Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету (відділення))

Кафедра інформаційних технологій
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи
перший (бакалаврський)
(рівень вищої освіти)

на тему: **Розроблення інформаційної системи для аналізу порушення координації рухів людини**

Виконав: студент V курсу, групи ІСТз-51
спеціальності
126 – "Інформаційні системи і технології"
(шифр і назва напряму підготовки, спеціальності)
Гренджа Я. Ф.
(прізвище та ініціали)

Керівник Опришко М. І.
(прізвище та ініціали)

Рецензент Салапак В. М.
(прізвище та ініціали)

Львів – 2023 р.

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ деревообробних та комп'ютерних технологій і дизайну

Кафедра інформаційних технологій

Рівень вищої освіти другий (магістерський)

Спеціальність 126 "Інформаційні системи та технології"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Крошний І. М.
"____" _____2023 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Гренджа Ярослав Федорович
(прізвище, ім'я, по батькові)

1. Тема роботи **Розроблення інформаційної системи для аналізу порушення координації рухів людини**

керівник роботи Опришко М. І.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від 21.11. 2022 року № С-523

2. Термін подання студентом роботи 17. 04. 2023 р.

3. Вихідні дані до роботи:

- вивчити предметну область, проаналізувати існуючі моделі прогнозування захворювань, а також відповідні програмні продукти;
- розглянути і використати алгоритми, які лежать в основі математичної моделі даного захворювання;
- спроектувати інформаційну медичну систему з допомогою мови програмування Python та відповідних бібліотек аналізу даних та візуалізації результатів.
- спроектувати веб-додаток для даної інформаційної системи;
- протестувати її роботу.

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

Розділ 1. Стан проблемної області

Розділ 2. Інформаційне та математичне забезпечення

Розділ 3. Програмне та технічне забезпечення

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Додаток А.
Додаток Б.

6. Дата видачі завдання 24 листопада 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Огляд літературних даних	24.11-30.12.2022	виконано
2	Розділ 1. Стан проблемної області	01.12-15.01.2023	виконано
3	Розділ 2. Інформаційне та математичне забезпечення	16.01-15.02.2023	виконано
4	Розділ 3. Програмне та технічне забезпечення	16.02-15.03.2023	виконано
5	Оформлення дипломної роботи	16.03-25.03.2023	виконано
6	Підготовка до захисту дипломної роботи, оформлення презентації	26.03-2.04. 2023	виконано

Студент _____ Гренджа Я. Ф.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Опришко М. І.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дипломна робота містить 72 сторінки пояснювальної записки, 11 рисунків, 20 джерел, 2 додатки.

В дипломній роботі розроблено та реалізовано інформаційну медичну систему для діагностики порушень координації рухів людини. Вона спроектована мовою програмування Python з використанням методики машинного навчання, залучивши бібліотеки аналізу даних та візуалізації результатів передбачення захворювання. Спроектовано веб-додаток з допомогою бібліотеки Streamlit для даної інформаційної системи. Ввівши в нього у відповідні віджети ознаки захворювання, отримують діагноз для людини.

Ця інформаційна система має характеристики, які були поставлені в технічному завданні при проектуванні. З її допомогою можна визначити, людина здорова чи має порушення координації рухів.

Ключові слова: машинне навчання, порушення координації рухів людини, діагностика захворювання, Google Colab, Python, Pandas, Streamlit.

ABSTRACT

Diploma paper contains 72 pages of explanatory note, 11 figures, 20 sources, 2 appendix.

The thesis developed and implemented an information medical system for diagnosing disorders of coordination of human movements. It is designed in the Python programming language using machine learning techniques, involving libraries of data analysis and visualization of disease prediction results. A web application was designed using the Streamlit library for this information system. By entering the symptoms of the disease into the corresponding widgets, a diagnosis for a person is obtained.

This information system has the characteristics that were set in the technical task during design. With its help, you can determine whether a person is healthy or has impaired coordination of movements.

Keywords: machine learning, impaired coordination of human movements, disease diagnosis, Google Colab, Python, Pandas, Streamlit.

ТЕХНІЧНЕ ЗАВДАННЯ

В дипломній роботі потрібно вирішити такі завдання:

- проаналізувати існуючі моделі діагностики порушень координації рухів людей, а також відповідні програмні продукти;
- розглянути і використати алгоритми, які лежать в основі математичної моделі даного захворювання;
- спроектувати інформаційну медичну систему з допомогою мови програмування Python та відповідних бібліотек аналізу даних та візуалізації результатів дослідження;
- спроектувати веб-додаток для даної інформаційної системи;
- протестувати її роботу.

ПЕРЕЛІК СКОРОЧЕНЬ І СПЕЦІАЛЬНИХ ТЕРМІНІВ

ЕЕГ – електроенцефалограма;

ЕМГ – електроміограма;

ІО – інформативні ознаки;

МТ – механічний тремор;

ЛКМ – ліва кнопка миші;

ПКМ – права кнопка миші;

ПКРЛ – порушення координації рухів людини;

ТТГ – тензотреморограма;

ТФГ – тензофорсограма;

ФС – функціональний стан;

ШІ – штучний інтелект.

ЗМІСТ

ВСТУП

Актуальність дипломної роботи

У всіх сферах своєї діяльності людина стикається з необхідністю виконання різних рухів, які є результатом узгодженої роботи багатьох систем організму. Завдання оцінки стабільності виконання людиною заданої рухової дії є актуальним для спорту, медицини, професійно-трудової діяльності та вимагає створення спеціалізованих технічних та програмних засобів для точної та швидкої діагностики рухових здібностей людини.

Під координаційною здатністю розуміється готовність людини до оптимального управління руховою дією та регулювання нею. Базові координаційні здібності мають широкий спектр застосування і включають: здатність орієнтуватися в просторі, диференціювати свої м’язові відчуття, реагувати на сигнали зовнішнього середовища, здатність зберігати статичну та динамічну рівновагу, почуття ритму. Ефективність виконання складних координаційних дій залежить від багатьох факторів, їх поєднання та взаємодії.

Сучасне апаратне, математичне та програмне забезпечення дозволяє автоматизувати діагностику практично будь-якого захворювання, для якого існує відповідний інструмент для реєстрації фізіологічних сигналів та відповідна методика проведення вимірювань. На цій підставі може бути побудована система правил, що дозволяє відокремлювати практично здорових людей від осіб, які мають ту чи іншу патологію. Діагностування захворювань і рухових порушень за допомогою машинного навчання дозволяє відстежувати динаміку відновлювальних заходів, що проводяться, а також може мінімізувати кількість невірних діагнозів.

Предмет дослідження – технології та засоби розробки моделі діагностики розладів рухів людей.

Об’єкт дослідження – розлади координації рухів людей.

Мета роботи – розробити інформаційну медичну систему для діагностики порушень координації рухів людей.

Завдання дипломної роботи:

- проаналізувати існуючі моделі діагностики порушень координації рухів людей, а також відповідні програмні продукти;
- розглянути і використати алгоритми, які лежать в основі математичної моделі даного захворювання;
- спроектувати інформаційну медичну систему з допомогою мови програмування Python та відповідних бібліотек аналізу даних та візуалізації результатів дослідження;
- спроектувати веб-додаток для даної інформаційної системи;
- протестувати її роботу.

Практична значимість

З допомогою цієї інформаційної системи можна діагностувати порушення розладів рухів в окремих людей за різними факторами ризику (ознаками захворювання). За вхідними факторами можна визначати здорових та хворих людей. Ввівши вхідні параметри (ознаки захворювання) у веб-додаток для цієї інформаційної системи, отримують діагноз для пацієнтів.

Ця інформаційна медична система може бути використана лікарями-невропатологами для вивлення на ранній стадії захворювань порушень координації рухів окремих груп людей, пришвидшить можливість діагностики цього небезпечного захворювання.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Моделі та методи діагностики порушень координації рухів людини

Розглянемо проблему автоматизації діагностики хвороби Паркінсона, що ґрунтується на математичному аналізі фізіологічних сигналів. Метою автоматизації є створення інструменту для об’єктивної, точної та ранньої діагностики хвороби порушень рухів. Для вирішення кожного з цих завдань потрібне створення та використання різноманітного апаратного та математичного забезпечення. Об’єктивна діагностика хвороби здійснена за допомогою простої програми, яка може функціонувати на будь-якому смартфоні, оснащеному стандартним набором датчиків. Точна діагностика цієї хвороби вимагає спеціалізованого вимірювального обладнання, що гарантує високу якість результатів вимірювань, спеціальним чином організованого вимірювального експерименту, що забезпечує отримання достатньо повної та достовірної інформації про процеси регуляції рухів, а також спеціального математичного забезпечення, що реалізує методи машинного навчання та методи математики. Рання діагностика хвороби виявляється можливою лише за комплексування різних видів вимірів, зокрема із залученням даних, що відповідають невідворотним проявам хвороби.

Сучасне апаратне, математичне та програмне забезпечення дозволяє автоматизувати діагностику практично будь-якого захворювання, для якого існує підходящий інструмент для реєстрації фізіологічних сигналів та відповідна методика проведення вимірювань. Їхнім результатом є часові ряди (ЧР) значень фізіологічних показників або фізіологічні сигнали. Як тільки у розпорядженні у дослідника виявляються фізіологічні сигнали прийнятної якості, з’являється можливість застосувати весь арсенал методів математичного аналізу ЧР; отримати точне кількісне уявлення у тому, що відповідає нормі, а що – патології. На цій підставі може бути

побудована система вирішальних правил, що дозволяє відокремлювати практично здорових людей від осіб, які мають ту чи іншу патологію.

За наявності у ЧР точних та достовірних ознак норми/ патології система вирішальних правил спиратиметься на безпосередній аналіз ЧР, який здійснюється за допомогою відповідного алгоритму обробки, що дозволяє надійно отримувати відповідні ознаки. У загальному випадку потрібен комплексний підхід до обробки експериментальних даних, які представлені фізіологічними сигналами та наборами потенційно інформативних ознак (ІО), які одержують за допомогою різних методів математичного аналізу ЧР. Тому доводиться використовувати методи машинного навчання, щоб вибирати з-поміж потенційно ІО справді інформативні; будувати засновані на ІО вирішальні правила.

Найбільш повний варіант автоматизації діагностики будь-якого захворювання полягає у побудові математичних моделей, що описують різні елементи та аспекти аналізованих явищ. Ці моделі дозволяють вибирати найбільш адекватні методи математичного аналізу ЧР та найбільш ефективні алгоритмічні моделі машинного навчання. В результаті з’являється можливість скласти добре обґрунтовану методику обробки експериментальних даних та побудувати засновану на ній систему підтримки прийняття лікарських рішень. Така система дозволить ефективним чином накопичувати експериментальні (діагностичні) дані і будувати вирішальні правила, а також коригувати їх з урахуванням нових даних.

Порушення координації рухів – це захворювання нервової системи, яке супроводжується поступовою деградацією рухової здатності людини і призводить з часом до повної втрати працездатності. На сьогоднішній день немає жодної єдиної теорії, що пояснює причини виникнення цієї хвороби і детально описує механізми її розвитку. У той же час існує уявлення про багатокільцеву систему побудови рухів, яка базується на фізіологічних процесах регулювання рухів людей.

Щоб автоматизувати діагностику, необхідно визначити коло тих проявів, які можна використовуватиме отримання інформації про систему побудови рухів. Прояви хвороби поділяються на моторні та немоторні. До моторних проявів традиційно відноситься тремор, а також такі симптоми, як акінезія (утрудненість рухів) та ригідність (підвищений тонус м’язів). Ці прояви хвороби виникають вже на пізній стадії захворювання. Крім того, існує цілий набір захворювань, які мають схожі прояви, але ніяк не пов’язані з нейронами. В загальному випадку степінь захворювання оцінюється за допомогою шкали по Хен і Яру.

Що ж до немоторних проявів хвороби, тут є декілька напрямів досліджень. Електроенцефалографічні дані свідчать про те, що ранніми проявами є зниження частоти коливань домінуючого діапазону електроенцефалограм, дезорганізація та нестабільність вейвлет-спектрограм. У той самий час спільний аналіз електроенцефалографічних даних і даних, що пов’язані з моторними проявами, дають додаткові ранні ознаки для пацієнтів, що перебувають у першій стадії за шкалою Хен Яра. Результати біохімічних досліджень мають суперечливий характер. Методи структурної нейровізуалізації дозволяють побачити лише окремі структурні зміни, які можуть виникати і за інших нейродегенеративних захворювань. Набагато більш інформативними виявляються методи функціональної нейровізуалізації. Однак ці методи в клінічній практиці поширені слабо – насамперед через складність та високу вартість.

Існують інші інструментальні дослідження, спрямовані на виявлення ранніх ознак хвороби. З них можна відзначити такі: викликані потенціали, магнітна стимуляція, ультразвукове обстеження, оцінка нюхової дисфункції. Окрема група досліджень пов’язана з оцінкою когнітивних функцій. Однак ці дані можуть бути дуже розмитими і невизначеними, так само як і ґрунтуються на них гіпотетичні висновки.

Таким чином, є досить багато різноманітних інструментальних підходів до діагностики цієї хвороби та виявлення ранніх ознак даного

захворювання. Однак особливий інтерес становлять ті методи, які дозволяють отримати точну та надійну інформацію про стан системи побудови рухів. Така постановка питання призводить до необхідності розглядати передусім моторні прояви захворювання. Тому доцільно створити автоматизовану систему діагностики.

1.2. Автоматизація діагностики порушень рухів людини

Автоматизація діагностики порушень рухів людини передбачає вирішення трьох важливих питань. Перше питання полягає в тому, що саме має бути результатом досліджень та розробок. Основна мета автоматизації – створення інструменту для об’єктивної, точної і, по можливості, ранньої діагностики хвороби. Насправді тут є три різні завдання. При цьому для вирішення кожного окремого завдання потрібен, свій власний інструмент.

Розробка інструменту для об’єктивної діагностики означає визначення тих ознак, які з високою надійністю свідчать про наявність патології. Це завдання може бути успішно вирішене за допомогою звичайного смартфона: датчики, вбудовані в апарат, дозволяють сформувати досить повне уявлення про поточний стан користувача смартфона, а встановлене на ньому програмне забезпечення, що реалізує методи машинного навчання, дозволяє будувати ефективні вирішальні правила.

Розробка інструменту для ранньої діагностики означає визначення таких ознак, які дозволяють виявити хворобу на стадії, коли можливо задіяти компенсаторні і відновлювальні процеси організму. Для вирішення цього завдання потрібно комплексування кількох різних видів вимірювань, що дозволяє ретельніше відсіювати аномалії та артефакти, що перешкоджають ефективному застосуванню методів математичного аналізу ЧР; будувати більш тонкі вирішальні правила, що описують глибші елементи регулювання рухів.

