```

Рисунок 3.25 – Застосування методу pack для віджетів

А метод mainloop() запускає головний цикл обробки подій, що в тому числі призводить до відображення головного вікна з усіма "упакованими" на ньому віджетами. Цей метод прослуховує події, такі як натискання кнопок або натискання клавіш, і блокує будь-який код, який приходить після нього, від запуску, доки не закриється вікно, до якого його викликають.

Якщо не включити mainloop() в кінець програми у файл Python, тоді програма Tkinter ніколи не запускатиметься і нічого не відображатиметься. Тому необхідно застосувати цей метод у програмі. (Рисунок 3.26)

```
root.mainloop()
```

Рисунок 3.26 – Застосування методу mainloop()

3.4.7 Метод перевірки даних

У програмі є кнопка «Результат» і для того щоб відбувалася, якась дія, при натисненні на кнопку було створено метод chek().

В цьому методі, обробляються всі дані, для поставлення діагнозу за допомогою навченої програми.

Для початку, необхідно створити list (Рисунок 3.27).

```
l = []
```

M

Рисунок 3.27 Створення масиву

List – являє собою структуру даних в Python , який є змінною, або впорядкованою послідовністю елементів. Кожен елемент або значення, що знаходиться всередині списку, називається елементом. Подібно до того, як рядки визначаються як символи між лапками, списки визначаються, маючи значення між квадратними дужками [ ].

Lists чудово використовувати, коли потрібно працювати з багатьма пов'язаними цінностями. Вони дозволяють зберігати дані, що належать разом, ущільнювати код і виконувати однакові методи та операції з кількома значеннями одночасно.

В цей list будуть записуватися значення, після обробки методом відповідей на поставлені питання в тесті.

Для того, щоб зчитати інформацію про вік користувача з поля Entry було застосовано метод get().(Рисунок 3.28)

```
def check():
    s = entry.get()
```

Рисунок 3.28 – Застосування методу get

Далі за допомогою цього ж методу необхідно зчитати значення всіх віджетів Radiobutton, при цьому порівнюючи значення, яке обрав користувач із значенням яке стояло за замовчуванням, для того щоб програма зрозуміла, яка кнопка активна, тобто яку відповідь обрав користувач. Зроблено це було за допомого логічних операторів if … else.

Оператор else може поєднуватися з оператором if . Інструкція else містить блок коду, який виконується, якщо умовний вираз у операторі if має значення 0 або значення FALSE. Оператор else є необов'язковим, і наступним може бути лише одне інше твердження if .

Синтаксис оператора if ... else – Рисунок 3.29

```
if expression:
   statement(s)
else:
   statement(s)
```

Рисунок 3.29 Синтаксис оператора if ... else

Принцип роботи даного оператора зображений на Рисунку 3.30

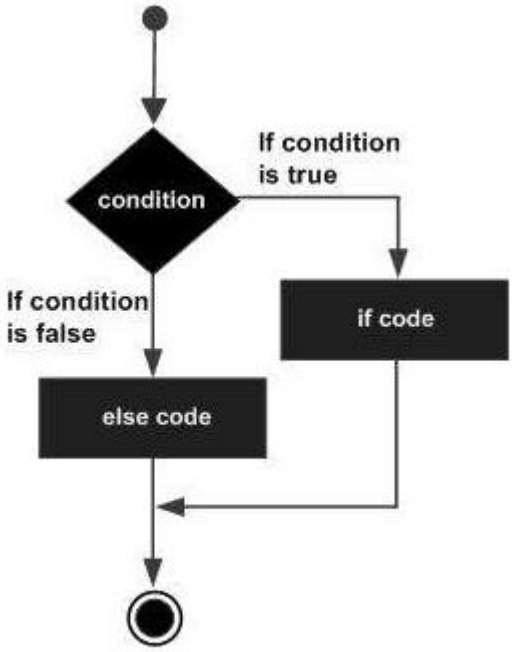


Рисунок 3.30 – Принцип роботи оператора if ... else

Після того, як програма зрозуміла, яка кнопка була активна, вона записує значення цієї кнопки у list.(Таблиця 3.1)

Таблиця 3.1 – Види значень у програмі

Відповідь користувача	Значення у програмі
Чоловік	1
Жінка	2
Так	1
Ні	0

Далі, перевіривши всі значення та записавши їх у list, перетворюємо його у numpy та за допомогою навченої програми ставимо діагнозу на основі відповідей користувача(Рисунок 3.31).

```
X_test1 = numpy.array([l])
y_pred1 = model.predict(X_test1)
predictions1 = [round(value) for value in y_pred1]
if predictions1[0] == 0:
    ll7['text'] = "Результат: Ймовірно у вас немає цукрового діабету!"
else:
    ll7['text'] = "Результат: Ймовірно ви маєте цукровий діабет. Рекомендуємо звернутися до лікаря! "
```

Рисунок 3.31 – Визначення діагнозу за допомогою навченої програми

Результат відображається у віджеті Label – «Результат». (Рисунок 3.32)

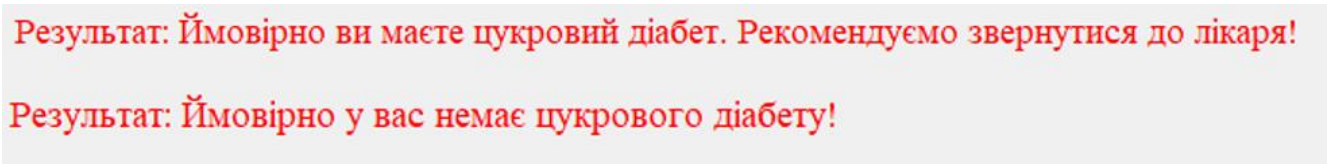


Рисунок 3.32 Відображення результату діагноза

В кінці даного методу ми очищуємо list за допомогою методу clear(), щоб при зміні відповідей користувачем, та постановки нового діагнозу, відображався правильний результат.(Рисунок 3.33)

```
l.clear()
```

Рисунок 3.33 – Видалення всіх елементів list

3.4.8 Клас Scrollable

У цій програмі дуже багато віджетів і вони всі не поміщаються на екран комп’ютера, тому для того щоб можна було дати відповіді на всі питання та побачити всі віджети було застосовано клас Scrollable, щоб зробити прокручуваний Frame, який буде містити всі віджети.

Scrollbar - це віджет, який корисний для прокрутки тексту в іншому віджеті. Наприклад, текст у тексті, рамці полотна чи списку можна прокручувати зверху вниз або зліва направо за допомогою смуг прокрутки. За допомогою віджета Scrollbar Tkinter можна створити горизонтальну смугу прокрутки також на віджетах Entry. Scrollbar Tkinter використовується для прокручування вмісту, щоб побачити весь вміст вертикально, коли смуга прокрутки встановлена у вертикальне положення або використовується горизонтальна смуга прокрутки для прокрутки вмісту по горизонталі.

Синтаксис використання віджета Scrollbar наведено на Рисунку 3.34

```
w = Scrollbar ( master, option, ... )
```

Рисунок 3.34 – Синтаксис віджета Scrollbar

Параметри:

- master – представляє батьківське вікно;
- options – параметри цього віджета. Ці параметри можна використовувати як пари ключ-значення, розділені комами.

Деякі опції Scrollbar, які використовуються у програмі:

- width – цей параметр використовується для представлення ширини смуги прокрутки;
- command – процедура, яку потрібно викликати при кожному переміщенні смуги прокрутки.

Методи Scrollbar:

- get() - повертає два числа (a, b), що описують поточне положення повзуна. Значення a надає положення лівого або верхнього краю повзуна для горизонтальної та вертикальної смуг прокрутки відповідно; значення b дає положення правого або нижнього краю;
- set (first, last) – щоб підключити смугу прокрутки до іншого віджету w, необхадно встановити w's xscrollcommand або yscrollcommand на метод set() смуги прокрутки.

Для прокрутки всіх елементів було створено клас полоси прокрутки Scrollable. (Рисунок 3.35)

```
scrollable_body = Scrollable(body, width=18)
```

Рисунок 3.35 –Створення класу Scrollable

У класі Scrollable було створено конструктор в якому створено власне полосу прокрутки Scrollbar та за допомогою методу pack() відображено цю полосу у вікні програми. (Рисунок 3.36)

```
scrollbar = Scrollbar(frame, width=width)
scrollbar.pack(side=RIGHT, fill=Y)
```

Рисунок 3.36 – Створення віджета Scrollbar

У Tkinter лише віджет Canvas - це власний прокручуваний контейнер. Це означає, що він може мати фактичний розмір, більший за розмір екрану, і користувач може переміщати область, яку він переглядаєте. Ось що насправді є прокрутка.

Синтаксис використання віджета Canvas наведено на Рисунку 3.37

```
w = Canvas ( master, option=value, ... )
```

Рисунок 3.37 – Синтаксис віджета Canvas

Параметри:

- master – представляє батьківське вікно;
- options – параметри цього віджета. Ці параметри можна використовувати як пари ключ-значення, розділені комами.

Деякі опції Canvas, які використовуються у програмі:

- width – розмір полотна в розмірності X;
- height – розмір полотна в розмірі Y;
- yscrollcommand - якщо Canvas можна прокручувати, цим атрибутом повинен бути метод set () вертикальної смуги прокрутки.

Тому необхідно створити Canvas, а всередині нього створити вікно. Це означає, що буде можливо встановити розмір Canvas, який завгодно, а потім, прокручуючи, користувач буди рухатися уздовж вікна всередині нього. За допомогою параметра yscrollcommand, який посилається на метод set(), слайдер буде переміщуватися (Рисунок 3.38)

```
self.canvas = Canvas(frame,height=1000,width=1500, yscrollcommand=scrollbar.set)
self.canvas.pack(side=LEFT,anchor=NW)
```

Рисунок 3.38 – Створення віджета Canvas

Далі необхідно налаштувати параметр command для створеної полоси прокрутки. (Рисунок 3.39)

```
scrollbar.config(command=self.canvas.yview)
```

Рисунок 3.39 Налаштування параметра command полоси прокрутки

Зв'язування з подією <Configure> допоможе правильно переналаштувати Canvas, коли розмір основного вікна змінюється. Метод C обробляє подію зміни розміру вікна і оновлює параметр width, що визначає ширину Canvas. За допомогою методу __fill_canvas буде збільшуватися елементи вікон до ширини Canvas. (Рисунок 3.40)

```
self.canvas.bind('<Configure>', self.__fill_canvas)

def __fill_canvas(self, event):
    "Enlarge the windows item to the canvas width"

    canvas_width = event.width
    self.canvas.itemconfig(self.windows_item, width = canvas_width)
```

Рисунок 3.40 – Збільшення елементів вікон по ширині Canvas

Наступний крок – було додано Frame за допомогою методу create_window(). Перший аргумент - положення, де потрібно розмістити віджет, який в свою чергу передається в аргументі window. Оскільки осі x і y віджета розміщуються в верхньому лівому кутку, розмістимо віджет в положенні (0, 0) і вирівняємо його в цьому кутку за допомогою anchor=tk.NW(північний захід). (Рисунок 3.41)

```
# assign this obj (the inner frame) to the windows item of the canvas
self.windows_item = self.canvas.create_window(0,0, window=self,anchor=NW)
```

Рисунок 3.41 – Створення внутрішньої рамки в Canvas

Метод update() оновлює Canvas та полоси прокрутки. Для отримання реального розміру контейнера потрібно зробити так, щоб geometry manager прорисовував всі дочірні віджети в першу чергу за допомогою виклику update_idletasks(). Цей віджет доступний у всіх класах віджета і він відповідає за

те, щоб Tkinter обробив всі події в процесі очікування: наприклад, перемальовування або нові обчислення розмірів (Рисунок 3.42)

```
def update(self):
    "Update the canvas and the scrollregion"

    self.update_idletasks()
    self.canvas.config(scrollregion=self.canvas.bbox(self.windows_item))
```

Рисунок 3.42 – Оновлення Canvas

root.mainloop()- це метод у головному вікні, який ми запускаємо, коли хочемо запустити наш додаток. Цей метод буде циклічно вічно, чекаючи подій від користувача, поки користувач не вийде з програми - або закривши вікно, або завершивши програму з перериванням клавіатури в консолі. (Рисунок 3.43)

```
root.mainloop()
```

Рисунок 3.43 – Метод запуску тесту

3.5 Тестування програми

Тестування програмного забезпечення необхідне, оскільки всі люди допускають помилки. Деякі з цих помилок є неважливими, але деякі з них є дорогими або небезпечними. Потрібно перевіряти все, що виробляється, тому що все може піти не так - люди постійно помиляються .

Тестування програмного забезпечення є найважливішим компонентом розробки програмного продукту, оскільки воно покращує узгодженість та продуктивність. Основною перевагою тестування є виявлення та подальше усунення помилок. Проте тестування також допомагає розробникам та тестувальникам порівнювати фактичні та очікувані результати з метою покращення якості. Якщо виробництво програмного забезпечення відбувається без його тестування, воно може виявитися марним або часом небезпечним для клієнтів.

Тому було проведено тестування цієї програми. В ході тестування були виявлені наступні помилки.

У даній програмі є лише одне поле для введення інформації – Entry, яке призначене для того щоб дізнатися вік користувача. Відповідно у це поле можуть вводитися лише цифри. Якщо користувач ввів випадково у це поле букву чи будь який інший символ програма показувала помилку (Рисунок 3.44) та припиняла свою роботу.