Друге питання, яке виникає під час автоматизації діагностики – які можна використовувати джерела експериментальних даних для побудови автоматизованої системи діагностики. Насамперед необхідно визначитися з тим, які саме прояви захворювання слід використовувати для автоматизації діагностики. З одного боку, всебічний аналіз моторних проявів має призводити до побудови об’єктивних шкал, що оцінюють стан системи побудови рухів. З іншого боку, немоторні прояви здатні підказати алгоритмам навчання спосіб більш тонкої класифікації фізіологічних сигналів. Саме тому, основний аналіз має будуватися виходячи з саме моторних проявів. При цьому дані, що відносяться до немоторних проявів, будуть оцінюватися за сформованими шкалами, обґрунтування яких повністю спирається на точний і об’єктивний аналіз рухів.

Існує кілька способів оцінки якості рухів та, відповідно, кілька видів фізіологічних сигналів, які можна обробляти методами математичного аналізу ЧР та отримувати з них інформативні ознаки. Насамперед йдеться про різні інструментальні методи реєстрації тремору.

Для реєстрації тремору зазвичай використовується електроміографія. Вона полягає у вимірюванні електричного потенціалу групи м’язів або окремого м’яза. Розрізняють реєстрацію електроміограми (ЕМГ) за допомогою голчастих електродів та за допомогою нашкірних електродів. Електроміографічний спосіб дозволяє проводити диференціальну діагностику різних видів тремору. У той же час, ЕМГ сигнали вимагають попереднього очищення від рухових артефактів та інших дефектів. Крім того, необхідно враховувати, що амплітуда ЕМГ-сигналів у типових випадках невелика і становить десятки мікровольт. Тому ключового значення набувають питання боротьби з електромагнітними перешкодами під час реєстрації ЕМГ.

Інший спосіб реєстрації тремору надає акселерометрія – реєстрація рухів окремих ланок моторного апарату за допомогою датчиків прискорення (акселерометрів). Однак цей спосіб реєстрації має ряд

недоліків. По-перше, аналіз треморограм, отриманих з допомогою акселерометрів, вимагає обліку антропометричних параметрів обстежуваних осіб, що істотно утрудняє створення однакових умов щодо вимірювань і знижує можливість зіставлення результатів, отриманих для різних людей. По-друге, треморограми, отримані за допомогою різних вимірювальних датчиків, будуть мати індивідуальні властивості, що визначаються технічними параметрами самих датчиків. Це природно ускладнює або навіть унеможливлює порівняльний аналіз результатів, отриманих за допомогою різного вимірювального обладнання.

Існують приклади складніших апаратно-програмних комплексів, що використовуються для дослідження тремору. Сюди відносяться: спільне використання акселерометрів та нашкірних датчиків для реєстрації ЕМГ, акселерометрів та гіроскопів; одночасне проведення ЕМГ та функціональної магнітно-резонансної томографії. У той же час для оцінки якості рухів, що здійснюються, використовуються методи, які базуються на аналізі відеозображення. Крім того, на окрему увагу заслуговує стабілометрія, спрямована на вивчення підтримки людиною пози. Тут безпосередньо реєструється положення проекції центру ваги тіла на горизонтальну площину. При цьому використовується візуальний канал зворотний зв’язок.

Кожен спосіб реєстрації фізіологічних сигналів має свої переваги та недоліки. У той же час потрібно використовувати такий спосіб реєстрації фізіологічних сигналів, який гарантує одноманітне отримання експериментальних даних високої якості та забезпечує можливість створення простих в експлуатації апаратно-програмних комплексів, призначених для широкого застосування клінічної практики. Для цього було запропоновано використовувати тензометричний спосіб реєстрації тремору для об’єктивної оцінки стану системи побудови рухів. Цей спосіб полягає у безпосередній реєстрації зусилля, що надається двома руками людини на чутливу платформу. Для кожної руки використовується

окремий датчик тензометра, який у свою чергу пов’язаний з певною міткою на екрані комп’ютера. Піддослідний, сидячи за комп’ютерним столом, утримує певний, заданий лікарем, рівень зусилля і стежить за процесом його утримання на екрані комп’ютера. При цьому випробуваний динамічно коригує рівень утримуваного ним зусилля, якщо фактичний рівень відхиляється від заданого. Це означає, що в процесі реєстрації використовується біологічний зворотний зв’язок по візуальному каналу, який замикає ще одне кільце багатокільцевої системи керування рухами. Ці коливання є справжнім тремором. Відповідні тремору сигнали називаються тензотреморограмами (ТТГ). У той же час сигнали, що відповідають зусиллю і є, по своїй суті, вихідними або сирими, слід називати тензофорсограмами (ТФГ).

Методика вимірювального експерименту передбачає послідовне виконання чотирьох різних функціональних тестів. Спочатку використовується реєстрація тремору із опорою на зап’ястя. Тут система управління рухами виявляється задіяною найменшим чином. Потім використовується реєстрація тремору для витягнутих рук, які тиснуть усією своєю вагою на чутливу платформу. Тут система управління рухами виявляється задіяної найбільшою мірою. Кожен спосіб реєстрації використовується двічі: при мінімально можливому рівні утримуваного досліджуваним зусилля і при максимально можливому рівні зусилля, що утримується випробовуваним. Реєстрація тремору в ізометричному режимі унеможливлює будь-які переміщення окремих ланок моторного апарату, що гарантує дотримання умов проведення вимірювань. Тензометричний спосіб реєстрації простий у реалізації, допускає тиражування як апаратно-програмних комплексів, які гарантують отримання даних високої якості, і може стати основою для побудови систем масової скринінгової діагностики хвороби. Приклад реалізації тензометричного способу реєстрації тремору є розроблений програмно-апаратний комплекс ПАК-ЦНС-01. Він забезпечує моніторинг функціонального стану

центральної нервової системи; виявлення та діагностику патологічних станів на ранній стадії рухових порушень, що ще не виявляються зовні; контроль дієвості лікарської терапії або інших методів лікування хворих з порушеннями рухів внаслідок уражень центральної нервової системи, яку можна використовувати як полігон для функціональних тестів, які базуються на завданні різних рівнів утримуваного випробуваним зусилля.

1.3. Тензометричний спосіб реєстрації зусилля та тремору

Реєстрація тремору, що здійснюється за допомогою тензометричного способу реєстрації, замикає ще одне кільце системи побудови рухів та вводить її у певний функціональний стан (ФС). Таким чином, об’єктом дослідження є система побудови рухів і ті ФС, в яких виявляється система побудови рухів при проведенні вимірювальних експериментів як практично для здорових, так і для людей, у яких є та чи інша патологія. Предметом дослідження виявляються ТТГ і потенційно інформативні ознаки, які можна одержати в результаті обробки ТТГ з допомогою різних методів математичного аналізу ЧР.

Мета обробки експериментальних даних – побудова системи розпізнавання. Для її побудови необхідно вибрати об’єкти, які фігурують у системі розпізнавання. По-перше, як об’єкти можуть виступати набори експериментальних даних, які пов’язані з кожним випробуваним. У цьому випадку система розпізнавання прийматиме рішення щодо кожного випробуваного на підставі всієї наявної інформації. По-друге, як об’єкти можуть виступати блоки експериментальних даних, які отримані в результаті окремих вимірювальних експериментів, проведених у певний момент часу. У цьому випадку може виникнути, наприклад, ситуація, коли який-небудь випробуваний, який до певного моменту вважався практично здоровим, став розпізнаватись як людина хвора. По-третє, як об’єкти можуть виступати блоки експериментальних даних, отриманих в результаті проведення окремих функціональних тестів, що становлять

вимірювальний експеримент за участю конкретного випробуваного. І тут система розпізнавання прийматиме рішення виходячи з математичного аналізу вже окремих ЧР, кожен з яких належить до певного функціональному тесту. У цьому кожному рівні ієрархії може бути побудована своя система розпізнавання. Якщо ж розглядати всі системи розпізнавання загалом, можна говорити про багаторівневу систему розпізнавання.

Вибір класифікації визначає структуру системи розпізнавання. Якщо розглядається класифікація об'єктів якихось двох класів, то йдеться про однокрокову процедуру розпізнавання з відносно простим критерієм якості розпізнавання. Відповідно, можна розглянути всі можливі поєднання двох класів та побудувати для кожного допустимого поєднання свою систему вирішальних правил. При цьому важливо, як формулюється критерій якості розпізнавання. Розгляд різних видів патології або різних стадій розвитку одного і того ж захворювання також може вимагати різного обліку помилок першого та другого роду. У цьому випадку кожен вид помилок буде входити до загального критерію зі своїм ваговим коефіцієнтом. Конкретні значення вагових коефіцієнтів визначатимуться внаслідок проведення процедури послідовного розпізнавання.

Як тільки кількість класів, що беруть участь у розпізнаванні, буде більше двох, з'являються різні варіанти. По-перше, більша кількість класів призводить до ускладнення вирішальних правил та критеріїв якості розпізнавання. По-друге, можливе введення додаткових штучних класів, що відповідає відмові від розпізнавання, або додаткового класу, який містить ті об'єкти заданої вибірки, щодо належності яких є якісь сумніви. По-третє, існує можливість організації послідовної процедури розпізнавання з вибором на кожному етапі свого набору класів, і критеріїв, що використовуються длч оцінки якості розпізнавання.

Класифікація піддослідних, яка базується на прямому співставленні класу піддослідних і ФС, у яких перебувають, породжує серйозну

проблему. Оскільки реєстрація фізіологічних сигналів відбувається у певний час, то, власне, можна оцінити ФС випробуваного лише у момент реєстрації фізіологічних сигналів. Крім того, проведення певного функціонального тесту вводить досліджуваного в деякий ФС, який лише опосередковано пов’язаний з класом досліджуваного. Це означає, що класи випробуваних є деякою оцінкою узагальнених функціональних макростанів. У цьому самі ФС є проявами певного функціонального макростану за умов конкретного функціонального тесту. Спочатку здорова людина може з часом захворіти, і, відповідно, має змінитися оцінка її загального макростану. Одному й тому ж функціональному макростану може відповідати цілий набір допустимих ФС, кожний може бути своєю чергою пов’язаний з тим чи іншим функціональним тестом. Таким чином, можна констатувати, що потрібно реконструювати структуру ФС, що відповідає заданому набору функціональних макростанів та структурі функціональних тестів. Традиційні методи машинного навчання мають справу лише з класами досліджуваних та з функціональними тестами, проте така постановка завдання є не зовсім коректною. Введення проміжного прихованого шару об’єктів істотно уточнює постановку завдання і уможливлює коректну інтерпретацію передбачуваних результатів.

Розширена постановка завдання має призвести до нової класифікації піддослідних, яка базується на їх ранжуванні відповідно до того, у яких ФС ці піддослідні реально виявляються в результаті виконання тих чи інших функціональних тестів. Це підказує можливість постановки та зворотного завдання: вибрати структуру функціональних тестів, яка дозволить оптимальним чином визначити структуру функціональних макростанів. Класифікація також повинна відповісти і на питання про групи ризику: якщо у початково здорової людини є певні особливості переходу з одних ФС в інші ФС, то ці особливості збережуться або стануть більш вираженими у випадку, якщо вона захворіє. В ідеальній ситуації

випробуваний, який потрапив за результатами виконання стандартних тестів у групу ризику, повинен проходити додаткові функціональні тести, що дозволить глибше дослідити структуру функціональних макростанів.

Таким чином, класифікація дозволяє побудувати абсолютну шкалу та використовувати її для оцінки ФС піддослідних. Відповідно до цього ранжування піддослідних може описуватися нечіткою класифікацією. Якщо розглядати структурні елементи аналізованих ЧР як об'єкти, які також можуть брати участь у побудові системи розпізнавання, виникає ситуація, аналогічна тій, яка має місце при оцінці ФС. Справді, самі структурні елементи заздалегідь невідомі, і їх необхідно виділити за допомогою того чи іншого методу обробки ЧР. Тут, як і в оцінці ФС, йдеться про реконструкцію структури аналізованих ЧР виходячи з заданої структури функціональних тестів. Для реалізації цього підходу потрібно, щоб структурні елементи можна було б інтерпретувати як події чи стану певної динамічної системи. Основна гіпотеза полягає в тому, що кожному функціональному макростану та кожному функціональному тесту відповідають певні набори станів та структура переходів між ними. Критерієм ефективності того чи іншого методу структурного аналізу ЧР буде можливість успішного зіставлення ФС, функціональних макростанів та структурних елементів ЧР, що аналізуються. Такий підхід дозволяє вибирати порогові значення у застосовуваних на практиці алгоритмах і формулювати зовнішні стосовно алгоритмів обробки ЧР критерії оптимальності, безпосередньо спираючись на дані про структуру функціональних макростанів при виконанні різних функціональних тестів.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Бібліотека Pandas для обробки даних

Розглянемо бібліотеки, які використовуються в машинному навчанні та аналізі даних. Бібліотека Pandas активно використовується для роботи з даними, які можуть бути представлені в виді таблиць. Вона надає дуже широкий функціонал для роботи з даними та їх обробки.

Для установки цієї бібліотеки використовується команда:

```
pip install pandas
```

Після цього для початку роботи потрібно імпортувати бібліотеки Numpy та Pandas:

```
import numpy as np
import pandas as pd
```

Тип даних pd.Series є одновимірним набором даних. Відсутні дані записуються як np.nan. Створимо Series зі списку:

```
some_list = [1, 3, 5, 6, 8]
ser_1 = pd.Series(some_list)
ser_1
```

Також можна в явному вигляді вказати індекси, щоб потім було зручніше звертатися до елементів:

```
ind = ['1st day', '2nd day', '3rd day', '4th day', '5rd day',
'6th day']
ser_2 = pd.Series(some_list, index=ind)
ser_2
ser_2['4th day']
```

Також можна дати pd.Series ім’я:

```
ser_3 = pd.Series(some_list, index=ind, name='Temperature')
ser_3
```

З індексами можна працювати так само, як і у випадку із звичайним list:

```
print(ser_3[0])
```

```
print('-----------')
print(ser_3[1:3])
print('-----------')
print(ser_3[::-1])
```

Розглянемо індексування pd.Series за умовами:

```
date_range = pd.date_range('20190101', periods=10)
ser_4 = pd.Series(np.random.rand(10), index=date_range)
ser_4
ser_4 > 0.5
```

Як індекс можна вказати вираз, будуть повернуті тільки ті елементи, для яких значення є True:

```
ser_4[ser_4 > 0.5]
ser_4[(ser_4 > 0.6) | (ser_4 < 0.2)]
ser_4[(ser_4 > 0.6) & (ser_4 < 0.2)]
```

Тип pd.Series можна відсортувати як за значеннями, так і за індексом:

```
ser_4.sort_index()
ser_4 = ser_4.sort_values()
ser_4
```

Розглянемо операції з series:

```
ser_4 + 100
np.exp(ser_4)
term_1 = pd.Series(np.random.randint(0, 10, 5))
term_2 = pd.Series(np.random.randint(0, 10, 6))
term_1 + term_2
term_1.shape
```

Тип даних pd.DataFrame є двомірною таблицею з даними. Має індекс та набір стовпців. Таблицю можна побудувати, наприклад, зі словника, значеннями якого є одномірні набори даних.

Dataframe можна створити із словника. Ключ буде відповідати стовпцю:

```
some_dict = {'one': pd.Series([1,2,3], index=['a','b','c']),
             'two': pd.Series([1,2,3,4],
index=['a','b','c','d']),
```

```
            'three': pd.Series([5,6,7,8],
index=['a','b','c','d'])}
df = pd.DataFrame(some_dict)
df
some_array = [[1,1,5], [2,2,6], [3,3,7], [np.nan, 4,8]]
df = pd.DataFrame(some_array, index=['a', 'b', 'c', 'd'],
columns=['one', 'two', 'three'])
df
df.values
df.columns
df.columns = ['first_column', 'second_column', 'third_column']
df.index = [1,2,3,4]
df
```

Є багато способів індексувати DataFrame в Pandas. Індексування по стовпцю повертає pd.Series. Можна вибирати не один стовпець, а відразу кілька. Тоді знову повернеться pd.DataFrame:

```
first_column = df['first_column']
first_column
df.first_column
subset_dataframe = df[['first_column', 'second_column']]
subset_dataframe
one_column_dataframe = df[['first_column']]
one_column_dataframe
```

По рядках. Можна писати будь-які слайси, як у Python-списку. Вони будуть застосовуватися до рядків:

```
df[:1]
df[1:4]
```

Універсальне індексування: .loc та .iloc. .loc і .iloc - це два взаємозамінні атрибути, які дозволяють індексувати по обох осях відразу. По індексах:

```
df.iloc[1:3, :2]
df.loc[1:3, ['first_column', 'second_column']]
```

Модифікації датасету; створення нових колонок. Можна просто брати і створювати нову колонку:

```
new_column = [5,2,1,4]
df['new_column'] = new_column
df
```

Аналогічно, можна застосовувати до окремих колонок арифметичні операції:

```
df['first_column'] = df['first_column'] * 10
df
```

Розглянемо роботу з датасетом titanic_data.csv. Він містить різноманітну інформацію про пасажирів Титаніка (квиток, клас, вік). titanic_surv.csv містить для кожного пасажира з першого файлу інформацію про те, чи вижив цей пасажир (мітка 1) чи ні (мітка 0).

Як правило, дані зберігаються у вигляді таблиць у файлах формату .csv або .xlsx. Завантажимо перший файл:

```
df_1 = pd.read_csv('titanic_data.csv')
pass_link =
'https://www.dropbox.com/s/lyzcuxu1pdrw5qb/titanic_data.csv?dl
=1'
titanic_passengers = pd.read_csv(pass_link,
index_col='PassengerId')
print('Всього пасажирів: ', len(titanic_passengers))
titanic_passengers.head(10)
```

Можна дізнатися про розмір таблиці, інформацію про значення таблиці, різні статистики за значеннями:

```
titanic_passengers.shape
titanic_passengers.info()
titanic_passengers.describe()
```

Опишемо цей датасет: який розподіл жінок/чоловіків у ньому? Скільки пасажирів їхало у кожному класі? Який середній/ мінімальний/ максимальний вік пасажирів?

```
(titanic_passengers['Age'].min(),
titanic_passengers['Age'].mean(),
titanic_passengers['Age'].max())
titanic_passengers['Sex'].value_counts()
titanic_passengers['Pclass'].value_counts()
```

Згрупувати записи за класами пасажирів, у кожній групі порахувати середній вік. Для цього можна використати метод pandas.DataFrame.groupby.

```
titanic_passengers.groupby(['Pclass']).mean()
titanic_passengers.groupby(['Pclass'])['Age'].mean()
```

Таблиці можна об’єднювати кількома способами. Розглянемо злиття за індексом: цей метод називається pd.join:

```
# df_2 = pd.read_csv('titanic_surv.csv')
surv_link =
'https://www.dropbox.com/s/v35x9i6a1tc7emm/titanic_surv.csv?dl
=1'
df_2 = pd.read_csv(surv_link)
df_2.head()
```

Об’єднати два датасета по стовпчику індекса:

```
df_2.index = np.arange(1, 892)
df_2 = df_2.sample(frac=1)
df_2.head()
titanic_passengers = titanic_passengers.join(df_2)
titanic_passengers.head()
```

Скільки всього пасажирів, що вижили? Скільки пасажирів, що вижили, по статях? Побудувати матрицю кореляцій факту виживання, статі та віку.

```
titanic_passengers['Survived'].sum()
titanic_passengers.groupby(['Sex'])['Survived'].sum()
corr_data = titanic_passengers[['Sex', 'Age', 'Survived']]
corr_data['Sex'] = (corr_data['Sex'] == 'female').astype(int)
corr_data.head()
corr_data.corr()
import seaborn as sns
```

```
sns.heatmap(corr_data.corr(), annot=True, cmap='coolwarm',
            vmin=-1, vmax=1, annot_kws={"size": 16})
```

2.2. Оцінка якості моделей з допомогою метрик машинного навчання

У завданнях машинного навчання з метою оцінки якості моделей і порівняння різних алгоритмів використовуються метрики. Розглянемо деякі критерії якості в задачах класифікації, що є важливим при виборі метрики.

Accuracy, precision та recall

Перед переходом до самих метрик потрібно розглянути confusion matrix (матриця помилок). Припустимо, що у нас є два класи та алгоритм, що передбачає передбачення кожного об’єкта одному з класів, тоді матриця помилок класифікації буде виглядати наступним чином:

	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

Помилки класифікації бувають двох видів: False Negative (FN) та False Positive (FP). Очевидною метрикою є accuracy – частка правильних відповідей алгоритму:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

Нехай потрібно оцінити роботу спам-фільтра пошти. Є 100 неспам листів, 90 з яких класифікатор визначив правильно (True Negative = 90, False Positive = 10), і 10 спам-листів, 5 з яких класифікатор також визначив правильно (True Positive = 5, False Negative = 5). Тоді accuracy:

$$accuracy = \frac{5 + 90}{5 + 90 + 10 + 5} = 86,4$$

Однак якщо будуть передбачати всі листи як не-спам, то отримають вищий accuracy:

$$accuracy = \frac{0 + 100}{0 + 100 + 0 + 10} = 90,9$$

Precision, recall, F-міра

Для оцінки якості роботи алгоритму кожному з класів окремо вводять метрики precision (точність) і recall (повнота):

$$precision = \frac{TP}{TP + FP} \tag{2.2}$$

$$recall = \frac{TP}{TP + FN} \tag{2.3}$$

Precision можна інтерпретувати як частку об’єктів, названих класифікатором позитивними і при цьому дійсно позитивними, а recall показує, яку частку об’єктів позитивного класу з усіх об’єктів позитивного класу знайшов алгоритм.

Саме введення precision не дозволяє записувати всі об’єкти в один клас, тому що в цьому випадку отримають зростання рівня False Positive. Recall демонструє здатність алгоритму виявляти цей клас взагалі, а precision – здатність відрізняти цей клас від інших класів. Помилки класифікації бувають двох видів: False Positive та False Negative. У статистиці перший вид помилок називають помилкою першого роду, а другий – помилкою другого роду.

Precision і recall не залежать, на відміну accuracy, від співвідношення класів і тому застосовні за умов незбалансованих вибірок. Часто у реальній практиці стоїть завдання знайти оптимальний баланс між цими двома метриками.

Зазвичай при оптимізації параметрів алгоритму використовується одна метрика, покращення якої очікують побачити на тестовій вибірці. Існує кілька різних способів об’єднати precision і recall в агрегований критерій якості. F-міра (загалом) – середнє гармонійне precision і recall:

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \quad (2.4)$$

AUC-ROC та AUC-PR

При конвертації дійсної відповіді алгоритму (ймовірності приналежності до класу) в бінарну мітку, потрібно вибрати будь-який поріг, при якому 0 стає 1. Природним і близьким здається поріг, що дорівнює 0.5, але він не завжди виявляється оптимальним.

Одним із способів оцінити модель загалом, не прив'язуючись до конкретного порогу, є AUC-ROC (або ROC AUC) – площа (Area Under Curve) під кривою помилок (Receiver Operating Characteristic curve). Дана крива являє собою лінію від (0,0) до (1,1) в координатах True Positive Rate (TPR) і False Positive Rate (FPR):

$$TPR = \frac{TP}{TP + FN} \quad (2.5)$$

$$FPR = \frac{FP}{FP + TN} \quad (2.6)$$

FPR показує, яку частку з об'єктів negative класу алгоритм передбачив неправильно. В ідеальному випадку, коли класифікатор не робить помилок (FPR = 0, TPR = 1), отримують площу під кривою, рівну одиниці; в іншому випадку, коли класифікатор випадково видає ймовірності класів, AUC-ROC буде прямувати до 0.5, оскільки класифікатор видаватиме однакову кількість TP та FP.

Кожна точка на графіку відповідає вибору деякого порогу. Площа під кривою показує якість алгоритму (більше – краще), крім цього, важливою є крутизна самої кривої – потрібно максимізувати TPR, мінімізуючи FPR, отже, крива в ідеалі має прямувати до точки (0,1).

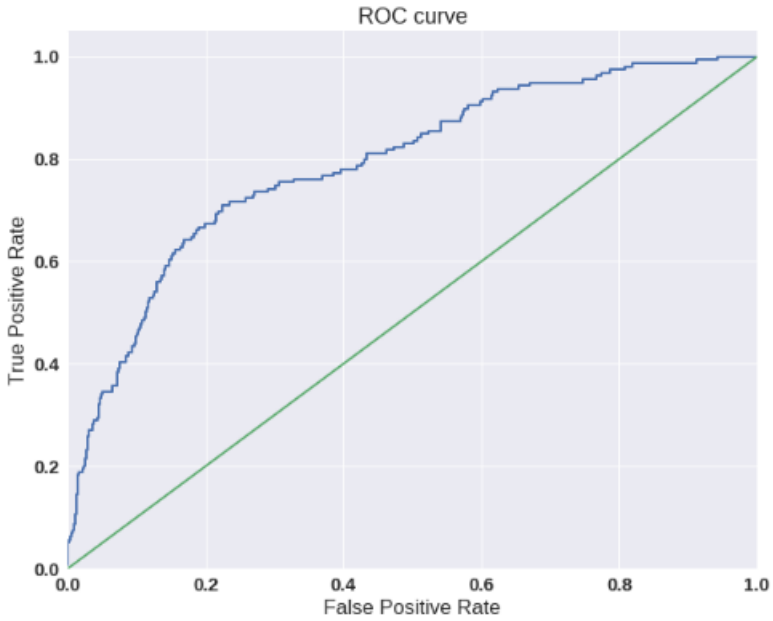


Рис. 2.1.

Критерій AUC-ROC стійкий до незбалансованих класів і може бути інтерпретований як ймовірність того, що випадково обраний positive об’єкт буде проранжований класифікатором вище (матиме більш високу ймовірність бути positive), ніж випадково обраний negative об’єкт.

2.3. Аналіз ЕЕГ з допомогою вейвлет перетворення

Відомо, що характерною рисою порушення координації рухів є синдром дезінтеграції, що проявляється на різних системних рівнях (рухові порушення, вегетативна, емоційні та психічні порушення). Виявлені методом вейвлет аналізу зміни частотно-часової структури засвідчили, що дезінтеграція може виявлятися у динаміці електричної активності мозку. Були аналізовані дані переважно у пацієнтів на 2-3 стадії захворювання за шкалою Хен-Яра.

Частотно-часова спектрограма безперервного вейвлету перетворення задається формулою:

$$S_x(\tau,\mathrm{f}) = |W(\tau,f)|^2, \qquad (2.7)$$

$$W(\tau,T)=\frac{1}{\sqrt{T}}\int x(t)\psi^{*}\left(\frac{t-\tau}{T}\right)dt \quad , \tag{2.8}$$

$$W(\tau,T)=\frac{1}{\sqrt{T}}\int x(t)\psi^{*}\left(\frac{t-\tau}{T}\right)dt \quad , \tag{2.9}$$

$$\psi(\eta)=\frac{1}{\sqrt{\pi F_b}}e^{2i\pi F_c\eta}e^{-\frac{\eta^2}{F_b}} \quad , \tag{2.10}$$

де $S_x(\tau,f)$ – спектральна густина потужності, f = 1/T, F_b, F_c – параметри, як правило приймають $F_b = F_c = 1$.

Вейвлет спектрограма ЕЕГ складається із серій піків, що відображають зміни амплітуди спектральних коефіцієнтів у різних частотних діапазонах. ЕЕГ складається з коливань різної частоти і тривалості. На рис. 2.1 представлені вейвлет перетворення ЕЕГ здорової людини та хворої людини за шкалою Хен-Яра (праворуч).

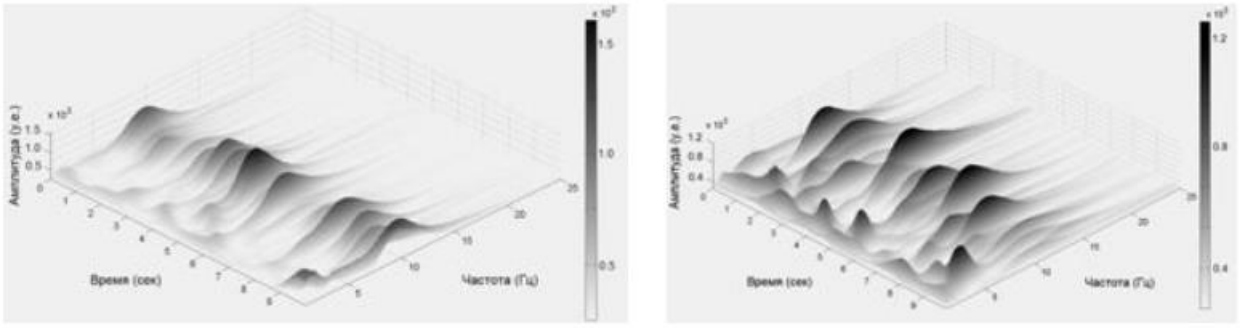


Рис. 2.2. Вейвлет спектрограми ЕЕГ. Зліва – контроль, справа – пацієнт на першій стадії захворювання.

З рис. 2.2 видно, що як і у здорових пацієнтів, так і у хворих, вейвлет спектрограми складаються із серії пологих піків (приблизно один – два рази на секунду) спектральної густини потужності на частотно-часовій площині. У здорових пацієнтів ці піки виникають приблизно на одній частоті і утворюють регулярні хребти, які за Фур'є аналізом дають

загальноприйняті ритми – дельта, тета, альфа, бета тощо. У пацієнтів положення та розкид частот піків істотно сильніше змінюється у часі та їх спектральна потужність перерозподіляється між частотними діапазонами. Зокрема, суттєво зростають піки у низькочастотному діапазоні (4-6 Гц).