```
ValueError: invalid literal for int() with base 10: 'f'
```

Рисунок 3.44 – Помилка ValueError – вік не заповнений цифрами

Python ValueError виникає, коли функція отримує аргумент правильного типу, але невідповідне значення.

Дане помилку було оброблено логічним оператором if ... else. Який за допомогою метода isdigit () - це вбудований метод, який використовується для обробки рядків. Метод isdigit () повертає значення "True", якщо всі символи в рядку є цифрами, інакше він повертає значення "False".

Для того щоб, користувачу було більш зрозуміло, яку помилку він допустив та як її витравити було застосовано віджет MessageBox.

Віджет MessageBox використовується для відображення вікон повідомлень у програмах python. Цей модуль використовується для відображення повідомлення, використовуючи ряд функцій.

Деякі з цих функцій - showinfo, showwarning, showerror, askquestion, askokcancel, askyesno та askretryignore.

Синтаксис для створення цього віджета – Рисунок 3.45

```
tkMessageBox.FunctionName(title, message [, options])
```

Рисунок 3.45 – Синтаксис віджета MessageBox

Параметри:

- FunctionName - це ім'я відповідної функції вікна повідомлення.
- title - це текст, який відображатиметься в рядку заголовка вікна повідомлення.

- message - це текст, який відображатиметься як повідомлення.
- options - це альтернативні варіанти, які ви можна використовувати для пристосування стандартного вікна повідомлення.

Для початку необхідно імпортувати цей віджет у програму (Рисунок 3.2). Потім застосувати цей віджет у програмі (Рисунок 3.46).

```
s = entry.get()

if not s.isdigit():
    mb.showerror(
        "Помилка",
        "Введіть будь ласка цифрами ваш вік!")
    l.append(0)
else:
    l.append(int(s))
```

Рисунок 3.46 – Усунення помилки – вік не заповнений цифрами

На Рисунку 3.47 зображено вікно, як воно відображається у програмі.

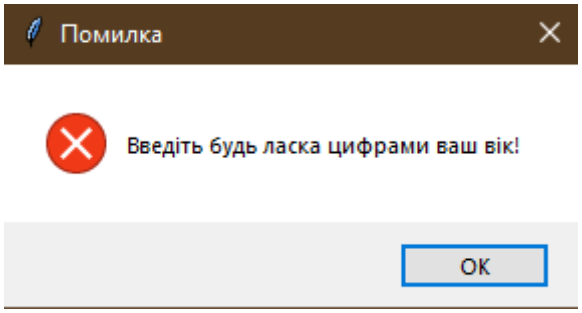


Рисунок 3.47 – Вікно помилки - вік не заповнений цифрами

Також було виявлено іншу помилку – не всі заповнені поля. Наприклад, якщо користувач міг забути відповісти на одне із питань тесту, тоді програма показувала помилку (Рисунок 3.48) і також припиняла свою роботу.

```
ValueError: feature_names mismatch: ['f0', 'f1', 'f2', 'f3', 'f4', 'f5', 'f6', 'f7', 'f8', 'f9', 'f10', 'f11', 'f12',
'f13', 'f14', 'f15'] ['f0', 'f1', 'f2', 'f3', 'f4', 'f5', 'f6', 'f7', 'f8', 'f9', 'f10', 'f11', 'f12', 'f13', 'f14']
expected f15 in input data
```

Рисунок 3.48 – Помилка ValueError – не всі поля заповнені

Для усунення цієї помилки було застосовано try and except.

Оператор Try and Except використовується для обробки помилок у коді на Python. Блок try використовується для перевірки деякого коду на наявність помилок, тобто код всередині блоку try буде виконуватися, коли в програмі немає помилок. Синтаксис цього оператора – Рисунок 3.49

```
try:
    # Some Code
except:
    # Executed if error in the
    # try block
```

Рисунок 3.49 – Синтаксис оператора Try and Except

В блоці except потрібно обробити дану помилку за допомогою віджета https://studfile.net/preview/5010027/page:2/ (Рисунок 3.50).