Домінуючий ритм ЕЕГ має амплітуду, що перевищує амплітуду інших частотних діапазонів. Множина частот окремих локальних максимумів відбиває діапазон домінуючого ритму ЕЕГ. У нормі всі вершини піків становлять чітко виражений хребет, що є альфа ритмом, що говорить про достатню стабільність частоти домінуючого ритму ЕЕГ у здорової людини. У пацієнтів на першій стадії хвороби відбувається значна дезорганізація цієї тривимірної картини: хребет вейвлет перетворення складається з піків, що мають різну порівняно з нормою частоту.

Виділяючи екстремуми піків вейвлет спектрограм та аналізуючи статистику розподілу частотно-часових координат екстремумів та їх потужності, можна виявити ознаки хвороби на ранній стадії. Крім того, ця статистика може бути різною у пацієнтів із різними стадіями захворювання. Як статистику використовуються гістограми розподілу за частотою кількості екстремумів та їх сумарної спектральної щільності потужності в деякому вузькому діапазоні частот.

Суть обробки та аналізу вейвлет спектрограм сигналів ЕЕГ полягає в тому, що визначаються амплітуди $A_i(F_i, t_i)$ піків спектрограм. Далі площина час-частота (0-T, F_{min}-F_{max}) розбивається на вікна з розмірами (T, F). Розміри вікна доцільно вибирати за часом ΔT=(0,05-1,00) T сек, а частотою – ΔF=(0,02-0,03) F_{max} Гц. Потім у кожному вікні обчислюються суми амплітуд піків спектрограм A_i і будуються гістограми розподілу сум A_i від частоти.

На рис. 2.3 наведено приклади розподілів сум амплітуд екстремумів у частотно-часових вікнах. Вгорі для пацєнта з контрольної групи, внизу – для пацієнта на 1 стадії захворювання. Ці розподіли показують асиметрію електричної активності мозку в початковій стадії хвороби порівняно з

контролем, що полягає в дезорганізації домінуючого ритму, а саме, у збільшенні частотного розкиду його піків у хворій півкулі.

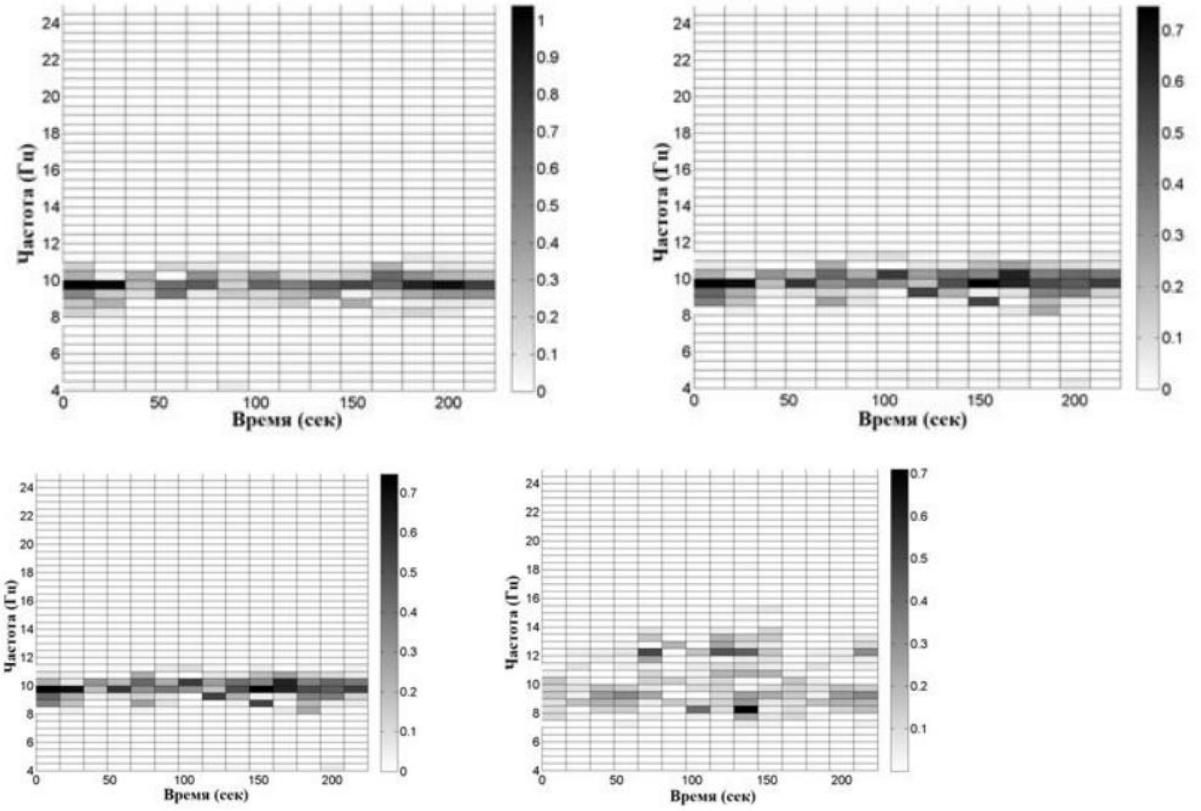


Рис. 2.3. Розподіл сум амплітуд екстремумів вейвлет спектрограм у частотно-часових вікнах.

Сигнали ЕЕГ за своєю природою є нестаціонарними, тому за доцільне введення кількісної оцінки цієї нестаціонарності і зіставлення її в нормі з ранніми стадіями хвороби. Суть запропонованої оцінки полягає в оцінці парних кореляцій частотних розподілів сум амплітуд екстремумів по часових вікнах.

Для прикладів, поданих на рис. 2.3, число таких вікон становить 14. Відповідно виходить симетрична матриця коефіцієнтів кореляції з одиничною діагоналлю. У нормі кореляційні матриці містять значну кількість великих коефіцієнтів кореляції, і навпаки, у пацієнта з захворюванням кореляційні матриці містять значну кількість малих коефіцієнтів кореляції. Тому доцільно з метою оцінки ступеня дезорганізації ритмів будувати гістограми коефіцієнтів кореляції в кореляційної матриці. На рис. 2.4 видно, що у нормі значення гістограм

коефіцієнтів кореляцій зосереджені у сфері великих значень, і, навпаки, у пацієнта ці значення розкидані за значеннями коефіцієнтів кореляції.

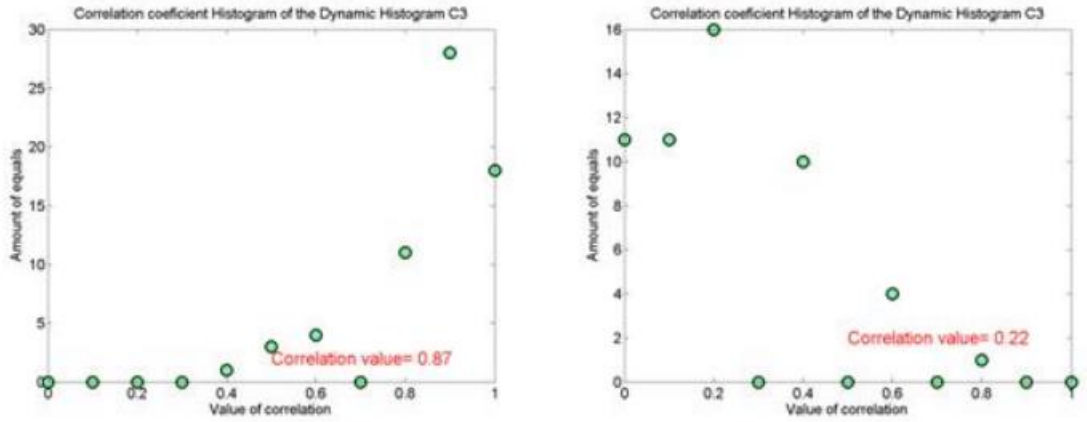


Рис. 2.4. Гістограми коефіцієнтів кореляції домінуючих ритмів у нормі (ліворуч) та пацієнта на другій стадії захворювання (праворуч).

При порівнянні ЕЕГ симетричних ділянок правої та лівої півкуль у обстежених пацієнтів постійно виявлялися суттєві відмінності у розподілі локальних максимумів вейвлет спектрограм. Ознаки дезорганізації ЕЕГ могли бути виражені сильніше або праворуч або ліворуч. Ці дані цілком відповідають уявленням про асиметрію перших проявів хвороби.

У нейрофізіології створено метод комп’ютерної реєстрації та кількісної оцінки тремору, що виникає при незмінному підтриманні пози суглобового кута. Метод дозволяє з широкого спектру електроміограми виділити частотний діапазон сигналу, який створює руховий акт. Основою методу є уявлення про те, що зусилля м’язів, що діють на суглоб, створюють рух, вигляд якого близький до кривої, що огинає ЕМГ.

Так як інформація про тремор руки лежить не в самому сигналі ЕМГ, а в його огинаючій, тому слід виділити окремо огинаючі ЕМГ. Огинаюча ЕМГ обчислюється з допомогою перетворення Гільберта.

Для виділення амплітуди та фази довільного сигналу u(t) (модульований високочастотний сигнал) необхідно створити на його основі аналітичний сигнал (4):

$$w(t)=u(t)+iv(t) \tag{2.11}$$

Дійсна частина аналітичного сигналу збігається з вихідним сигналом u(t). Уявна частина w(t) називається перетворенням Гільберта сигналу u(t). Обчислюється перетворення Гільберта так:

$$v(t)=\int_{-\infty}^{+\infty}\frac{u(\tau)}{\pi(t-\tau)}d(\tau) \tag{2.12}$$

Підставляючи (2.11) у формулу (2.12), можна ідентифікувати огинаючу ЕМГ:

$$w(t)=u(t)+iv(t)=a(t)e^{i\pi(\omega t)}, \tag{2.13}$$

де a(t) – огинаюча сигналу.

$$a(t)=\sqrt{(u(t))^2+(v(t))^2} \tag{2.14}$$

Оцифровані записи ЕЕГ були оброблені фільтром Баттерворта 4-го порядку для видалення частот 50 Гц і шуму на 100 Гц.

На рис. 2.5 і 2.6 представлені частотно-часові розподіли екстремумів вейвлет спектрограм ЕЕГ моторної зони мозку, а також екстремуми огинаючої ЕМГ та тремору в кінцівках.

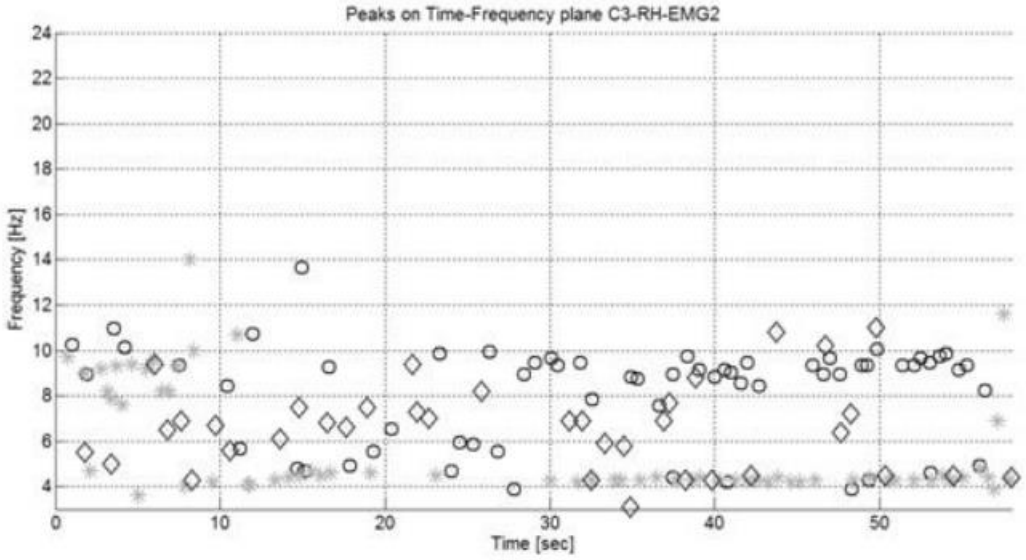


Рис. 2.5. Локальні максимуми на частотно-часовому діапазоні ЕЕГ у моторній зоні кори мозку (круги) та ЕМГ (ромби) хворого на першій стадії захворювання.

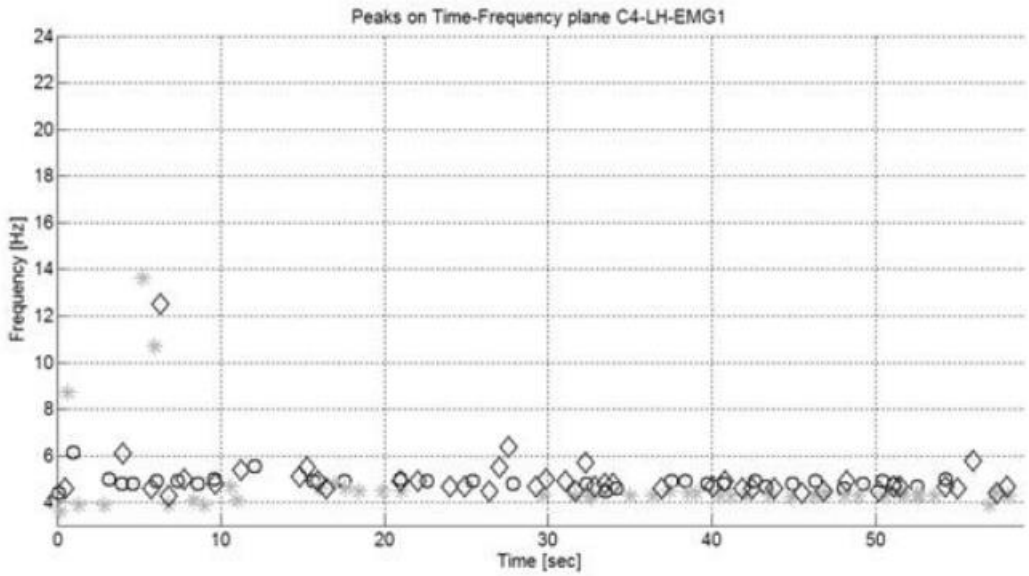


Рис. 2.6. Локальні максимуми на частотно-часовому діапазоні міжпівкульно-симетричного відведення.

Відповідні інтегральні гістограми розподілу частот локальних максимумів відображаються на рис. 2.7, 2.8.