```
except:
    mb.showerror(
        "Помилка",
        "Не всі поля заповнені!")
```

Рисунок 3.50 Усунення помилки – не всі поля заповнені

На Рисунку 3.51 зображено вікно, як воно відображається у програмі.

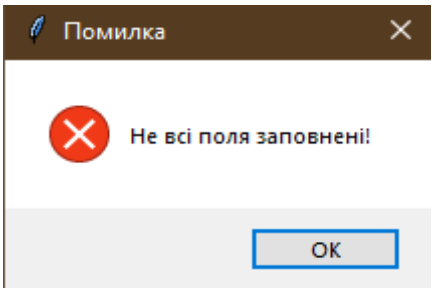


Рисунок 3.51 – Вікно помилки - не всі поля заповнені

3.6 Етапи проходження тесту

Помилки знайдені та усунені, можна проходити тест.

На першому етапі необхідно запустити програму. Після запуску програми з’являється вікно з питаннями до тесту для поставлення діагнозу цукровий діабет на ранніх стадіях.(Рисунок 3.52)

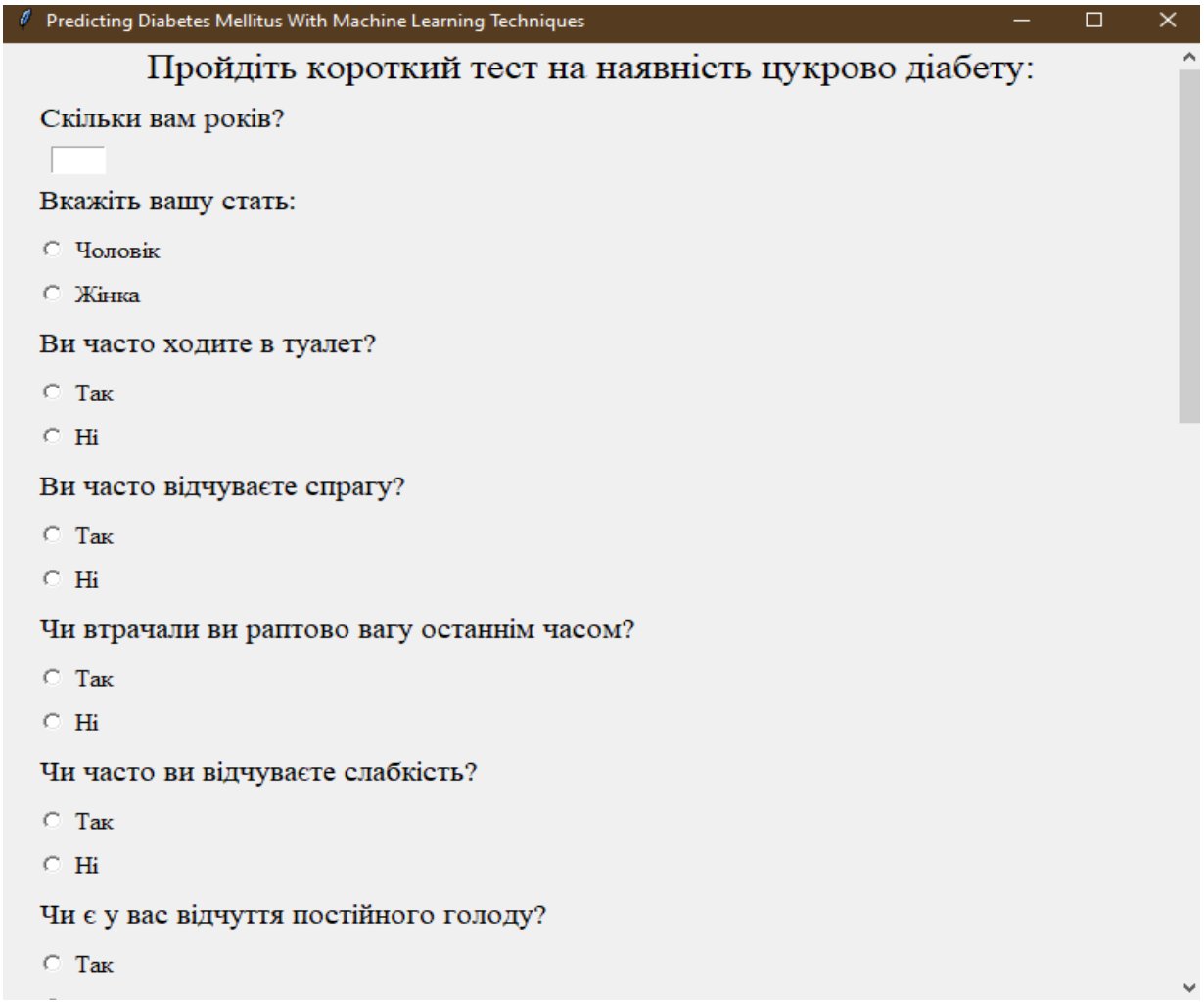


Рисунок 3.52 – Вікно програми

Якщо програму розгорнути на весь екран, то всі елементи розташуються відповідно то нового розширення екрану.(Рисунок 3.53 та 3.54)

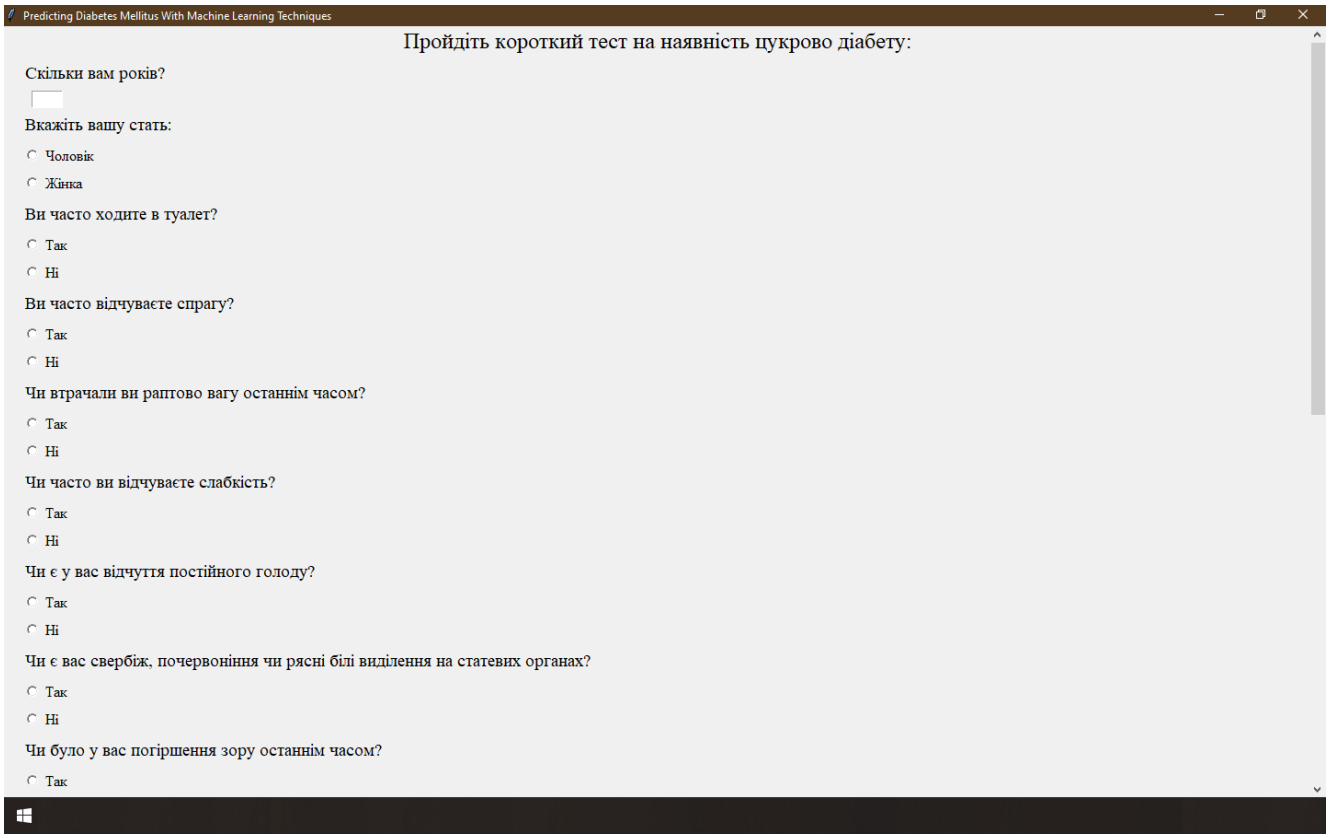


Рисунок 3.53 – Вікно програми у повному розмірі – верхня частина

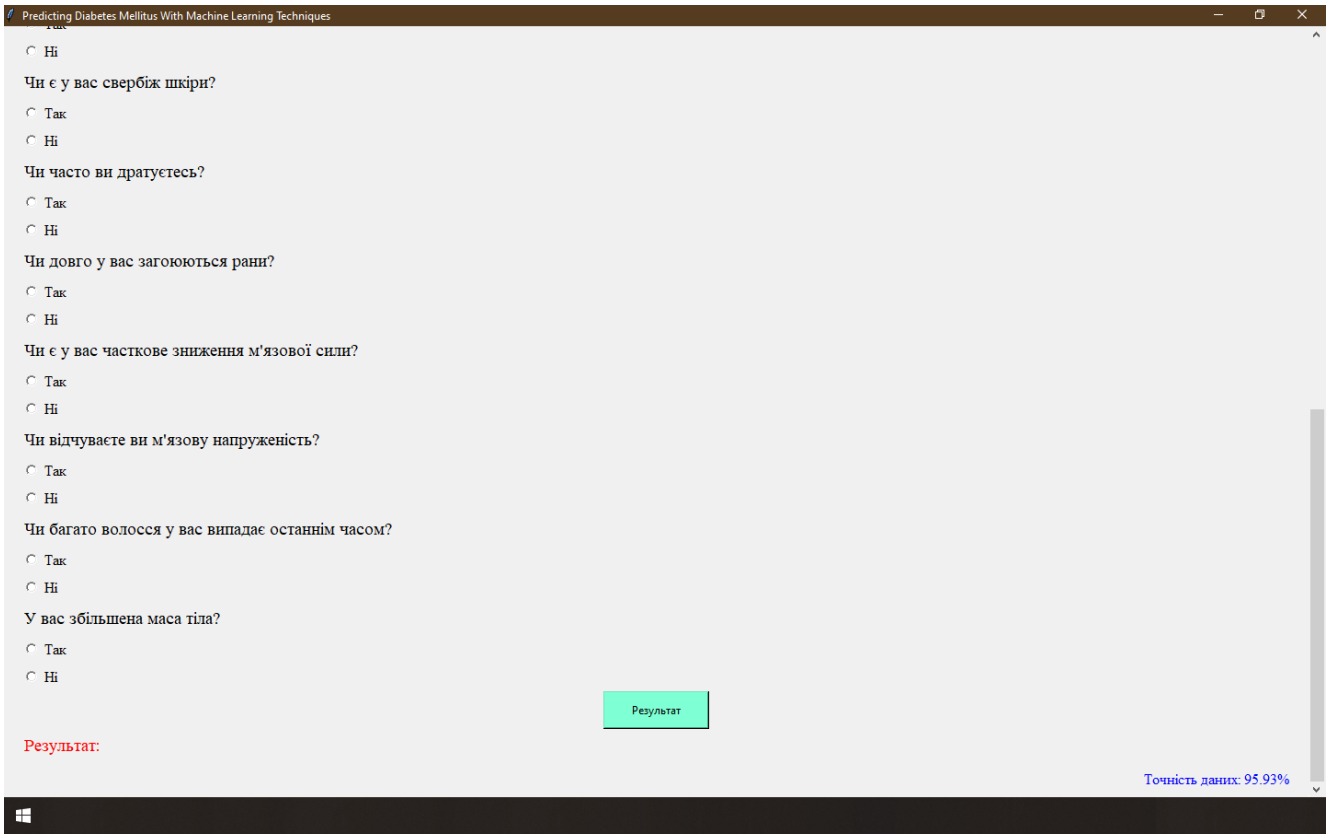


Рисунок 3.54 – Вікно програми у повному розмірі – нижня частина

На другому етапі необхідно заповнити поле вік (вписавши цифрами) та вказати стать (вибравши із запропонованих варіантів) – Рисунок 3.55

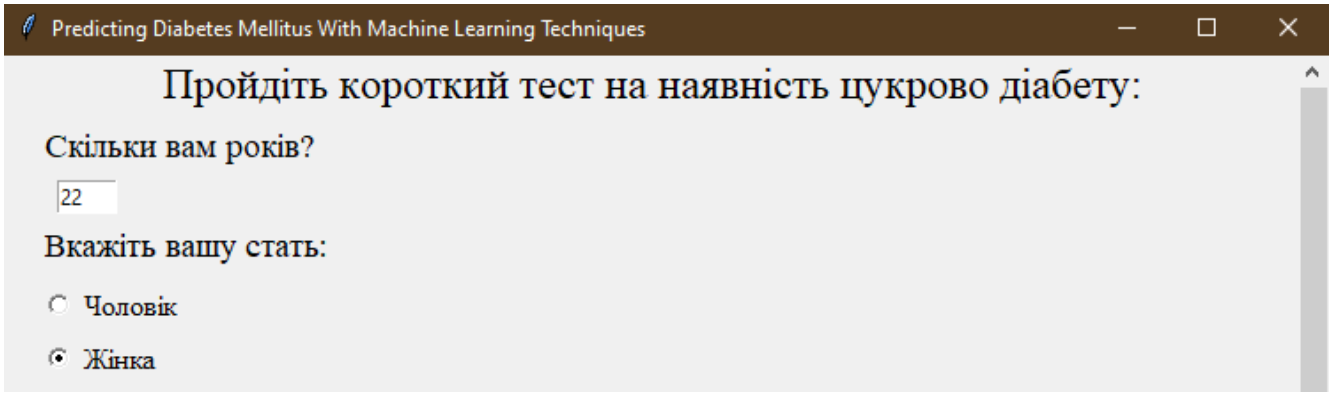


Рисунок 3.55 – Заповнення поля – вік

На третьому етапі необхідно дати відповіді на всі питання, які залишилися (вибираючи відповідь Так або Ні) – Рисунок 3.56

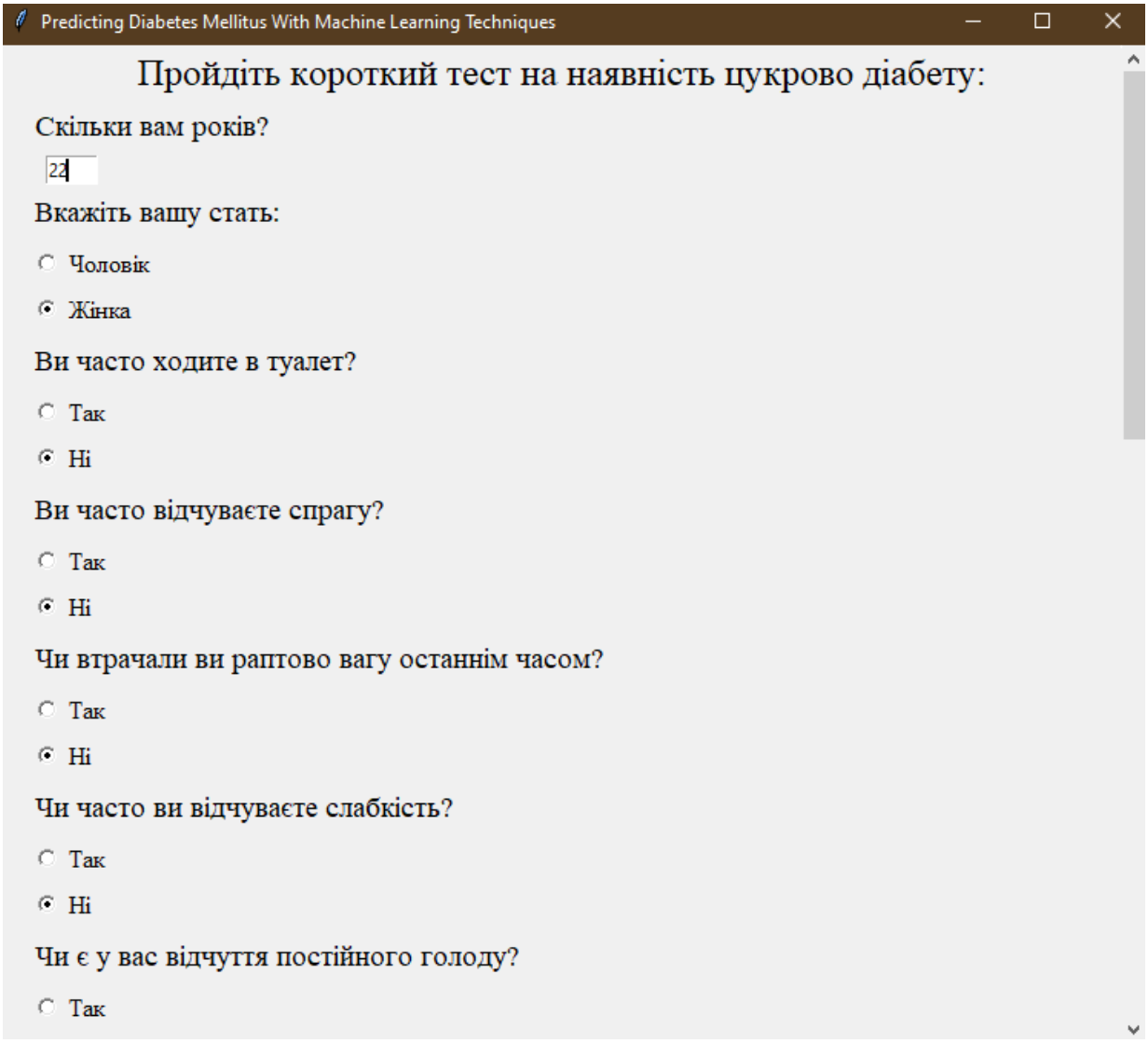


Рисунок 3.56 – Відповіді на питання

Для того щоб, побачити інші запитання необхідно за допомогою полоси прокрутки прогорнути вміст вікна – Рисунок 3.57

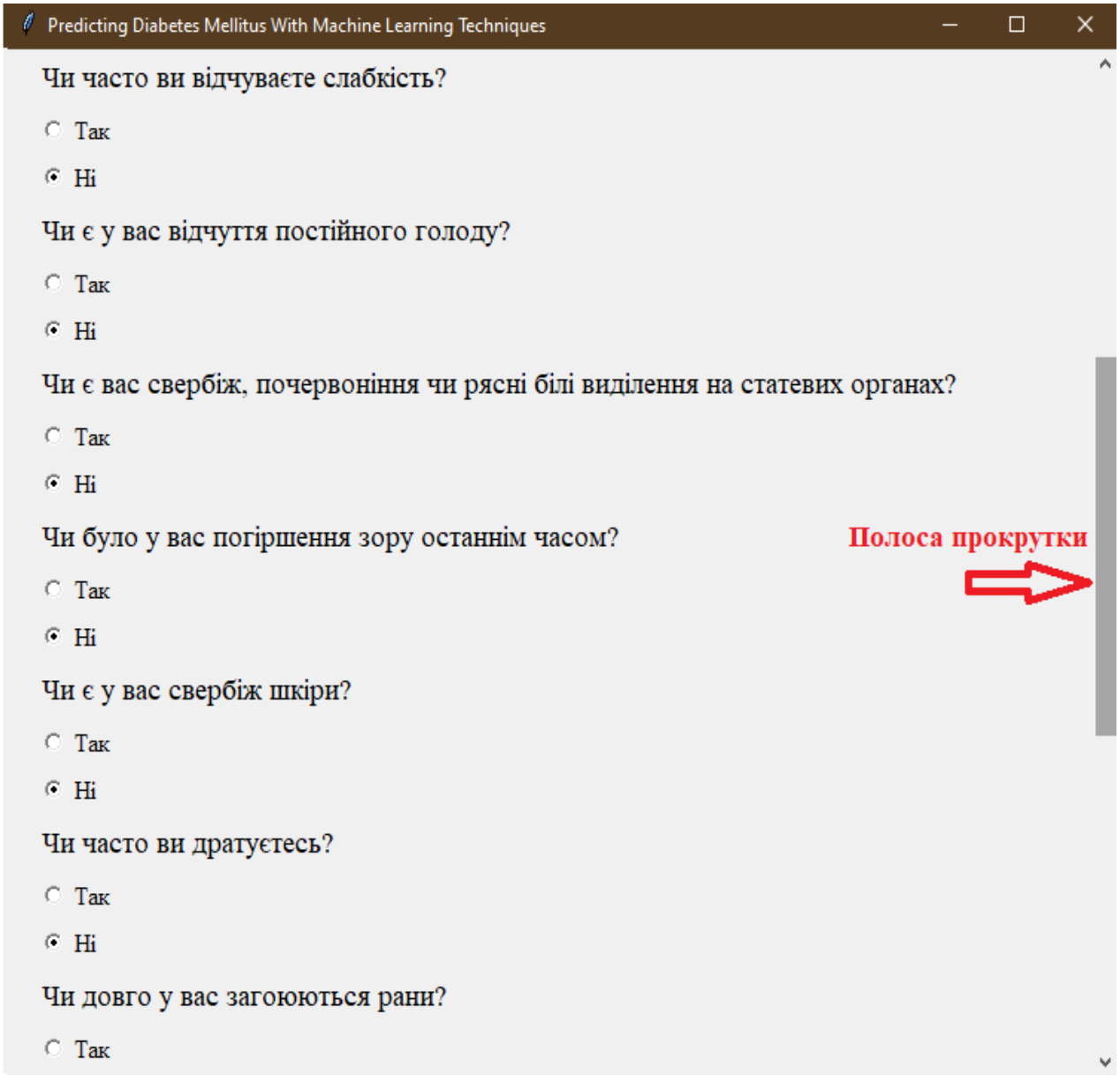


Рисунок 3.57 – Прогортання вмісту вікна

Після того, як користувач дав відповіді на всі запитання йде третій етап перевірки інформації. Після натиснення на кнопку «Результат», дані внесенні користувачем перевіряються на наявність помилок (вік заповнений цифрами та чи всі поля заповнені). Якщо помилки знайдені про це користувача повідомляють відповідні вікна (Рисунок 3.47 та 3.51). Далі користувач виправляє зроблені помилки та знову натиска кнопку «Результат».

Якщо ж помилок немає то починається четвертий етап – постановка діагнозу. У полі результат програма відображає діагноз на основі відповідей

користувача за допомогою раніше навченої програми алгоритмом штучного інтелекту XGBoost. (Рисунок 3.58 та 3.59)

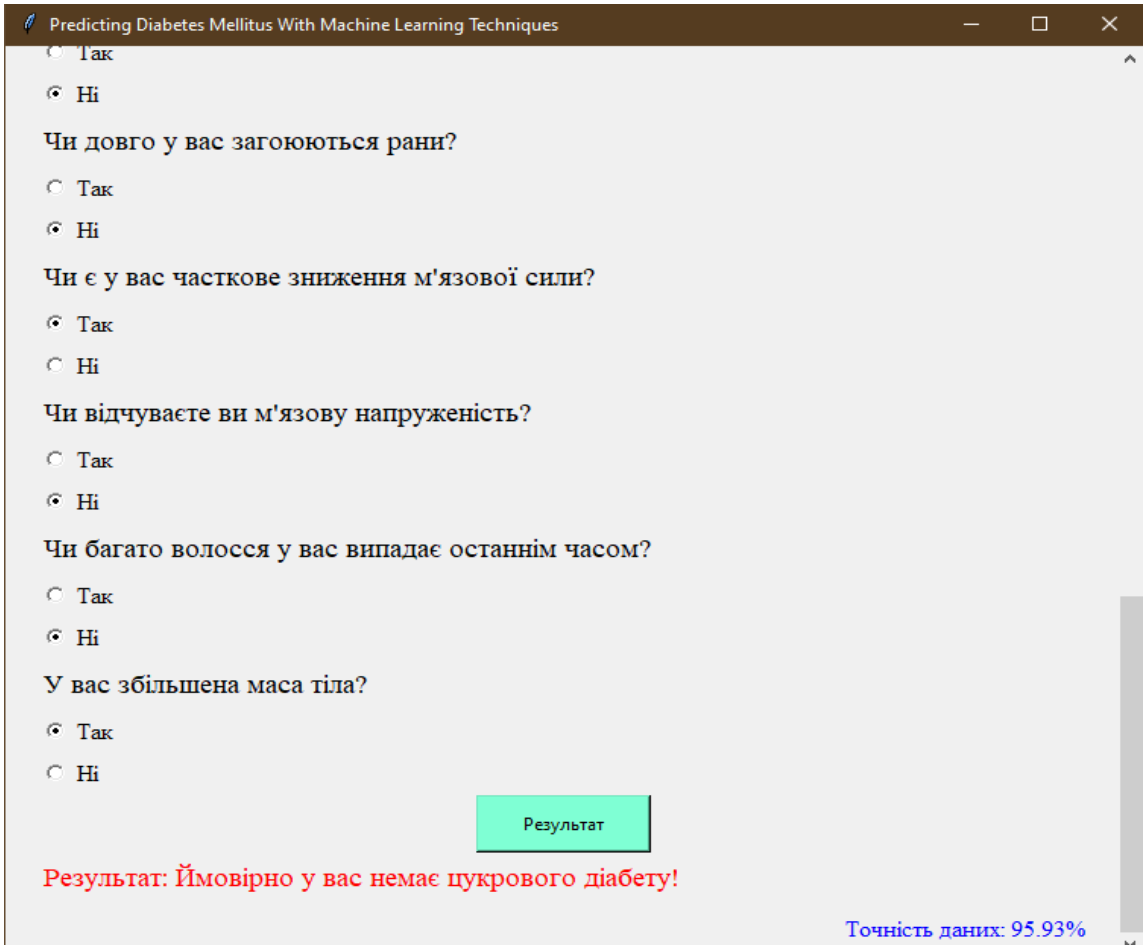


Рисунок 3.58 – Негативний діагноз

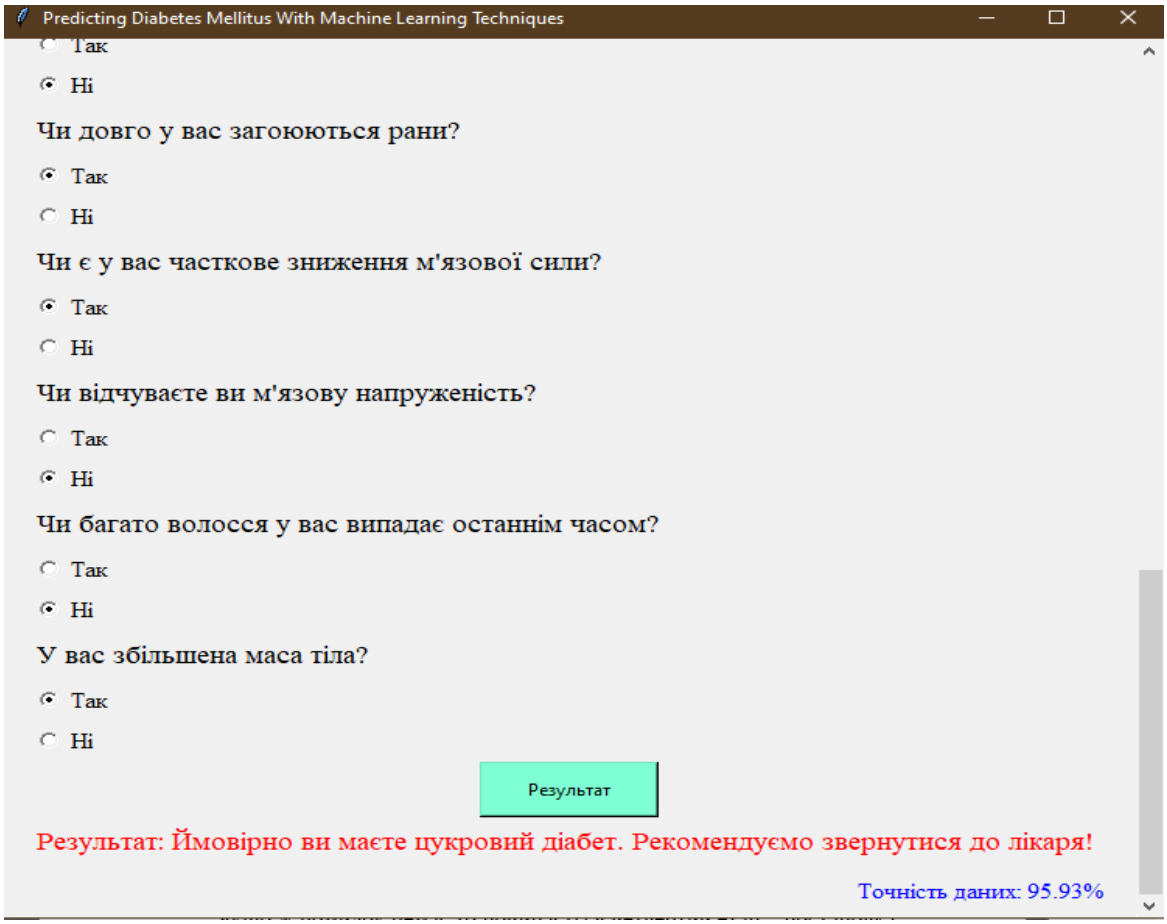


Рисунок 3.59 – Позитивний діагноз

На Рисунку 3.60, 3.61, 3.62, 3.63 відображена схема, для кращого розуміння коду та орієнтування по-ньому.

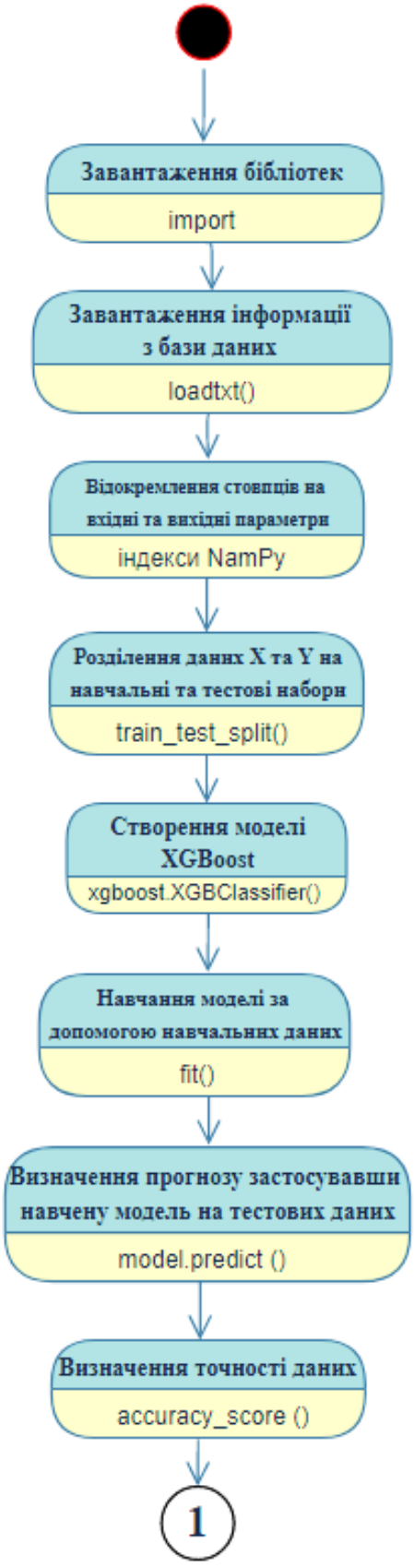


Рисунок 3.60 Схема написання коду

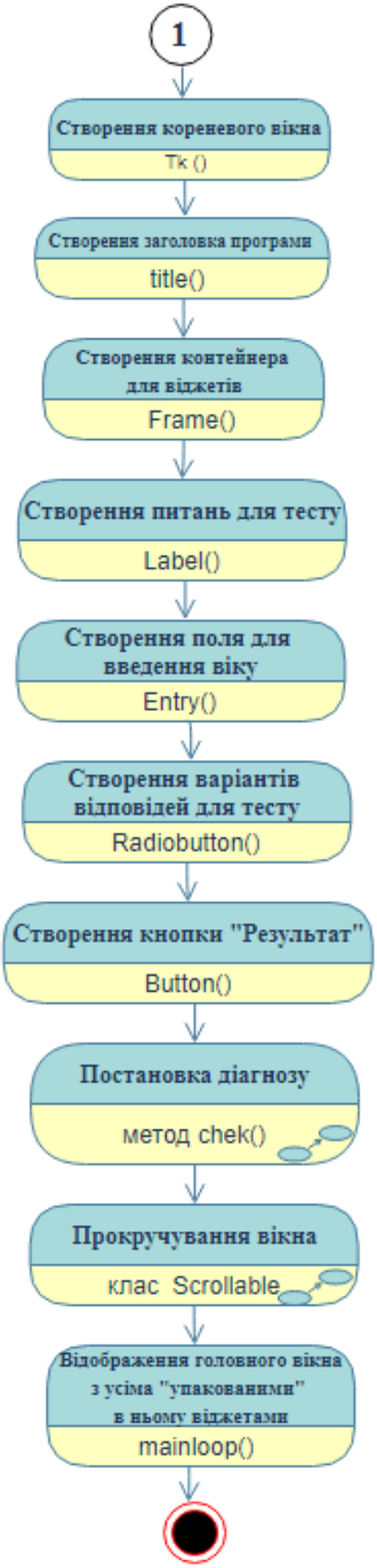


Рисунок 3.61.Продовження схеми написання коду

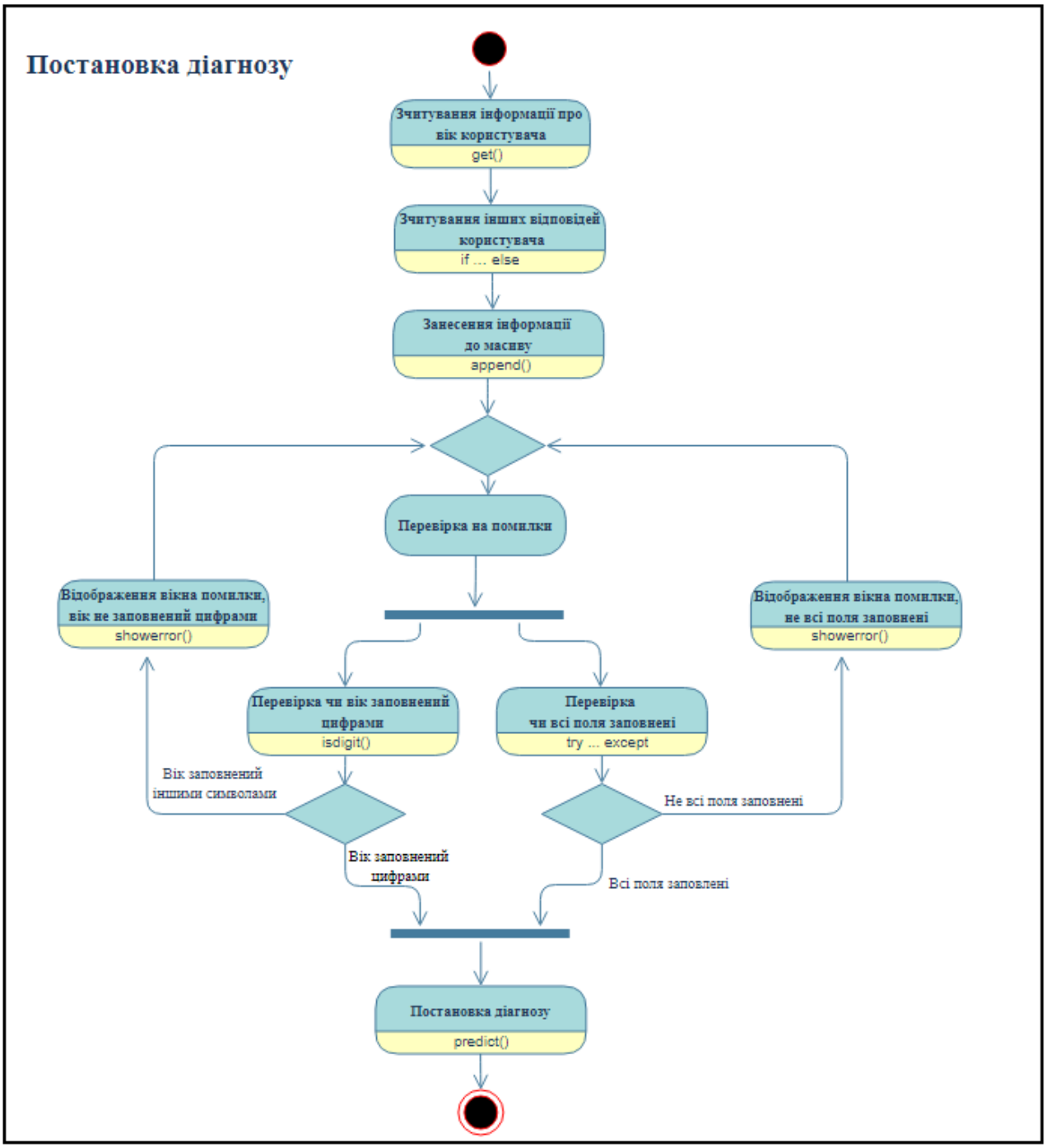


Рисунок 3.62 – Декомпозиція постановки діагнозу схеми написання коду

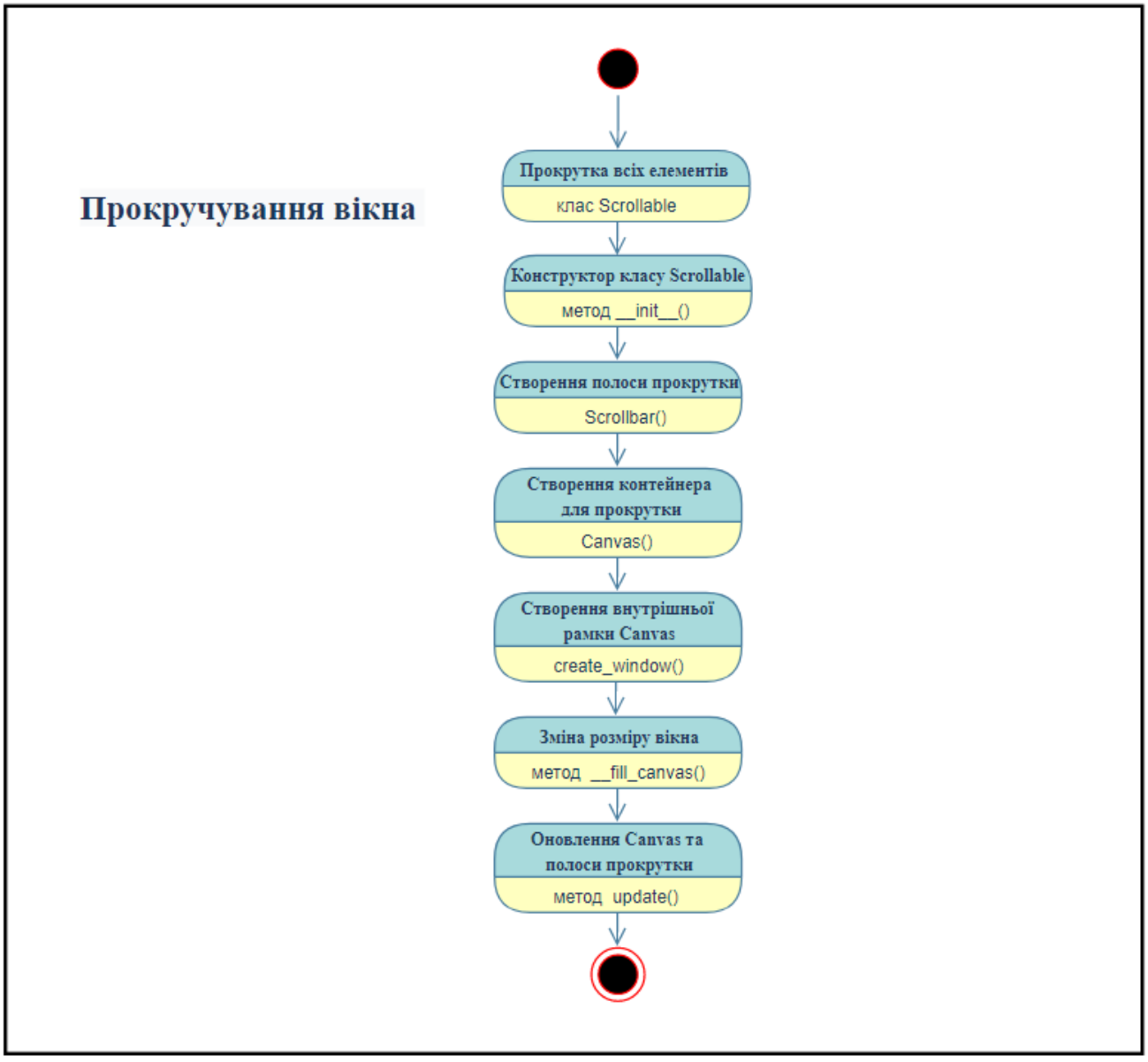


Рисунок 3.63 – Декомпозиція прокручування вікна схеми написання коду

ВИСНОВКИ

У цьому дослідженні було виявлено підходи машинного навчання для аналізу даних щодо прогнозування діагностики цукрового діабету. Цукровий діабет швидко стає однією з найбільших проблем охорони здоров'я у XXI столітті.

Вибір даних є цінним активом для вирішення рясних клінічних даних, зібраних від пацієнтів та отриманих у результаті досліджень та лікування цукрового діабету.