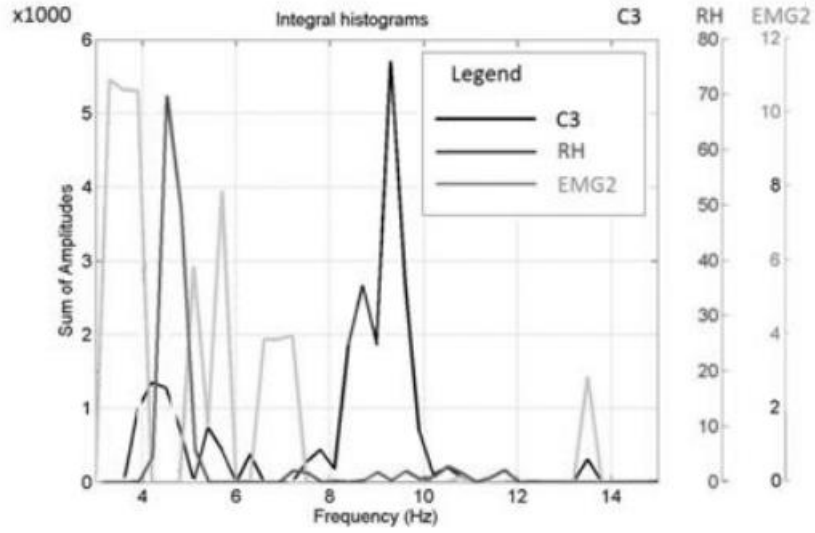


Рис. 2.7. Інтегральні гістограми частотних розподілів локальних максимумів (клінічно здорова півкуля) із частотною розсинхронізацією в тета-діапазоні.

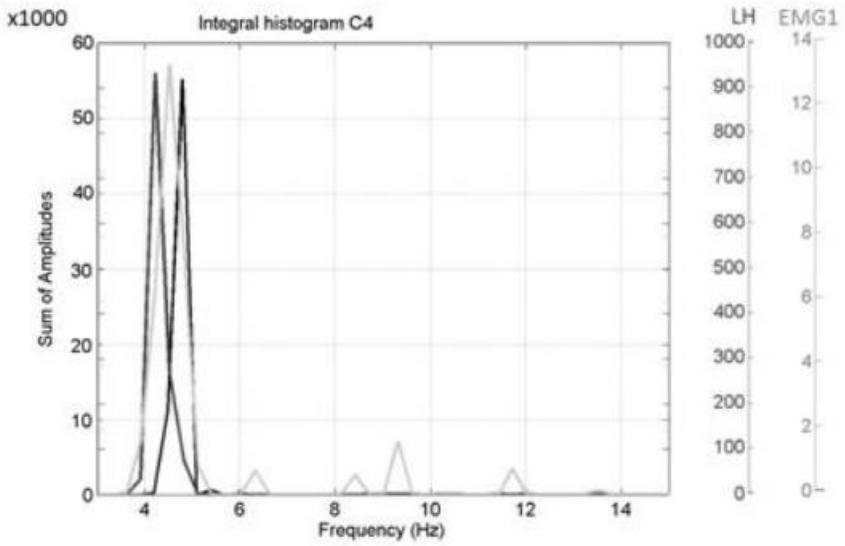


Рис. 2.8. Інтегральні гістограми частотних розподілів локальних максимумів (хвора півкуля) із частотною синхронізацією в тета-діапазоні.

На рис. 2.8 видно, що екстремуми в хворій моторній зоні правої півкулі частотно скорелювали з екстремумами МТ і ЕМГ. Навпаки, ще в клінічно здоровій лівій півкулі мозку такої кореляції немає (рис. 2.7).

За допомогою розроблених методів та програм отримані основні ознаки захворювання на ранній стадії: міжпівкульна асиметрія частотно-часових характеристик ЕЕГ; виникнення ритму в частотному діапазоні 4-6 Гц та його частотна синхронізація з електроміографічною активністю та механічним тремором кінцівок; дезорганізація домінуючого ритму ЕЕГ, що відповідає загальним уявленням про дезорганізацію різних систем при захворюванні. Отримано кількісні оцінки дезорганізації домінуючого ритму, які дозволяють розрізняти групи практично здорових людей від пацієнтів з хворобою на 1 стадії та пацієнтів на 1 стадії від пацієнтів на 2 стадії.

Таким чином, за допомогою вейвлет перетворення та його подальшого кількісного аналізу було підтверджено низку фактів, що характеризують особливості ЕЕГ на 2-3 стадіях захворювання, та виявлено низку специфічних особливостей частотно-часової організації ЕЕГ першої стадії захворювання.

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Розроблення інформаційної системи аналізу порушень рухів людини

Порушення координації рухів людини – це прогресуючий розлад центральної нервової системи, що впливає на її рухливість і викликає тремор і скутість в рухах. Це хронічне захворювання, нейродегенеративний розлад людини, що вражає нейрони мозку, ця хвороба щороку вражає досить багато людей. Симптоми захворювання починаються поступово, іноді починаючись із ледь помітного тремтіння кінцівок. Тремтіння є поширеним явищем, але розлад нервової системи людини також часто викликає скутість або уповільнення її рухів. До симптомів хвороби відносяться наступні.

Тремор. Тремор або тремтіння зазвичай починається в кінцівці, часто у руці або пальцях. Рука може тремтіти, коли вона знаходиться в стані спокою.

Уповільнення рухів. З часом хвороба може уповільнювати рухи людини, що ускладнює виконання нею простих завдань. Кроки людини можуть стати коротшими під час ходьби. Може бути важко встати зі стільця. Людина може волочити ноги, намагаючись ходити.

Жорсткі м’язи. Жорсткість м’язів може виникнути в будь-якій частині тіла, вони можуть бути болючими та обмежувати діапазон рухів. Також можливі порушення постави та рівноваги: постава може стати згорбленою або можуть виникнути проблеми з рівновагою під час ходіння чи стояння.

Втрата автоматичних рухів: може бути знижена здатність виконувати несвідомі рухи, включаючи моргання, усмішку або розмахування руками під час ходьби.

Зміна мови: людина може говорити тихо, швидко, невиразно або вагатися перед тим, як щось сказати.

Зміни в почерку: людині може стати важко писати, написане може бути дрібними буквами і нерозбірливим.

Проектування інформаційної системи містить такі етапи.

1. Імпорт необхідних бібліотек.

2. Завантаження набору даних.

3. Вивчення набору даних.

4. Візуалізація набору даних.

5. Пошук кореляції між вхідними даними.

6. Попередня обробка набору даних (розділення даних на набір для тестування та набір для навчання).

7. Проектування моделі.

8. Оцінка моделі.

9. Тестування моделі з використанням нових даних.

У цьому проекті з допомогою спроектованої інформаційної медичної системи можна виявляти наявність захворювання порушення координації рухів в окремих людей за різними факторами.

Спочатку імпортують всі необхідні бібліотеки Python для створення інформаційної системи прогнозування захворювання. Це Numpy для роботи з масивами даних, Pandas для обробки та аналізу даних, Matplotlib та Seaborn для візуалізації даних:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Потрібно завантажити класифікатор XGBClassifier:

```
from xgboost import XGBClassifier
```

Також потрібно завантажити з бібліотеки Scikit-Learn метрики точності accuracy_score, confusion_matrix, classification_report, roc_auc_score, plot_roc_curve для оцінки якості спроектованої інформаційної системи:

```
from sklearn.metrics import accuracy_score, confusion_matrix,
plot_confusion_matrix, classification_report, roc_auc_score,
plot_roc_curve
```

З бібліотеки Scikit-Learn імпортують функцію MinMaxScaler з пакету preprocessing попередньої обробки вхідних даних для їх масштабування:

```
from sklearn.preprocessing import MinMaxScaler
```

З бібліотеки Scikit-Learn імпортують функцію train_test_split для поділу вхідного набору даних на навчальну та тестову вибірки:

```
from sklearn.model_selection import train_test_split
```

На другому етапі потрібно завантажити вхідний набір даних:

```
data=pd.read_csv("1.data")
```

Розглянемо цей набір даних. За допомогою функції .columns можна отримати імена стовпців вхідного набору даних:

```
data.columns
```

```
Index(['name', 'MDVP:Fo(Hz)', 'MDVP:Fhi(Hz)', 'MDVP:Flo(Hz)', 'MDVP:Jitter(%)',
       'MDVP:Jitter(Abs)', 'MDVP:RAP', 'MDVP:PPQ', 'Jitter:DDP',
       'MDVP:Shimmer', 'MDVP:Shimmer(dB)', 'Shimmer:APQ3', 'Shimmer:APQ5',
       'MDVP:APQ', 'Shimmer:DDA', 'NHR', 'HNR', 'status', 'RPDE', 'DFA',
       'spread1', 'spread2', 'D2', 'PPE'],
      dtype='object')
```

Використавши функцію .shape, можна отримати відомості про кількість рядків і стовпців у цьому наборі даних:

```
data.shape
```

```
(195, 24)
```

Далі з допомогою функції head() датафрейму data та tail() отримують перші п’ять та останні п’ять записів з набору даних:

```
data.head()
```

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...	Shimmer:DDA
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.01109	0.04374	...	0.06545
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.01394	0.06134	...	0.09403
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	0.05233	...	0.08270
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.00698	0.01505	0.05492	...	0.08771
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.00908	0.01966	0.06425	...	0.10470

5 rows × 24 columns

```
data.tail()
```

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...	Shimmer:DDA	
190	phon_R01_S50_2	174.188	230.978	94.261	0.00459	0.00003	0.00263	0.00259	0.00790	0.04087	...	0.07008	0
191	phon_R01_S50_3	209.516	253.017	89.488	0.00564	0.00003	0.00331	0.00292	0.00994	0.02751	...	0.04812	0
192	phon_R01_S50_4	174.688	240.005	74.287	0.01360	0.00008	0.00624	0.00564	0.01873	0.02308	...	0.03804	0
193	phon_R01_S50_5	198.764	396.961	74.904	0.00740	0.00004	0.00370	0.00390	0.01109	0.02296	...	0.03794	0
194	phon_R01_S50_6	214.289	260.277	77.973	0.00567	0.00003	0.00295	0.00317	0.00885	0.01884	...	0.03078	0

5 rows × 24 columns

Для отримання інформації про датафрейм використовують такий фрагмент коду:

```
data.info
```

```
<bound method DataFrame.info of              name  MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  MDVP:Jitter(%)  \
0    phon_R01_S01_1      119.992       157.302        74.997         0.00784
1    phon_R01_S01_2      122.400       148.650       113.819         0.00968
2    phon_R01_S01_3      116.682       131.111       111.555         0.01050
3    phon_R01_S01_4      116.676       137.871       111.366         0.00997
4    phon_R01_S01_5      116.014       141.781       110.655         0.01284
..              ...          ...           ...           ...             ...
190  phon_R01_S50_2      174.188       230.978        94.261         0.00459
191  phon_R01_S50_3      209.516       253.017        89.488         0.00564
192  phon_R01_S50_4      174.688       240.005        74.287         0.01360
193  phon_R01_S50_5      198.764       396.961        74.904         0.00740
194  phon_R01_S50_6      214.289       260.277        77.973         0.00567

     MDVP:Jitter(Abs)  MDVP:RAP  MDVP:PPQ  Jitter:DDP  MDVP:Shimmer  ...  \
0             0.00007   0.00370   0.00554     0.01109       0.04374  ...
1             0.00008   0.00465   0.00696     0.01394       0.06134  ...
2             0.00009   0.00544   0.00781     0.01633       0.05233  ...
3             0.00009   0.00502   0.00698     0.01505       0.05492  ...
4             0.00011   0.00655   0.00908     0.01966       0.06425  ...
..                ...       ...       ...         ...           ...  ...
190           0.00003   0.00263   0.00259     0.00790       0.04087  ...
191           0.00003   0.00331   0.00292     0.00994       0.02751  ...
192           0.00008   0.00624   0.00564     0.01873       0.02308  ...
193           0.00004   0.00370   0.00390     0.01109       0.02296  ...
194           0.00003   0.00295   0.00317     0.00885       0.01884  ...

     Shimmer:DDA      NHR     HNR  status      RPDE       DFA   spread1  \
0         0.06545  0.02211  21.033       1  0.414783  0.815285 -4.813031
1         0.09403  0.01929  19.085       1  0.458359  0.819521 -4.075192
2         0.08270  0.01309  20.651       1  0.429895  0.825288 -4.443179
3         0.08771  0.01353  20.644       1  0.434969  0.819235 -4.117501
4         0.10470  0.01767  19.649       1  0.417356  0.823484 -3.747787
..            ...      ...     ...     ...       ...       ...       ...
190       0.07008  0.02764  19.517       0  0.448439  0.657899 -6.538586
191       0.04812  0.01810  19.147       0  0.431674  0.683244 -6.195325
192       0.03804  0.10715  17.883       0  0.407567  0.655683 -6.787197
193       0.03794  0.07223  19.020       0  0.451221  0.643956 -6.744577
194       0.03078  0.04398  21.209       0  0.462803  0.664357 -5.724056

      spread2        D2       PPE
0    0.266482  2.301442  0.284654
1    0.335590  2.486855  0.368674
2    0.311173  2.342259  0.332634
3    0.334147  2.405554  0.368975
4    0.234513  2.332180  0.410335
..        ...       ...       ...
190  0.121952  2.657476  0.133050
191  0.129303  2.784312  0.168895
192  0.158453  2.679772  0.131728
193  0.207454  2.138608  0.123306
194  0.190667  2.555477  0.148569

[195 rows x 24 columns]>
```

Для перевірки типу даних, які знаходяться в стовпцях, використовують функцію dtypes:

```
data.dtypes
```

```
name                 object
MDVP:Fo(Hz)         float64
MDVP:Fhi(Hz)        float64
MDVP:Flo(Hz)        float64
MDVP:Jitter(%)      float64
MDVP:Jitter(Abs)    float64
MDVP:RAP            float64
MDVP:PPQ            float64
Jitter:DDP          float64
MDVP:Shimmer        float64
MDVP:Shimmer(dB)    float64
Shimmer:APQ3        float64
Shimmer:APQ5        float64
MDVP:APQ            float64
Shimmer:DDA         float64
NHR                 float64
HNR                 float64
status                int64
RPDE                float64
DFA                 float64
spread1             float64
spread2             float64
D2                  float64
PPE                 float64
dtype: object
```

Більшість з них містить тип даних float24 за винятком стовпця status, який містить тип даних int64. На наступному етапі перевіряють даатфрейм на наявність нульових значень:

```
data.isnull().sum()
```

```
name                0
MDVP:Fo(Hz)         0
MDVP:Fhi(Hz)        0
MDVP:Flo(Hz)        0
MDVP:Jitter(%)      0
MDVP:Jitter(Abs)    0
MDVP:RAP            0
MDVP:PPQ            0
Jitter:DDP          0
MDVP:Shimmer        0
MDVP:Shimmer(dB)    0
Shimmer:APQ3        0
Shimmer:APQ5        0
MDVP:APQ            0
Shimmer:DDA         0
NHR                 0
HNR                 0
status              0
RPDE                0
DFA                 0
spread1             0
spread2             0
D2                  0
PPE                 0
dtype: int64
```

Інформація про набір вхідних даних 1.data. Цей набір даних складається з ряду біомедичних вимірювань голосу 31 людини, 23 з яких мають хворобу порушення координації рухів. Кожен стовпець у таблиці є певним показником голосу, а кожен рядок відповідає одному із 195 записів голосу цих осіб (стовпець name). Основне призначення цього набору вхідних даних полягає в тому, щоб відрізнити здорових людей від тих, хто страждає на розлади координації рухів, відповідно до стовпця status, в якому встановлено значення 0 для здорових і 1 для хворих людей.