1. Проаналізовано популярні методи машинного навчання, створено таблицю їх переваг та недоліків.
2. Визначено найкращий метод машинного навчання. Результати показали, що алгоритм XGBoost має найвищу точність порівняно з іншими вивченими методами. Він має найкращі показники серед інших алгоритмів і запроваджений як найефективніший метод прогнозування ймовірності діагностики цукрового діабету на ранніх стадіях.
3. Створено алгоритм роботи програми в нотації діаграми діяльності.
4. Було обрано та встановлено всі необхідні бібліотеки для подальшого створення програми.
5. Дані для дослідження були отримані від 520 пацієнтів Sylhet Diabetes Hospital у Сілхеті, Бангладеш, шляхом прямих опитувальних листів пацієнтів у 2020 році та схвалено лікарем.
6. За допомогою моделі XGBoost було навчено програму для прогнозування діагнозу на основі отриманої бази даних.
7. Використовуючи імпортовані бібліотеки було створено зручний та простий у використанні інтерфейс користувача.
8. Проведено тестування розробленої програми. У ході тестування було виявлено, оброблено та усунуто усі знайдені помилки.

Перевагою даної програми є те що, користувач на основі власних даних щоденного фізичного обстеження, може швидко та легко дізнатися ймовірність діагностики цукрового діабету на ранніх стадіях.

ПЕРЕЛІК ПОСИЛАНЬ

1) Aiswarya Iyer, S. Jeyalatha and Ronak Sumbaly,” Diagnosis of Diabetes Using Classification Mining Techniques”, International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol.5, No.1, January 2015.

2) B.M. Patil, R.C. Joshi and Durga Toshniwal,”Association Rule for Classification of Type-2 Diabetic Patients”, ICMLC '10 Proceedings of the 2010 Second International Conference on Machine Learning and Computing, February 09 - 11, 2010.

3) B. Nithya and Dr. V. Ilango,” Predictive Analytics in Health Care Using Machine Learning Tools and Techniques”, International Conference on Intelligent Computing and Control Systems, 978-1-5386-2745-7,2017.

4) Cha Zhang , Yunqian Ma, Ensemble Machine Learning: Methods and Applications, 2012.

5) Control Flow Tools – Python [Інтернет-портал]. – Режим доступу: https://docs.python.org/3/tutorial/controlflow.html.

6) Dost Muhammad Khan1, Nawaz Mohamudally2, “An Integration of K-means and Decision Tree (ID3) towards a more Efficient Data Mining Algorithm ”, Journal Of Computing, Volume 3, Issue 12, December 2011.

7) Dr Saravana kumar N M, Eswari T, Sampath P and Lavanya S,” Predictive Methodology for Diabetic Data Analysis in Big Data”, 2nd International Symposium on Big Data and Cloud Computing,2015.

8) Gauri D. Kalyankar, Shivananda R. Poojara and Nagaraj V. Dharwadkar,” Predictive Analysis of Diabetic Patient Data Using Machine Learning and Hadoop”, International Conference On I-SMAC,978-1-5090-3243-3,2017.

9) Graphical User Interfaces with Tk - Python [Інтернет-портал]. – Режим доступу: https://docs.python.org/3/library/tk.html.

10) Humar Kahramanli and Novruz Allahverdi,”Design of a Hybrid System for the Diabetes and Heart Disease”, Expert Systems with Applications: An International Journal, Volume 35 Issue 1-2, July, 2008.

11) Ian H. Witten, Eibe Frank, Mark A. Hall, Christopher J. Pal, Data Mining: Practical Machine Learning Tools and Techniques (Morgan Kaufmann Series in Data Management Systems) 4th Edition, 2016.

12) Introduction to Boosted Treesv - XGBoost [Електронний ресурс]. – Режим доступу: https://xgboost.readthedocs.io/en/latest/tutorials/model.html.

13) Islam, MM Faniqul, et al. 'Likelihood prediction of diabetes at early stage using data mining techniques.' Computer Vision and Machine Intelligence in Medical Image Analysis. Springer, Singapore, 2020, 113-125.

14) Jayalakshmi T, Santhakumaran A (2010) A novel classification method for diagnosis of diabetes mellitus using artificial neural networks. In: 2010 International conference on data storage and data engineering. IEEE Computer Society, pp 159–163

15) K. Rajesh and V. Sangeetha, "Application of Data Mining Methods and Techniques for Diabetes Diagnosis", International Journal of Engineering and Innovative Technology (IJEIT) Volume 2, Issue 3, September 2012.

16) Mathworks. *What Is Machine Learning?*, Режим доступу: https://www.mathworks.com/discovery/machine-learning.html.

17) Osisanwo F.Y, T. AkinsolaJ.E., O. Awodele, O. HinmikaiyeJ., O. Olakanmi, J. Akinjobi, International Journal of Computer Trends and Technology, International Journal of Computer Trends and Technology, 2017.

18) Perveen S, Shahbaz M, Guergachi A, Keshavjee K (2016) Performance analysis of data mining classification techniques to predict diabetes. Procedia Procedia Comput Sci 82:115–121

19) Python - Error Types – TutorialsTeacher [Інтернет-портал]. – Режим доступу: https://www.tutorialsteacher.com/python/error-types-in-python.

20) Python Tkinter Canvas Tutorial - PythonGuides [Інтернет-портал]. – Режим доступу: https://pythonguides.com/python-tkinter-canvas/.

21) Python Tkinter – Frame Widget - GeeksforGeeks [Інтернет-портал]. – Режим доступу: https://www.geeksforgeeks.org/python-tkinter-frame-widget/.

22) Scrollable Frames in Tkinter - Teclado [Інтернет-портал]. – Режим доступу: https://blog.teclado.com/tkinter-scrollable-frames/.

23) Tianqi Chen, Carlos Guestrin, XGBoost: A Scalable Tree Boosting System, 10.1145/2939672.2939785, 2016.

24) Tkinter buttons - PythonProgramming [Інтернет-портал]. – Режим доступу: https://pythonprogramming.net/tkinter-python-3-tutorial-adding-buttons/.

25) Tkinter messagebox - Pythonbasics [Інтернет-портал]. – Режим доступу: https://pythonbasics.org/tkinter-messagebox/.

26) Trevor Hastie, Robert Tibshirani , Jerome Friedman The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics) 2nd Edition, 2016

27) UCI (2020) Набір даних про цукровий діабет. Режим доступу: https://archive.ics.uci.edu/ml/datasets/Early+stage+diabetes+risk+prediction+dataset.

28) Vijayan V, Ravi K (2015) Prediction and diagnosis of diabetes mellitus—a machine learning approach, December, pp 122–127

29) What is Python? – Python [Інтернет-портал]. – Режим доступу: https://www.python.org/doc/essays/blurb.

30) Zhi-Hua Zhou, Chapman & Hall/CRC Data Mining and Knowledge Discovery Serie, Ensemble Methods: Foundations and Algorithms, 2012.

31) Ендокринологія. Підручник \ За ред. проф. П.М. Боднара. – Нова Книга. – Вінниця. – 2010. – 464 с

32) Скачко, Б. Г. Цукровий діабет: хвороба століття, чи розплата за легковажність [Текст] / Б. Г. Скачко, Г. О. Орещук. - К. : Здоров'я, 2012. - 96 с. - ISBN 978-966-463-039-6

33) Умовні позначення діаграми діяльності -StudeFile [Інтернет-портал]. – Режим доступу: https://studfile.net/preview/5010027/page:2/.

ДОДАТОК А

КОД ПРОГРАМИ