Для виявлення порушень координації рухів людини було використано набір даних. Він базується на аналізі тембру голосу пацієнта (низький він чи високий). В цьому датасеті визначаються зміни в різних частотах голосу людини. Цей набір даних містить 24 стовпці та 195 записів. Розглянемо набір даних, який було використано в роботі. Для його створення були записані мовні сигнали здорових людей та людей, які мали вже ознаки захворювання. З цих записів було отримано різні параметри голосу, які були зведені в таблицю і було встановлено зв’язок між цими атрибутами.

Розглянемо вхідні дані набору даних.

name – назва суб’єкта і номер запису;

MDVP:Fo(Hz) – середня частота голосу;

MDVP:Fhi(Hz) – максимальна частота голосу;

MDVP:Flo(Hz) – мінімальна частота голосу;

MDVP:Jitter(%), **MDVP:Jitter(Abs)**, **MDVP:RAP**, **MDVP:PPQ**, **Jitter:DDP** – параметри зміни частоти голосу;

MDVP:Shimmer, **MDVP:Shimmer(dB)**, **Shimmer:APQ3**, **Shimmer:APQ5**, **MDVP:APQ**, **Shimmer:DDA** – вимірювання величини зміни амплітуди голосу;

NHR, **HNR** – співвідношення шумових і тональних компонентів голосу;

status – стан здоров'я пацієнта: 0 – здоровий, 1 – хворий;

RPDE, **D2** – нелінійні динамічні комплексні виміри;

DFA – експонента фрактального масштабування сигналу;

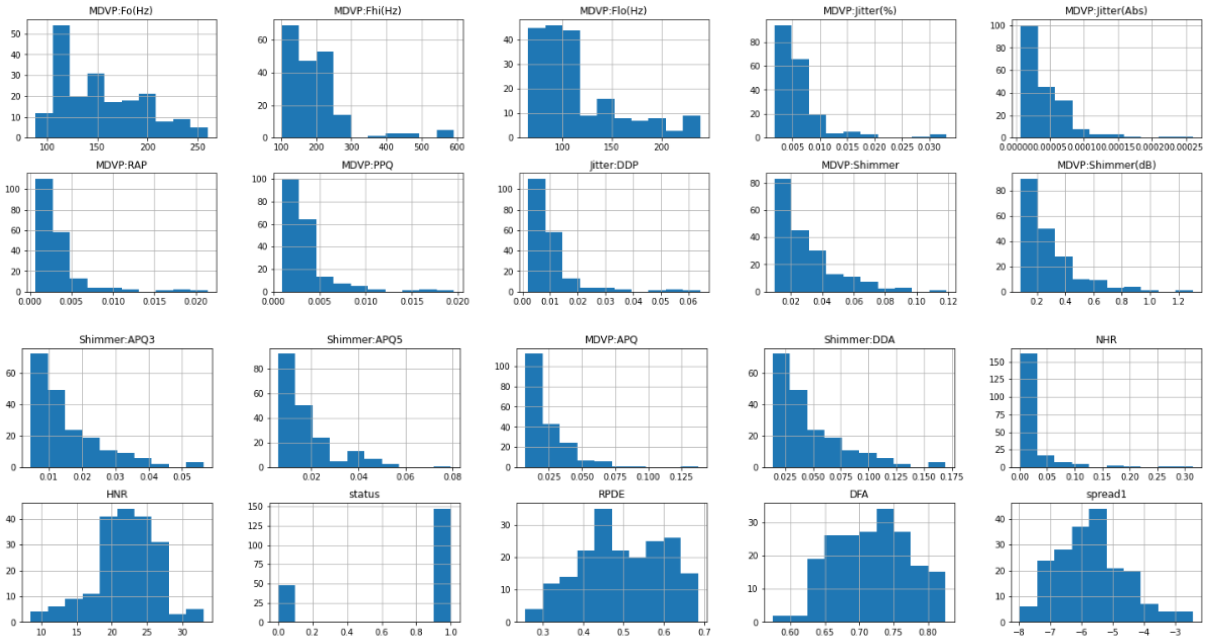
spread1, **spread2**, **PPE** – три нелінійні міри зміни частоти.

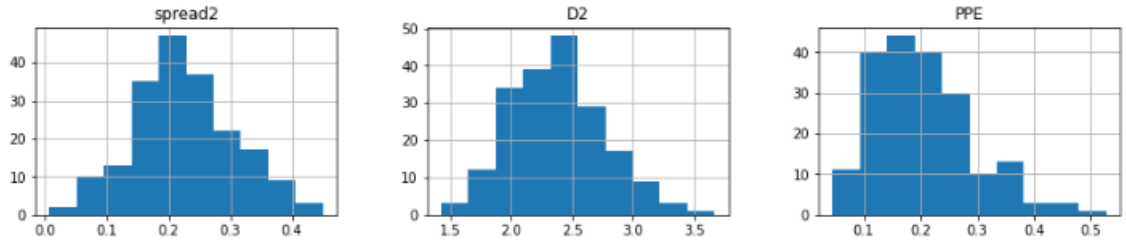
Оскільки немає жодних нульових значень, можна перейти до третього етапу розробки моделі – попередньої обробки даних EDA.

Побудуємо гістограми вхідних даних:

```
data.hist(figsize=(25,16))
plt.show()
```

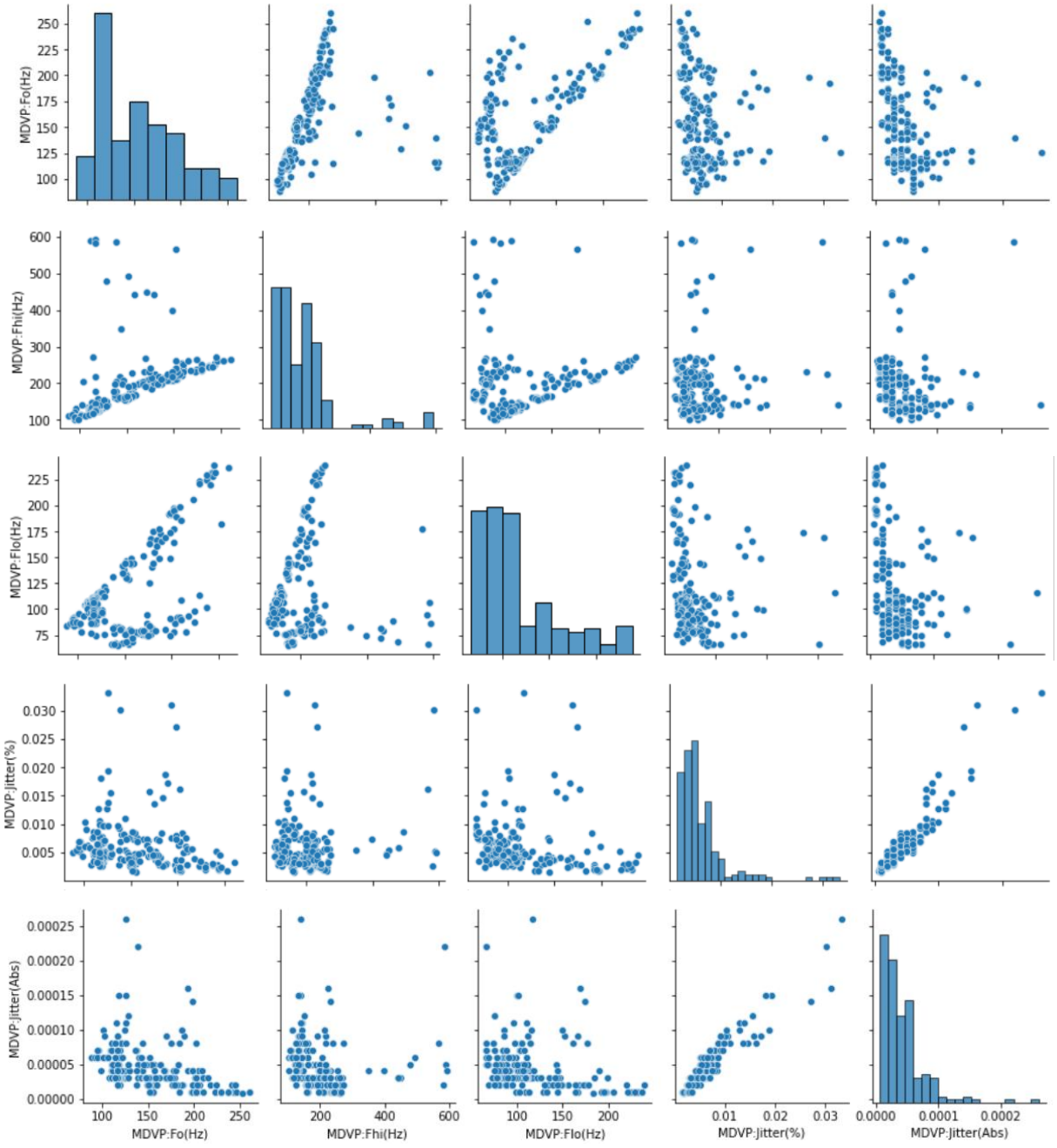
На ньому будуть представлені гістограми всіх значень стовпців:





Після цього будують графік pairplot для вхідних даних:

```
sns.pairplot(data.iloc[: ,0:6])
plt.show()
```



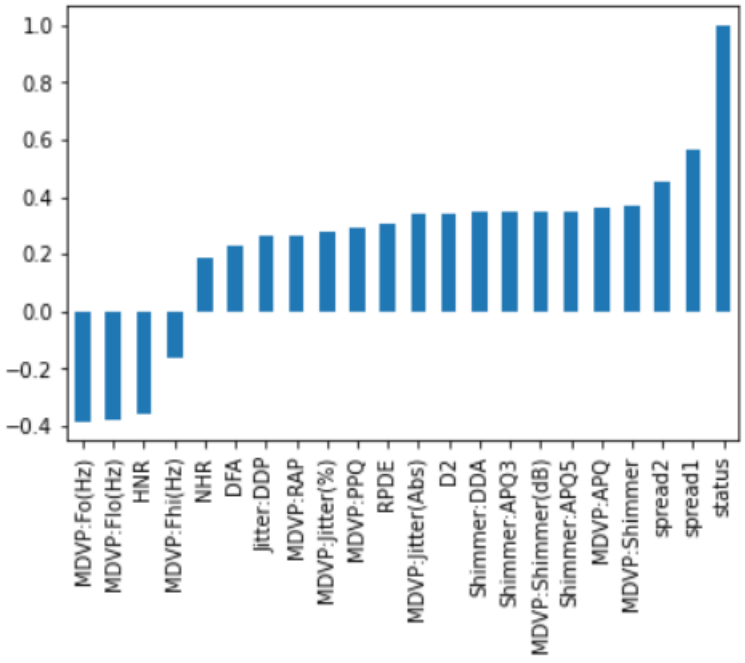
На наступному етапі аналізу даних переходять до кореляції між вхідними параметрами, після цього будують її графік з допомогою теплової карти:

```
data.corr()
```

Отримаємо:

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	MDVP:Shimmer(dB)	...	Shimmer:DDA
MDVP:Fo(Hz)	1.000000	0.400985	0.596546	-0.118003	-0.382027	-0.076194	-0.112165	-0.076213	-0.098374	-0.073742	...	-0.094732
MDVP:Fhi(Hz)	0.400985	1.000000	0.084951	0.102086	-0.029198	0.097177	0.091126	0.097150	0.002281	0.043465	...	-0.003733
MDVP:Flo(Hz)	0.596546	0.084951	1.000000	-0.139919	-0.277815	-0.100519	-0.095828	-0.100488	-0.144543	-0.119089	...	-0.150737
MDVP:Jitter(%)	-0.118003	0.102086	-0.139919	1.000000	0.935714	0.990276	0.974256	0.990276	0.769063	0.804289	...	0.746635
MDVP:Jitter(Abs)	-0.382027	-0.029198	-0.277815	0.935714	1.000000	0.922911	0.897778	0.922913	0.703322	0.716601	...	0.697170
MDVP:RAP	-0.076194	0.097177	-0.100519	0.990276	0.922911	1.000000	0.957317	1.000000	0.759581	0.790652	...	0.744919
MDVP:PPQ	-0.112165	0.091126	-0.095828	0.974256	0.897778	0.957317	1.000000	0.957319	0.797826	0.839239	...	0.763592
Jitter:DDP	-0.076213	0.097150	-0.100488	0.990276	0.922913	1.000000	0.957319	1.000000	0.759555	0.790621	...	0.744901
MDVP:Shimmer	-0.098374	0.002281	-0.144543	0.769063	0.703322	0.759581	0.797826	0.759555	1.000000	0.987258	...	0.987626
MDVP:Shimmer(dB)	-0.073742	0.043465	-0.119089	0.804289	0.716601	0.790652	0.839239	0.790621	0.987258	1.000000	...	0.963202
Shimmer:APQ3	-0.094717	-0.003743	-0.150747	0.746625	0.697153	0.744912	0.763580	0.744894	0.987625	0.963198	...	1.000000
Shimmer:APQ5	-0.070682	-0.009997	-0.101095	0.725561	0.648961	0.709927	0.786780	0.709907	0.982835	0.973751	...	0.960072
MDVP:APQ	-0.077774	0.004937	-0.107293	0.758255	0.648793	0.737455	0.804139	0.737439	0.950083	0.960977	...	0.896647
Shimmer:DDA	-0.094732	-0.003733	-0.150737	0.746635	0.697170	0.744919	0.763592	0.744901	0.987626	0.963202	...	1.000000
NHR	-0.021981	0.163766	-0.108670	0.906959	0.834972	0.919521	0.844604	0.919548	0.722194	0.744477	...	0.716215
HNR	0.059144	-0.024893	0.210851	-0.728165	-0.656810	-0.721543	-0.731510	-0.721494	-0.835271	-0.827805	...	-0.827130

```
data.corr()['status'][:-1].sort_values().plot(kind='bar')
plt.show()
```

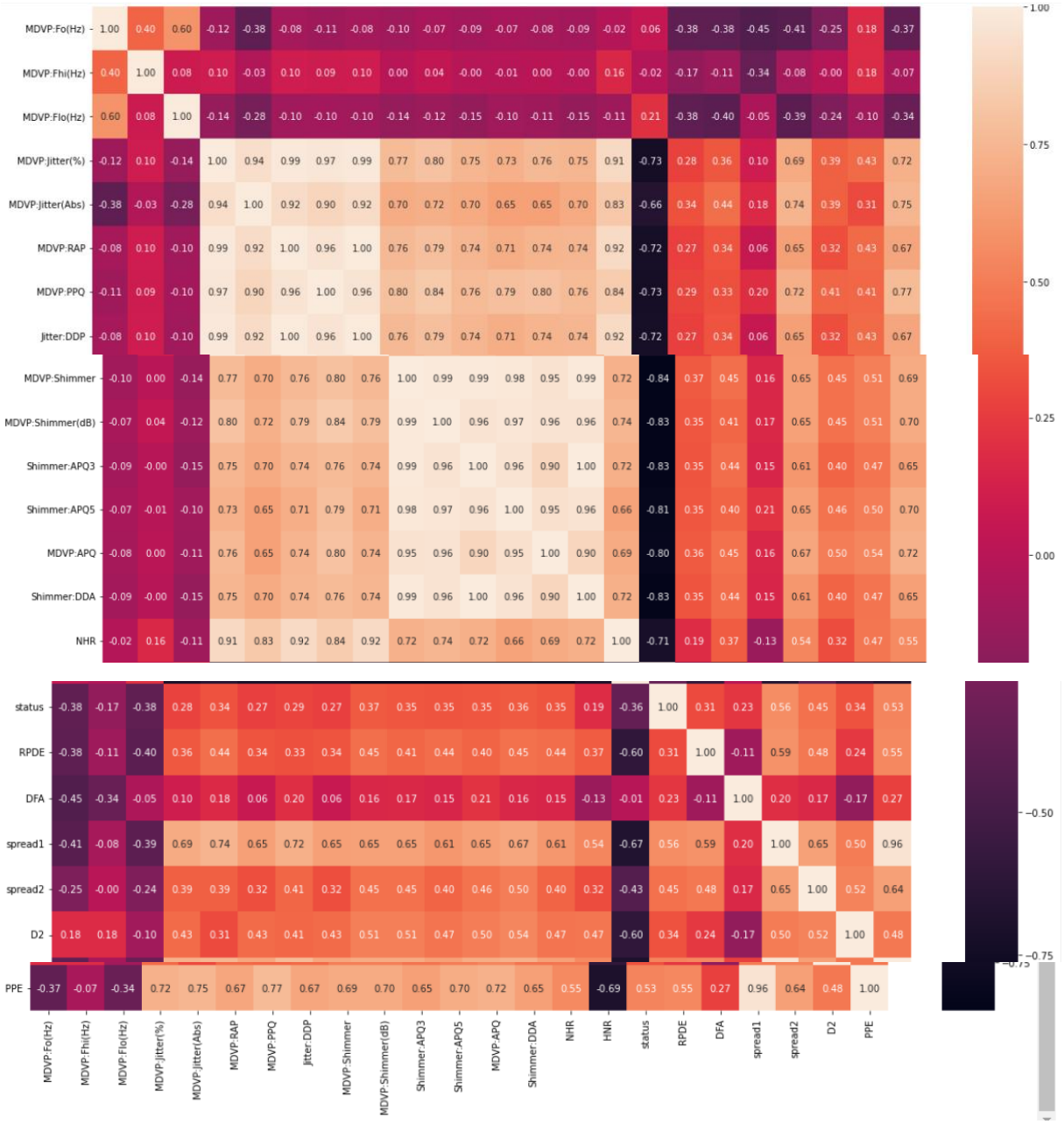


Отримаємо діаграму гістоподібного типу. Після цього використовують теплову карту, щоб отримати детальнішу картину. З цього графіка можна отримати інформацію про те, наскільки дані з кожного стовпця пов’язані з відповідним рядком:

```
f,ax = plt.subplots(figsize=(20, 20))
```

```
sns.heatmap(data.corr(), annot = True, fmt= '.2f')
plt.show()
```

Теплова карта виглядає таким чином:



Далі потрібно отримати ознаки (features) та мітки (labels) з датафрейму набору даних. Ознаки — це всі стовпці, крім стовпця з іменем status, а мітки — ті, що в стовпці status. Стовпець з іменем status – це цільова змінна, в якій знаходиться інформація про те, ци людина хвора чи здорова на дане захворювання рухового апарату. В ньому знаходяться значення 0 (людина здорова) та 1 (людина хвора):

```
features=data.loc[:,data.columns!='status'].values[:,1:]
```

```
labels=data.loc[:,'status'].values
```

Стовпець з іменем status має значення 0 і 1, отримаємо кількість цих значень для 0 та 1:

```
print(labels[labels==1].shape[0], labels[labels==0].shape[0])
```

```
147 48
```

Видно, що у вхідному наборі даних у стовпці status 147 одиниць та 48 нулів.

Далі потрібно ініціалізувати метод MinMaxScaler і відмасштабуввати ознаки від -1 до 1, щоб нормалізувати їх. MinMaxScaler перетворює ознаки, масштабуючи їх до заданого діапазону. Метод fit_transform() підходить до даних, а потім перетворює їх. Мітки не потрібно масштабувати:

```
scaler=MinMaxScaler((-1,1))
x=scaler.fit_transform(features)
y=labels
```

На наступному етапі проектування моделі потрібно розділити вхідні дані на навчальну та тестову вибірки. 20 % даних будуть використані для тестування:

```
x_train,x_test,y_train,y_test=train_test_split(x, y,
test_size=0.2, random_state=7)
```

Після цього переходять до створення моделі класифікатора XGBoost. Для цього ініціалізують класифікатор XGBClassifier з бібліотеки Scikit-Learn і проводять навчання моделі:

```
model = XGBClassifier(learning_rate=0.1, max_depth=10,
                      scale_pos_weight=1.5,
                      eval_metric='mlogloss')
model.fit(x_train, y_train)
```

```
XGBClassifier(eval_metric='mlogloss', max_depth=10, scale_pos_weight=1.5)
```

На наступному етапі потрібно згенерувати значення y_pred (прогнозовані значення для тестових значень x_test):

```
y_pred=model.predict(x_test)
```

Надалі потрібно розрахувати значення точності для моделі та отримати її результат:

```
print(accuracy_score(y_test, y_pred)*100)
```

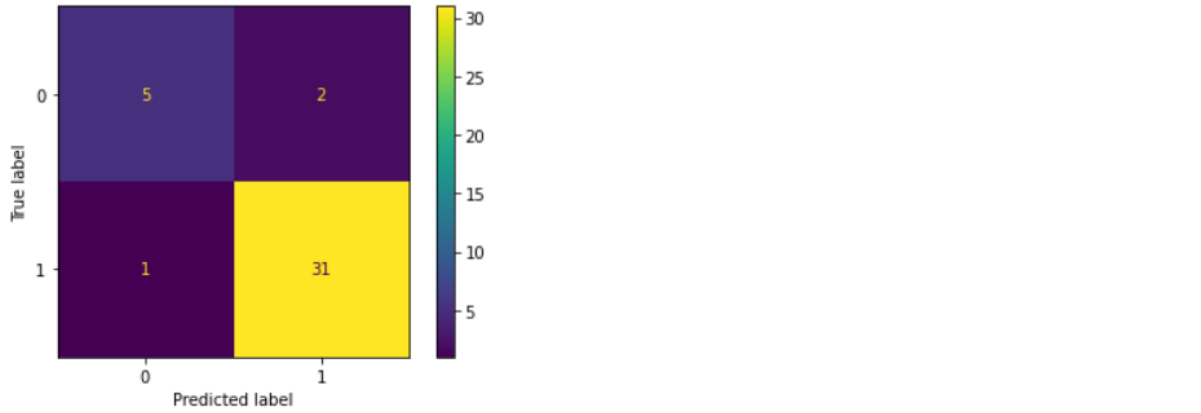
```
92.3076923076923
```

Точність моделі accuracy_score становить 92,3 %.

За отримання матриці Confusion Matrix для оцінки точності моделі відповідає нступний фрагмент коду:

```
print(confusion_matrix(y_test, y_pred))
plot_confusion_matrix(model, x_test, y_test)
plt.show()
```

```
[[ 5  2]
 [ 1 31]]
/usr/local/lib/python3.8/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function
  warnings.warn(msg, category=FutureWarning)
```



Для отримання та відображення класифікаційного звіту classification report:

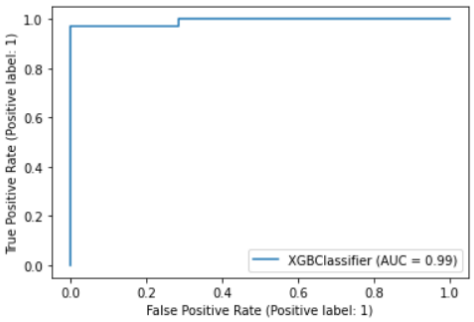
```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.83	0.71	0.77	7
1	0.94	0.97	0.95	32
accuracy			0.92	39
macro avg	0.89	0.84	0.86	39
weighted avg	0.92	0.92	0.92	39

Далі потрібно побудувати ROC-криву на основі тестових даних:

```
plot_roc_curve(model, x_test, y_test)
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_roc_curve is deprecated;
  warnings.warn(msg, category=FutureWarning)
```



Протестуємо результати роботи спроектованої інформаційної системи на довільному наборі даних для деякої людини, для визначення того, чи хвора людина на дане неврологічне захворювання чи ні. Тут ми використаємо модель для прогнозування результату на основі нових значень. Для цього потрібно підставити потрібні дані в модель і глянути на результат:

```
newinput=[[203.18400,211.52600,196.16000,0.00178,0.000009,
0.00094,0.00106,0.00283,0.00958,0.08500,0.00468,0.00610,
0.00726,0.01403,0.00065,33.04700,0.340068,0.741899,
-0.964984,0.163519,1.423287,0.044539]]

output=model.predict(newinput)
output
if output == 1:
    print(True)
else:
    print(False)
```

1 означає True (людина хвора), 0 означає False – людина здорова.

3.2. Проектування графічного інтерфейсу інформаційної системи з допомогою бібліотеки Streamlit

Розглянемо, як можна розробити графічний веб інтерфейс інформаційної системи на Python, використовуючи технології машинного навчання з допомогою бібліотеки Streamlit. Вона буде класифікувати захворювання координації рухів людини, відносячи їх до одного з двох видів: людина здорова (0), людина хвора (1). Програма буде передбачати ймовірність даного захворювання з врахуванням вхідних параметрів (22 ознаки захворювання).

Щоб побудувати модель і опублікувати її, було використано бібліотеки Pickle, Streamlit, Streamlit Option Menu. Бібліотека Pickle призначена для збереження попередньо навченої моделі машинного навчання та подальшого завантаження її у веб-додаток. Бібліотека Streamlit призначена безпосередньо для проектування веб-додатку інформаційної медичної системи та подальшого прогнозування захворювання. Бібліотека Streamlit Option Menu використана для акуратного розміщення елементів інтерфейсу та віджетів на веб сторінці у рядках та стовпцях.

Для інсталяції даних бібліотек потрібно запустити термінал і ввести в нього почергово наступні команди:

```
pip install pickle5
pip install streamlit
pip install streamlit-option-menu
```

Загальна структура проекту буде складатися з двох частин: фронтенду та бекенду. На фронтенд-частині програми, на веб-сторінці, буде знаходитися бокова панель, яка знаходиться зліва, і в яку можна буде вводити вхідні параметри моделі, які пов’язані з ознаками захворювання. Ці дані будуть передаватися бекенду, де попередньо навчена модель буде класифікувати це захворювання, використовуючи задані характеристики. Фактично мова йде про функцію, яка, отримавши ознаки хвороби,

визначає стан людини (0 – здорова, 1 – хвора). Результати класифікації відправляються назад фронтенду.

В бекенд-частині додатку те, що ввів користувач, зберігається у датафреймі, який буде використовуватися у виді тестових даних для моделі. Потім буде побудована модель для обробки цих даних. В ній буде використано алгоритм XGBoost з бібліотеки Scikit-Learn. В кінці модель буде застосована для класифікації даних, які ввів користувач, тобто для визначення стану людини.

Щоб запустити програму, потрібно зайти в командний рядок і ввести команду:

streamlit run 2.py.

Після цього запуститься веб-броузер за локальною адресою:

http://localhost:8501

Веб-програма складається із двох компонентів. Перший компонент – це головна панель, яка знаходиться в броузері справа. Другий компонент знаходиться зліва – це бокова панель. На боковій панелі знаходиться назва параметрів вводу користувача. На цій панелі приймаються ознаки захворювання, які призначені для майбутнього прогнозування.

Якщо змінити дані в текстових полях справа на головній панелі, даатфрейм з новими даними обновлюється. Числа в ньому будуть обновлені і відповідно сам прогноз буде змінений. Модель буде застосована для того, щоб отримати прогноз на основі нових введених даних.

Розглянемо код даного веб-додатку. Спочатку потрібно імпортувати бібліотеки Pickle, Streamlit, Streamlit Option Menu:

```
import pickle
import streamlit as st
from streamlit_option_menu import option_menu
```

Розглянемо, як сформувати бокову панель. Тут описують заголовок бокової панелі та ознаки хвороби, використавши функцію st.sidebar. Ця функція призначена для того, щоб помістити заголовок в бокову панель.