```
import xgboost
import numpy
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from tkinter import *
from tkinter import messagebox as mb


class Scrollable(Frame):
   """
      Make a frame scrollable with scrollbar on the right.
      After adding or removing widgets to the scrollable frame,
      call the update() method to refresh the scrollable area.
   """

   def __init__(self, frame, width=16):

      scrollbar = Scrollbar(frame, width=width)
      scrollbar.pack(side=RIGHT, fill=Y)

      self.canvas = Canvas(frame,height=1000,width=1500,
yscrollcommand=scrollbar.set)
      self.canvas.pack(side=LEFT,anchor=NW)

      scrollbar.config(command=self.canvas.yview)

      self.canvas.bind('<Configure>', self.__fill_canvas)

      # base class initialization
```

```
        Frame.__init__(self, frame)

        # assign this obj (the inner frame) to the windows item of the canvas
        self.windows_item = self.canvas.create_window(0,0, window=self,anchor=NW)



    def __fill_canvas(self, event):
        "Enlarge the windows item to the canvas width"

        canvas_width = event.width
        self.canvas.itemconfig(self.windows_item, width = canvas_width)

    def update(self):
        "Update the canvas and the scrollregion"

        self.update_idletasks()
        self.canvas.config(scrollregion=self.canvas.bbox(self.windows_item))



# load data
dataset = numpy.loadtxt('diabetes.csv', delimiter=",")

# split data into X and y
X1 = dataset[:,0:16]
Y1 = dataset[:,16]

# split data into train and test sets
seed = 17
test_size = 0.33
```

```
X_train, X_test, y_train, y_test = train_test_split(X1, Y1, test_size=test_size,
random_state=seed)

# fit model no training data
model = xgboost.XGBClassifier()
model.fit(X_train, y_train)

# make predictions for test data
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]

# evaluate predictions
accuracy = accuracy_score(y_test, predictions)*100.0

def check():
    s = entry.get()
    if not s.isdigit():
        mb.showerror(
            "Помилка",
            "Введіть будь ласка цифрами ваш вік!")
        l.append(0)
    else:
        l.append(int(s))
    try:
        if var.get() == 1:
            l.append(1)
        elif var.get() == 2:
            l.append(2)
        if var1.get() == 3:
            l.append(1)
```

```
elif var1.get() == 4:
    l.append(0)
if var2.get() == 5:
    l.append(1)
elif var2.get() == 6:
    l.append(0)
if var3.get() == 7:
    l.append(1)
elif var3.get() == 8:
    l.append(0)
if var4.get() == 9:
    l.append(1)
elif var4.get() == 10:
    l.append(0)
if var5.get() == 11:
    l.append(1)
elif var5.get() == 12:
    l.append(0)
if var6.get() == 13:
    l.append(1)
elif var6.get() == 14:
    l.append(0)
if var7.get() == 15:
    l.append(1)
elif var7.get() == 16:
    l.append(0)
if var8.get() == 17:
    l.append(1)
elif var8.get() == 18:
    l.append(0)
```

```
if var9.get() == 19:
    l.append(1)
elif var9.get() == 20:
    l.append(0)
if var10.get() == 21:
    l.append(1)
elif var10.get() == 22:
    l.append(0)
if var11.get() == 23:
    l.append(1)
elif var11.get() == 24:
    l.append(0)
if var12.get() == 25:
    l.append(1)
elif var12.get() == 26:
    l.append(0)
if var13.get() == 27:
    l.append(1)
elif var13.get() == 28:
    l.append(0)
if var14.get() == 29:
    l.append(1)
elif var14.get() == 30:
    l.append(0)
X_test1 = numpy.array([l])
y_pred1 = model.predict(X_test1)
predictions1 = [round(value) for value in y_pred1]
if predictions1[0] == 0:
    l17['text'] = "Результат: Ймовірно у вас немає цукрового діабету!"
else:
```

```
        l17['text'] = "Результат: Ймовірно ви маєте цукровий діабет. Рекомендуємо звернутися до лікаря! "

    except:
        mb.showerror(
            "Помилка",
            "Не всі поля заповнені!")
    l.clear()

l = []

root = Tk()
root.geometry('750x650+10+10')
root.title("Predicting Diabetes Mellitus With Machine Learning Techniques")

body=Frame(root,height=400)
body.pack()

scrollable_body = Scrollable(body, width=18)

entry = Entry(scrollable_body, width=5)

l0 = Label(scrollable_body,text = "Пройдіть короткий тест на наявність цукрово діабету:", font = ("Times New Roman", 18))
l1 = Label(scrollable_body,text = "Скільки вам років?", font = ("Times New Roman", 14))
l2 = Label(scrollable_body,text = "Вкажіть вашу стать:", font = ("Times New Roman", 14))
l3 = Label(scrollable_body,text = "Ви часто ходите в туалет?", font = ("Times New Roman", 14))
```

```
l4 = Label(scrollable_body,text = "Ви часто відчуваєте спрагу?", font = ("Times New Roman", 14))
l5 = Label(scrollable_body,text = "Чи втрачали ви раптово вагу останнім часом?", font = ("Times New Roman", 14))
l6 = Label(scrollable_body,text = "Чи часто ви відчуваєте слабкість?", font = ("Times New Roman", 14))
l7 = Label(scrollable_body,text = "Чи є у вас відчуття постійного голоду?", font = ("Times New Roman", 14))
l8 = Label(scrollable_body,text = "Чи є вас свербіж, почервоніння чи рясні білі виділення на статевих органах?", font = ("Times New Roman", 14))
l9 = Label(scrollable_body,text = "Чи було у вас погіршення зору останнім часом?", font = ("Times New Roman", 14))
l10 = Label(scrollable_body,text = "Чи є у вас свербіж шкіри?", font = ("Times New Roman", 14))
l11 = Label(scrollable_body,text = "Чи часто ви дратуєтесь?", font = ("Times New Roman", 14))
l12 = Label(scrollable_body,text = "Чи довго у вас загоюються рани?", font = ("Times New Roman", 14))
l13 = Label(scrollable_body,text = "Чи є у вас часткове зниження м'язової сили?", font = ("Times New Roman", 14))
l14 = Label(scrollable_body,text = "Чи відчуваєте ви м'язову напруженість?", font = ("Times New Roman", 14))
l15 = Label(scrollable_body,text = "Чи багато волосся у вас випадає останнім часом?", font = ("Times New Roman", 14))
l16 = Label(scrollable_body,text = "У вас збільшена маса тіла?", font = ("Times New Roman", 14))
l17 = Label(scrollable_body,text = "Результат:", font = ("Times New Roman", 14), fg="red")
l18 = Label(scrollable_body,text = "Точність даних: %.2f%%" % accuracy, font = ("Times New Roman", 12), fg="blue")
```

```
var = IntVar()
var.set(0)
b1 = Radiobutton(scrollable_body,text="Чоловік",font = ("Times New Roman", 12),
                variable=var, value=1)
b2 = Radiobutton(scrollable_body,text="Жінка",font = ("Times New Roman", 12),
                 variable=var, value=2)

var1 = IntVar()
var1.set(0)
b3 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
                variable=var1, value=3)
b4 = Radiobutton(scrollable_body,text="Ні",font = ("Times New Roman", 12),
                 variable=var1, value=4)

var2 = IntVar()
var2.set(0)
b5 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
                variable=var2, value=5)
b6 = Radiobutton(scrollable_body,text="Ні",font = ("Times New Roman", 12),
                 variable=var2, value=6)

var3 = IntVar()
var3.set(0)
b7 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
                variable=var3, value=7)
b8 = Radiobutton(scrollable_body,text="Ні",font = ("Times New Roman", 12),
                 variable=var3, value=8)

var4 = IntVar()
```

```
var4.set(0)
b9 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
                variable=var4, value=9)
b10 = Radiobutton(scrollable_body,text="Ні",font = ("Times New Roman", 12),
                 variable=var4, value=10)

var5 = IntVar()
var5.set(0)
b11 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
                variable=var5, value=11)
b12 = Radiobutton(scrollable_body,text="Ні",font = ("Times New Roman", 12),
                 variable=var5, value=12)

var6 = IntVar()
var6.set(0)
b13 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
                variable=var6, value=13)
b14 = Radiobutton(scrollable_body,text="Ні",font = ("Times New Roman", 12),
                 variable=var6, value=14)

var7 = IntVar()
var7.set(0)
b15 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
                variable=var7, value=15)
b16 = Radiobutton(scrollable_body,text="Ні",font = ("Times New Roman", 12),
                 variable=var7, value=16)

var8 = IntVar()
var8.set(0)
b17 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
```

```
                variable=var8, value=17)
b18 = Radiobutton(scrollable_body,text="Hi",font = ("Times New Roman", 12),
                  variable=var8, value=18)

var9 = IntVar()
var9.set(0)
b19 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
                variable=var9, value=19)
b20 = Radiobutton(scrollable_body,text="Hi",font = ("Times New Roman", 12),
                  variable=var9, value=20)

var10 = IntVar()
var10.set(0)
b21 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
                variable=var10, value=21)
b22 = Radiobutton(scrollable_body,text="Hi",font = ("Times New Roman", 12),
                  variable=var10, value=22)

var11 = IntVar()
var11.set(0)
b23 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
                variable=var11, value=23)
b24 = Radiobutton(scrollable_body,text="Hi",font = ("Times New Roman", 12),
                  variable=var11, value=24)

var12 = IntVar()
var12.set(0)
b25 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
                variable=var12, value=25)
b26 = Radiobutton(scrollable_body,text="Hi",font = ("Times New Roman", 12),
```

```
                variable=var12, value=26)

var13 = IntVar()
var13.set(0)
b27 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
              variable=var13, value=27)
b28 = Radiobutton(scrollable_body,text="Ні",font = ("Times New Roman", 12),
                variable=var13, value=28)

var14 = IntVar()
var14.set(0)
b29 = Radiobutton(scrollable_body,text="Так",font = ("Times New Roman", 12),
              variable=var14, value=29)
b30 = Radiobutton(scrollable_body,text="Ні",font = ("Times New Roman", 12),
                variable=var14, value=30)

but = Button(scrollable_body,text="Результат",width=15, height=2,command=check)
but['bg']='#7FFFD4'

l0.pack()
l1.pack(anchor=NW, padx=20,pady=5)
entry.pack(anchor=NW, padx=30)
l2.pack(anchor=NW, padx=20,pady=5)
b1.pack(anchor=NW,padx=20)
b2.pack(anchor=NW,padx=20)
l3.pack(anchor=NW, padx=20,pady=5)
b3.pack(anchor=NW,padx=20)
b4.pack(anchor=NW,padx=20)
l4.pack(anchor=NW, padx=20,pady=5)
b5.pack(anchor=NW,padx=20)
```

```
b6.pack(anchor=NW,padx=20)
l5.pack(anchor=NW, padx=20,pady=5)
b7.pack(anchor=NW,padx=20)
b8.pack(anchor=NW,padx=20)
l6.pack(anchor=NW, padx=20,pady=5)
b9.pack(anchor=NW,padx=20)
b10.pack(anchor=NW,padx=20)
l7.pack(anchor=NW, padx=20,pady=5)
b11.pack(anchor=NW,padx=20)
b12.pack(anchor=NW,padx=20)
l8.pack(anchor=NW, padx=20,pady=5)
b13.pack(anchor=NW,padx=20)
b14.pack(anchor=NW,padx=20)
l9.pack(anchor=NW, padx=20,pady=5)
b15.pack(anchor=NW,padx=20)
b16.pack(anchor=NW,padx=20)
l10.pack(anchor=NW, padx=20,pady=5)
b17.pack(anchor=NW,padx=20)
b18.pack(anchor=NW,padx=20)
l11.pack(anchor=NW, padx=20,pady=5)
b19.pack(anchor=NW,padx=20)
b20.pack(anchor=NW,padx=20)
l12.pack(anchor=NW, padx=20,pady=5)
b21.pack(anchor=NW,padx=20)
b22.pack(anchor=NW,padx=20)
l13.pack(anchor=NW, padx=20,pady=5)
b23.pack(anchor=NW,padx=20)
b24.pack(anchor=NW,padx=20)
l14.pack(anchor=NW, padx=20,pady=5)
b25.pack(anchor=NW,padx=20)
```

```
b26.pack(anchor=NW,padx=20)
l15.pack(anchor=NW, padx=20,pady=5)
b27.pack(anchor=NW,padx=20)
b28.pack(anchor=NW,padx=20)
l16.pack(anchor=NW, padx=20,pady=5)
b29.pack(anchor=NW,padx=20)
b30.pack(anchor=NW,padx=20)
but.pack()
l17.pack(anchor=NW,padx=20,pady=5)
l18.pack(anchor=SE,padx=20,pady=5)

scrollable_body.update()

root.mainloop()
```

ДОДАТОК Б

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНІ НАУКИ

ПРОГНОЗУВАННЯ РОЗВИТКУ ЦУКРОВОГО ДІАБЕТУ ЗА ДОПОМОГОЮ МЕТОДІВ МАШИННОГО НАВЧАННЯ НА МОВІ ПРОГРАМУВАННЯ PYTHON

Виконала: студентка 4 курсу, групи КНД-41
Карпік К.О.
Керівник: к.т.н,., доцент каф. ІПЗ
Жебка В.В.

м Київ 2021р.

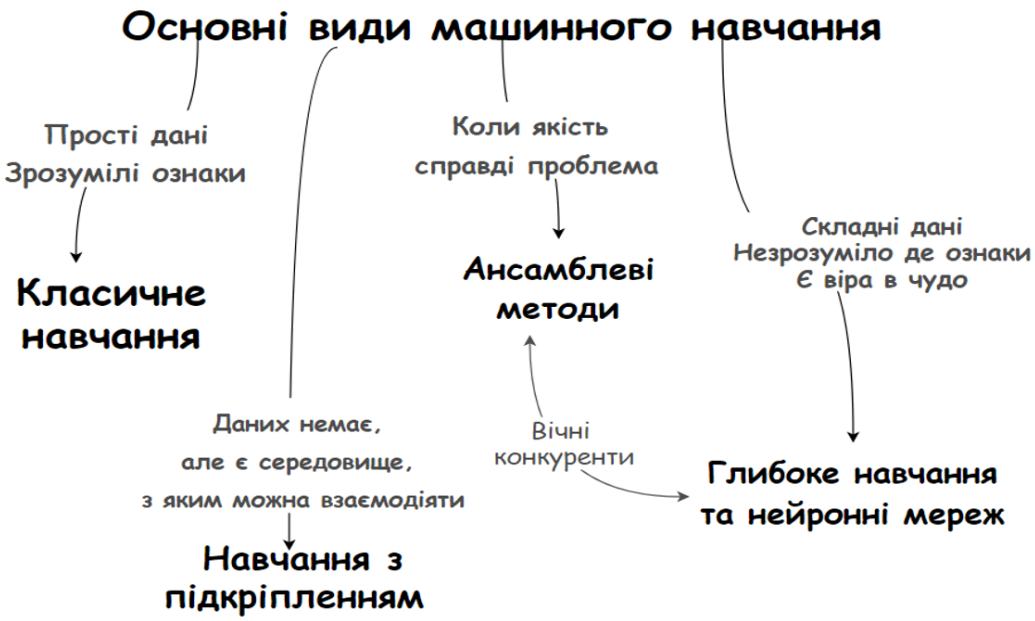
Актуальність роботи

Об'єкт дослідження – прогнозування ймовірності діагностики цукрового діабету на ранніх стадіях за допомогою ансамблевого методу машинного навчання – XGBoost реалізованого на мові програмування Phyton.

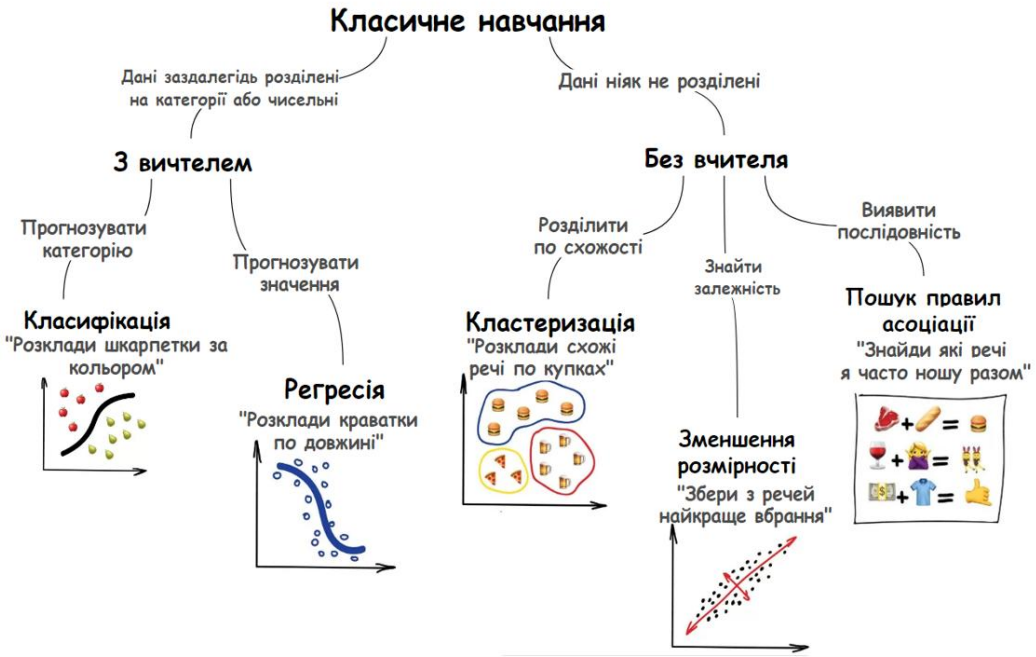
Предмет роботи – програма для прогнозування поставлення діагнозу цукровий діабет на ранніх стадіях за допомогою методу машинного навчання.

Мета роботи – створення моделі для прогнозування ймовірності діагностики цукрового діабету на ранніх стадіях для досягнення швидкої та недорогої діагностики захворювання.

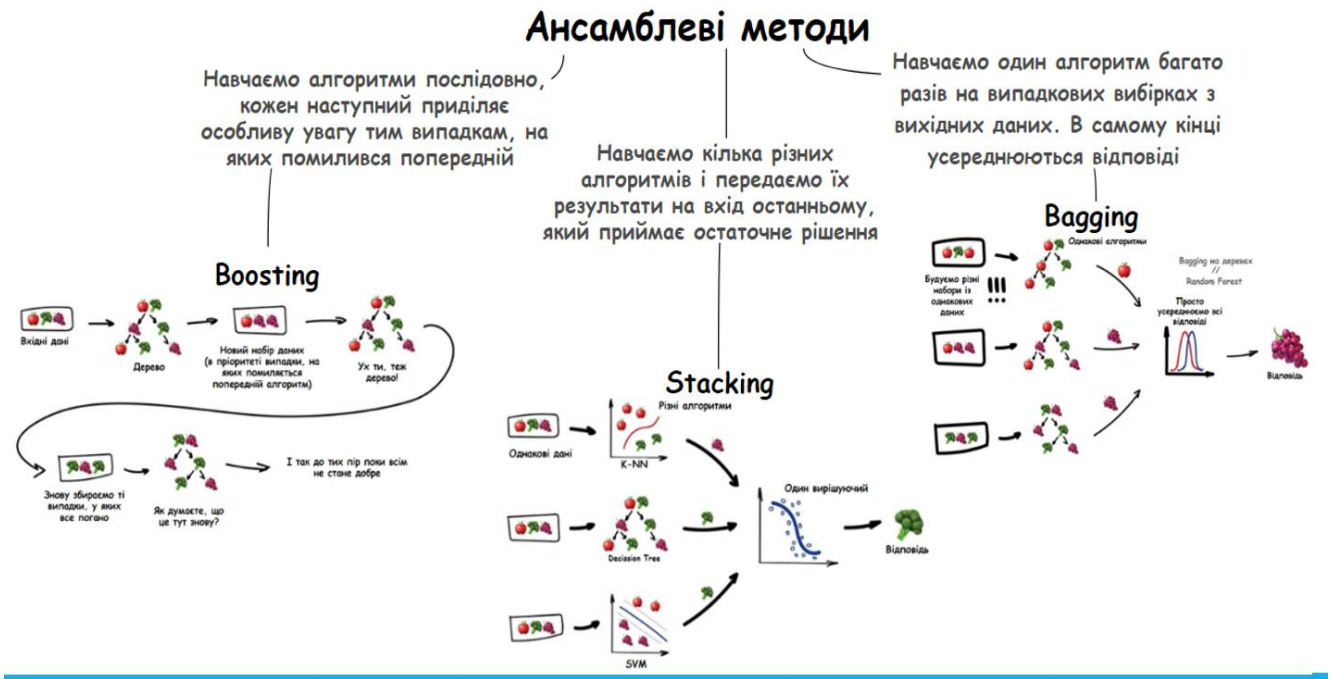
Завдання роботи – розробка програми для поставлення діагнозу цукровий діабет на ранніх стадіях за допомогою методу машинного навчання.



Класифікація машинного навчання

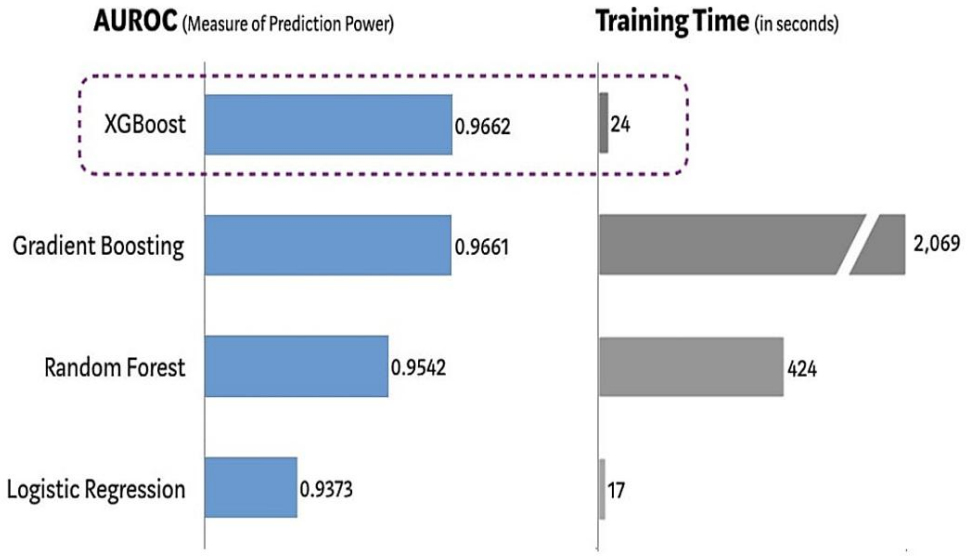


Класифікація машинного навчання

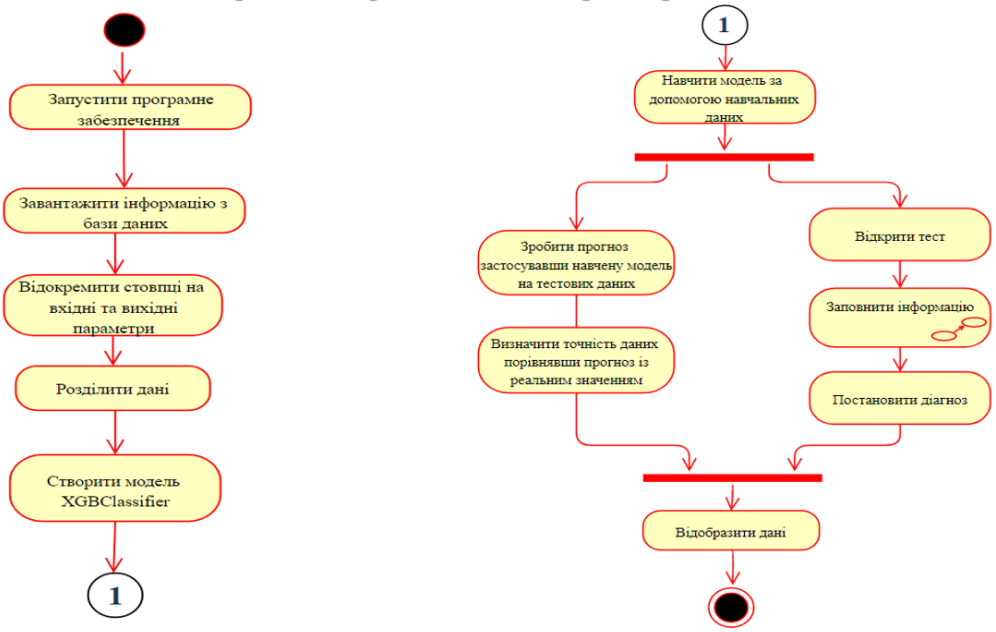


Класифікація машинного навчання

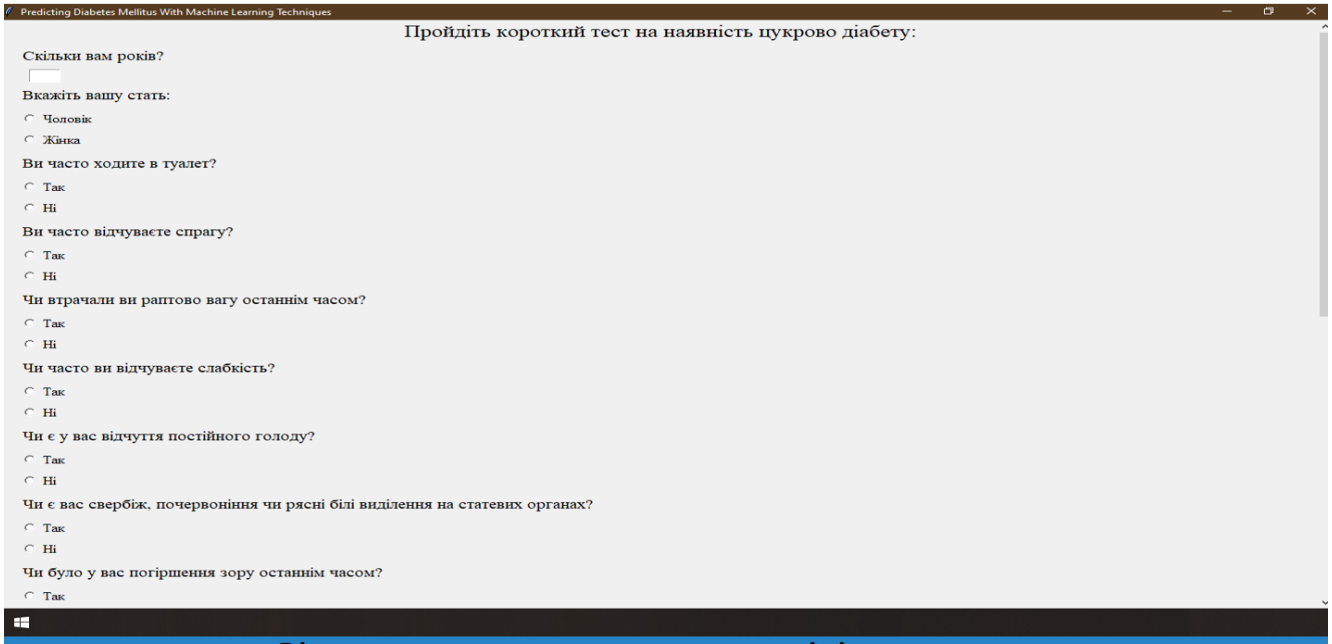
Порівняння деяких методів машинного навчання



Агоритм роботи програми

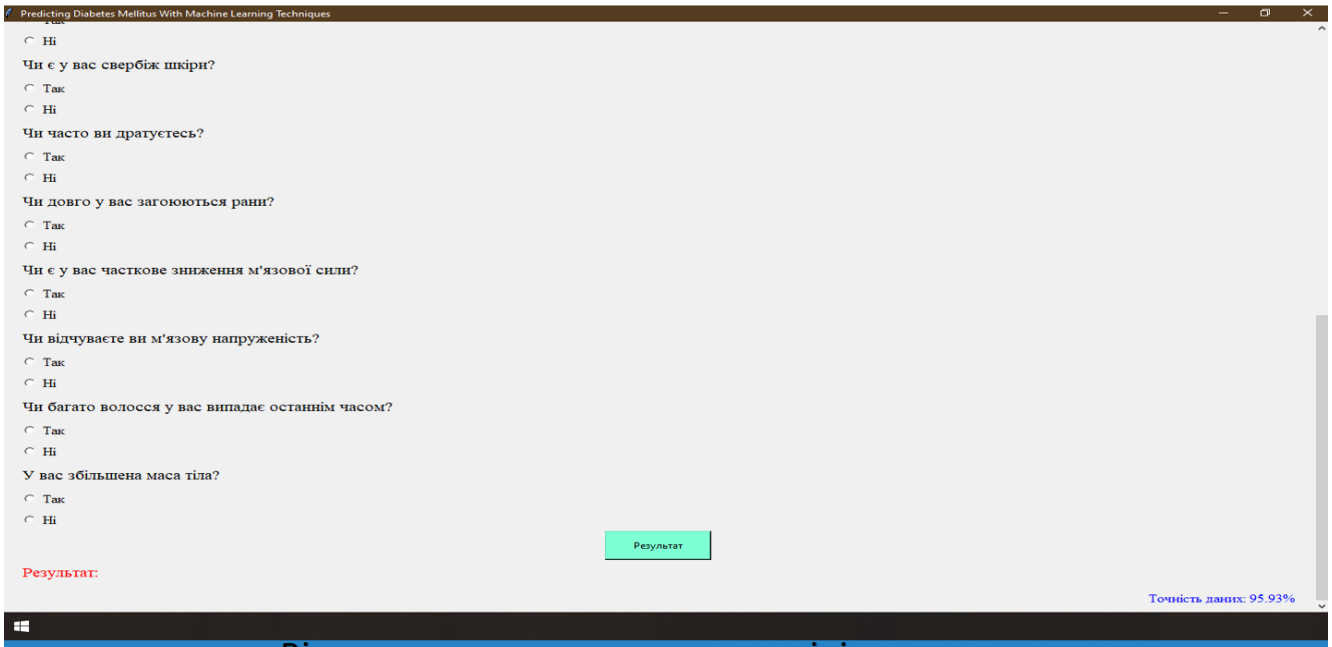


Інтерфейс розробленої програми



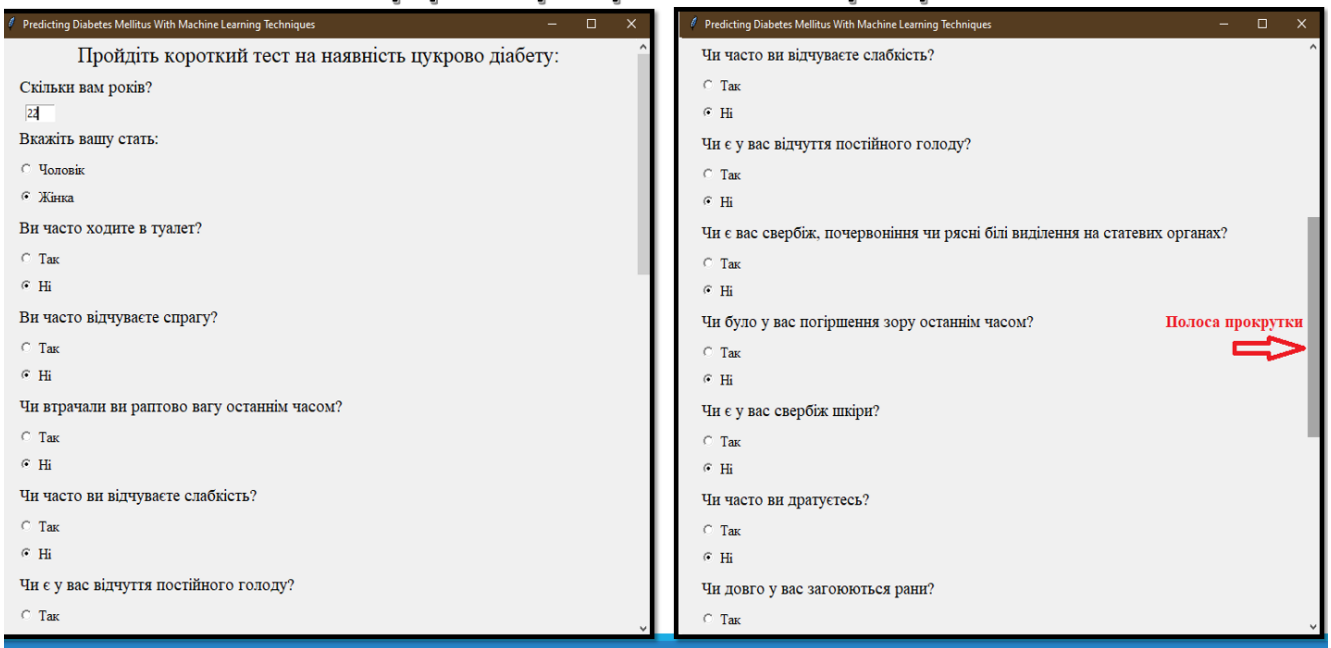
Вікно програми у повному розмірі – верхня частина

Інтерфейс розробленої програми



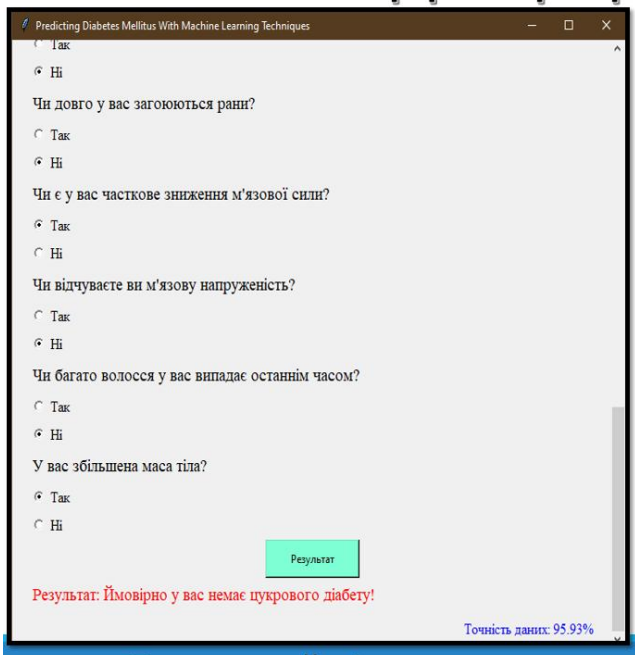
Вікно програми у повному розмірі –нижня частина

Інтерфейс розробленої програми

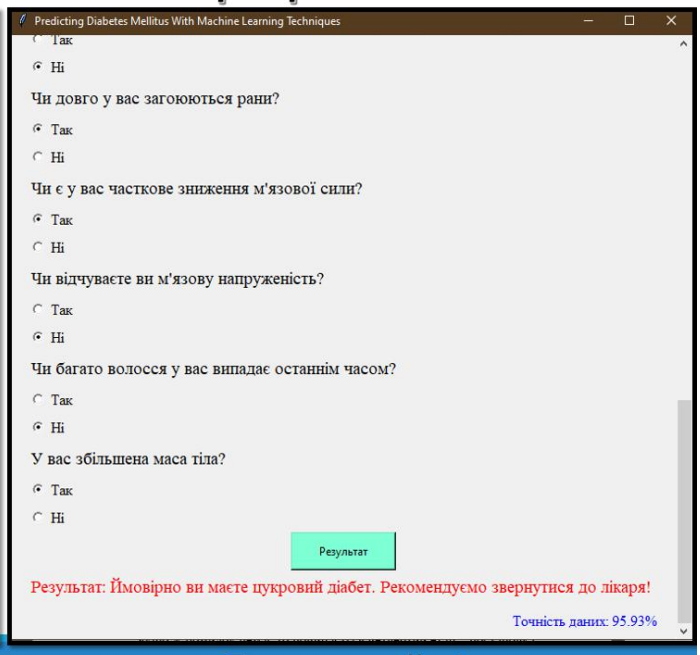


Відповіді на питання

Інтерфейс розробленої програми



Негативний діагноз



Позитивний діагноз

Висновки

- ✓ Проаналізовано популярні методи машинного навчання, створено таблицю їх переваг та недоліків.
- ✓ Визначено найкращий метод машинного навчання. Результати показали, що алгоритм XGBoost має найвищу точність порівняно з іншими вивченими методами.
- ✓ Створено алгоритм роботи програми в нотації діаграми діяльності.
- ✓ Було обрано та встановлено всі необхідні бібліотеки для подальшого створення програми.
- ✓ Дані для дослідження були отримані від 520 пацієнтів Sylhet Diabetes Hospital у Сілхеті, Бангладеш, шляхом прямих опитувальних листів пацієнтів у 2020 році та схвалено лікарем.
- ✓ За допомогою моделі XGBoost було навчено програму для прогнозування діагнозу на основі отриманої бази даних.
- ✓ Використовуючи імпортовані бібліотеки було створено зручний та простий у використанні інтерфейс користувача.

Апробація

За матеріалами були опубліковані тези:

- Карпік К.О., Жебка В.В Як поліпшити ефективність прогнозування, комбінуючи прогнози з декількох моделей. // СУЧАСНІ ДОСЯГНЕННЯ КОМПАНІЇ HEWLETT PACKARD ENTERPRISE В ГАЛУЗІ ІТ ТА НОВІ МОЖЛИВОСТІ ЇХ ВИВЧЕННЯ І ЗАСТОСУВАННЯ: Матеріали міжнародної науково-практичної конференції. Збірник тез. 16.12.2020, ДУТ, м. Київ – К.: ДУТ, 2020. – с. 8.
- Карпік К.О., Жебка В.В Прогнозування розвитку цукрового діабету за допомогою методів машинного навчання. // СУЧАСНІ ІНФОКОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ: Матеріали дванадцятої науково-технічної конференції. Збірник тез. , ДУТ, м. Київ – К.: ДУТ, 2021. – с.

Дякую за увагу!