```
with st.sidebar:
    selected = option_menu('',
                           ['Ознаки хвороби'],
                           icons=['activity'],
                           default_index=0)
```

Далі переходять до проектування інтерфейсу головної панелі. На ній зверху буде розміщено заголовок веб-додатку, тобто для чого він призначений:

```
if (selected == "Ознаки хвороби"):
    st.title("Інформаційна система аналізу порушень
              координації рухів людини")
```

Якщо зліва на боковій панелі вибрати пункт меню Ознаки хвороби, справа на головній панелі вгорі появиться заголовок: Інформаційна система аналізу порушень координації рухів людини.

Після цього переходять до розміщення віджетів (елементів інтерфейсу) на головній панелі. В даній інформаційній системі є 22 вхідні параметри (ознаки хвороби). Вони приймають числові значення і тип даних в них float 64. Для вводу цих даних обрано віджети текстові поля. Крім того, так як цих полів буде досить багато, їх потрібно компактно розмістити для зручності вводу користувачам інформації. За це відповідає наступний фрагмент коду:

```
    col1, col2, col3, col4, col5 = st.columns(5)
    with col1:
        fo = st.text_input('MDVP:Fo(Hz)')
    with col2:
        fhi = st.text_input('MDVP:Fhi(Hz)')
    with col3:
        flo = st.text_input('MDVP:Flo(Hz)')
    with col4:
        Jitter_percent = st.text_input('MDVP:Jitter(%)')
```

```
with col5:
    Jitter_Abs = st.text_input('MDVP:Jitter(Abs)')
with col1:
    RAP = st.text_input('MDVP:RAP')
with col2:
    PPQ = st.text_input('MDVP:PPQ')
with col3:
    DDP = st.text_input('Jitter:DDP')
with col4:
    Shimmer = st.text_input('MDVP:Shimmer')
with col5:
    Shimmer_dB = st.text_input('MDVP:Shimmer(dB)')
with col1:
    APQ3 = st.text_input('Shimmer:APQ3')
with col2:
    APQ5 = st.text_input('Shimmer:APQ5')
with col3:
    APQ = st.text_input('MDVP:APQ')
with col4:
    DDA = st.text_input('Shimmer:DDA')
with col5:
    NHR = st.text_input('NHR')
with col1:
    HNR = st.text_input('HNR')
with col2:
    RPDE = st.text_input('RPDE')
with col3:
    DFA = st.text_input('DFA')
with col4:
    spread1 = st.text_input('spread1')
with col5:
    spread2 = st.text_input('spread2')
with col1:
    D2 = st.text_input('D2')
with col2:
    PPE = st.text_input('PPE')
```

На наступному етапі розробки створюють змінну, в яку буде присвоєно результати аналізу захворювання на основі введених вхідних параметрів:

```
diagnosis = ''
```

Після цього створюють командну кнопку з назвою Результат аналізу:

```
if st.button("Результат аналізу"):
    prediction = model.predict([[fo, fhi, flo,
Jitter_percent, Jitter_Abs, RAP, PPQ, DDP, Shimmer,
Shimmer_dB, APQ3, APQ5, APQ, DDA, NHR, HNR, RPDE, DFA,
spread1, spread2, D2, PPE]])
```

Натиснувши на неї, в змінну prediction передадуться результати аналізу нових введених вхідних параметрів (ознак захворювання). Для виводу повідомлення про результат роботи інформаційної медичної системи призначений наступний фрагмент коду:

```
if (prediction[0] == 1):
     diagnosis = "Людина має порушення координації рухів"
else:
     diagnosis = "Людина не має даного захворювання"
st.success(diagnosis)
```

Нижче приведено спроектований інтерфейс інформаційної системи:

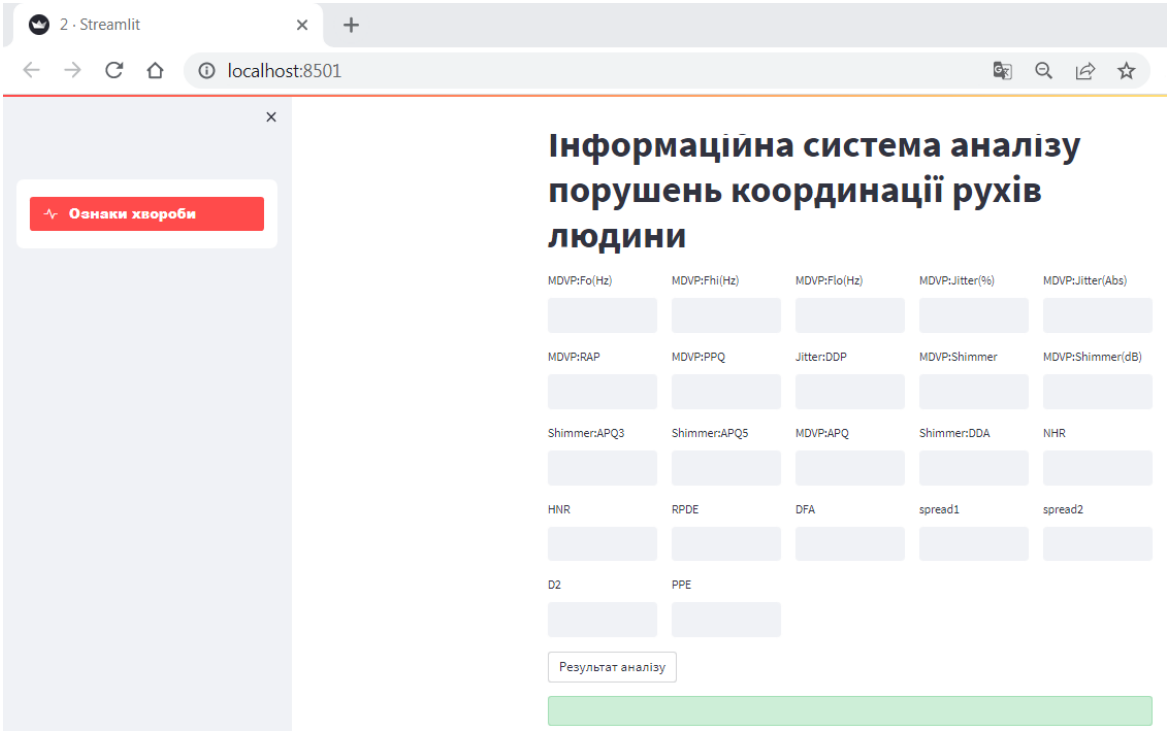


Рис. 3.1. Інтерфейс веб-додатку інформаційної системи аналізу порушення координації рухів людини.

3.3. Результати роботи інформаційної системи

Для перевірки коректності роботи інформаційної системи було проведено тестування якості її роботи. Для цього в якості ознак захворювання було взято дані для двох нових пацієнтів, даних про яких не було у вхідному наборі даних.

Дані для першого пацієнта (22 вхідні дані):

```
newinput=[[214.28900,260.27700,77.97300,0.00567,0.00003,
0.00295,0.00317,0.00885,0.01884,0.19000,0.01026,0.01161,
0.01373,0.03078,0.04398,21.20900,0.462803,0.664357,
-5.724056,0.190667,2.555477,0.148569]]
```

Ввівши необхідні вхідні параметри у текстові поля і натиснувши кнопку Результат аналізу, нижче появиться його результат: Людина має порушення координації рухів.

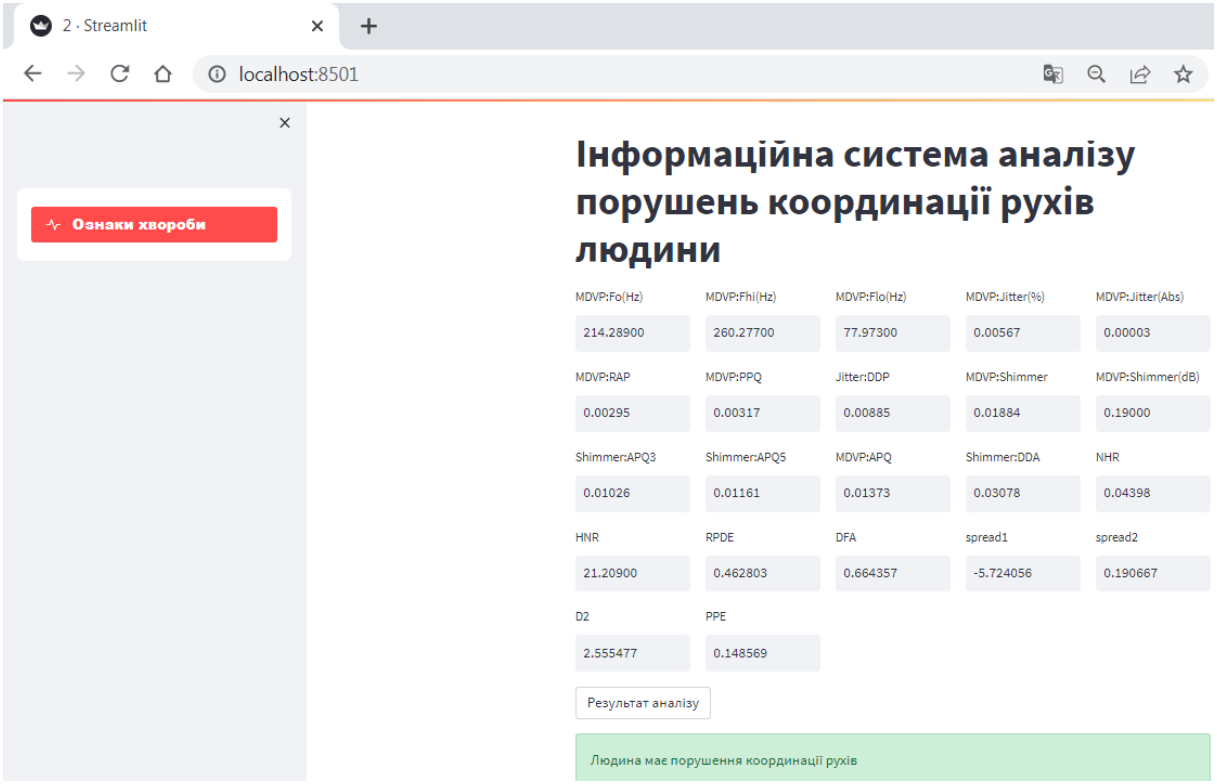


Рис. 3.2. Результати роботи інформаційної медичної системи для першого пацієнта.

Дані для другого пацієнта:

```
newinput=[[203.18400,211.52600,196.16000,0.00178,0.000009,
0.00094,0.00106,0.00283,0.00958,0.08500,0.00468,0.00610,
0.00726,0.01403,0.00065,33.04700,0.340068,0.741899,
-7.964984,0.163519,1.423287,0.044539]]
```

Ввівши дані в інформаційну систему, отримаємо результат: Людина не має даного захворювання.

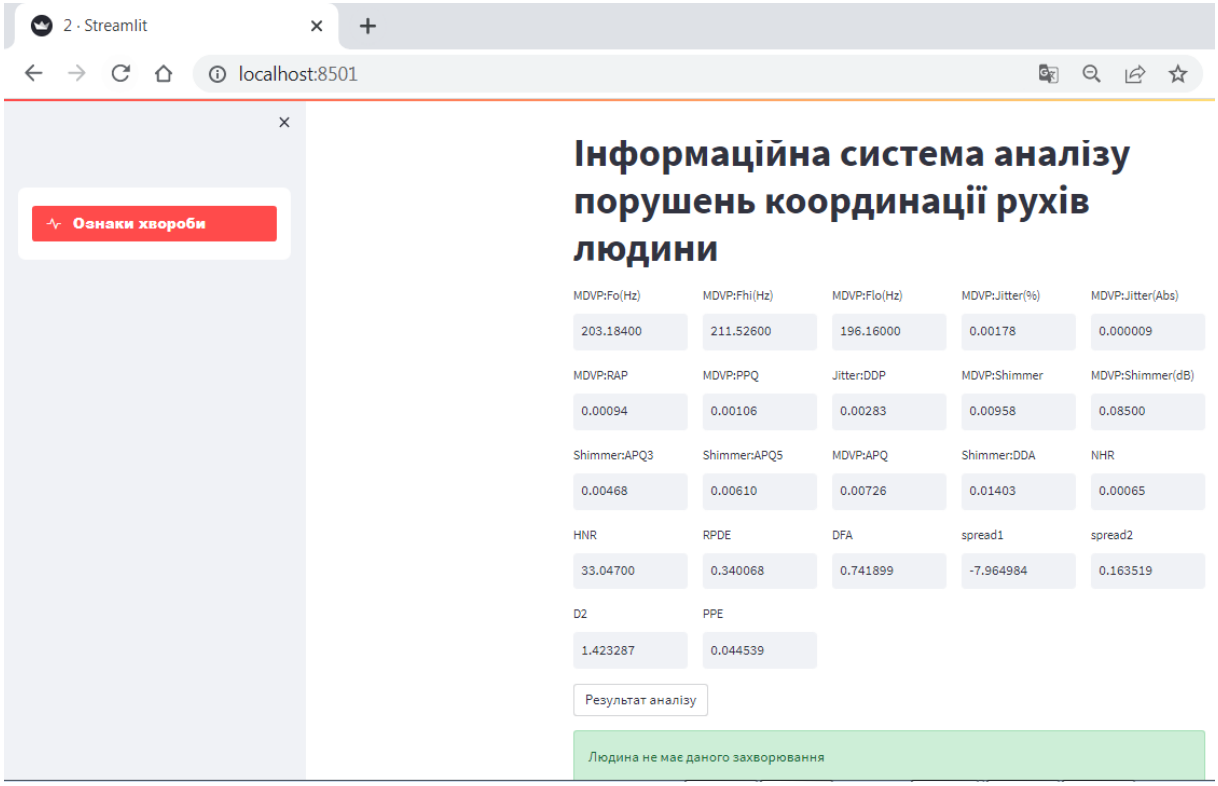


Рис. 3.3. Результати роботи інформаційної медичної системи для другого пацієнта.

Дана інформаційна система з точністю 92,3 % видає діагноз захворювання порушення координації рухів людини.

3.4. Характеристики апаратного та програмного забезпечення

Для проектування інформаційної медичної системи для виявлення патологій координації рухів використовувався персональний комп'ютер з такими характеристиками.

Табл. 3.1. Характеристики персонального комп'ютера

Процесор	Intel Pentium	Core i5
Материнська плата	Asus	ZY60-24-Premium
ОП	DDR3	4 GB
Відеокарта	Nvidia GeForce	1024 МГц
Жорсткий диск	Kingstone	300 Гб
Монітор	22″	Samsung

Для програмного забезпечення висуваються наступні вимоги:

Табл. 3.2. Програмне забезпечення

Операційна система	Windows 10
Середовище розробки	Python, Tkinter, Google Colab
Растровий графічний редактор	Adobe Photoshop CC
Векторний графічний редактор	CorelDraw X6
Текстовий редактор	Microsoft Word 2016

СПИСОК ЛІТЕРАТУРИ

1. Вьюгин В. Математические основы машинного обучения и прогнозирования / Москва: МЦНМО, 2018. – 384 с.

2. Домингос П. Верховный алгоритм: как машинное обучение изменит наш мир / Москва : Манн, Иванов и Фербер, 2016. – 336 с.

3. Флах П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных/ Москва: ДМК Пресс, 2015. – 400 с.

4. Маккинни У. Python и анализ данных / – М.: ДМК Пресс, 2015. – 486 с.

5. Любушкин А. А. Фрактальный анализ временных рядов / Москва: РГГРУ, 2006. – 22 с.

6. Малла С. Вейвлеты в обработке сигналов / Москва : Мир, 2005. – 671 с.

7. Немирко А. П., Манило Л. А., Калиниченко А. Н.. Математический анализ биомедицинских сигналов и данных / Санкт-Петербург: Физматлит, 2017. – 248 с.

8. Рангайян Р. М. Анализ биомедицинских сигналов. Практический подход / СПб.: Физматлит, 2010. – 440 с.

9. Страчунская Е. Я. Паркинсонизм с позиций современных информационных концепций медицины / Смоленск: СГМА, 2008. – 207 с.

10. Бернштейн Н. А. О построении движений / Москва : Медгиз, 1947. – 254 с.

11. Голубев В. Л., Левин Я. И., Вейн А. М.. Болезнь Паркинсона и синдром паркинсонизма / Москва : МЕДпресс, 1999. – 416 с.

12. Г. Н. Крыжановский, И. Н. Карабань, С. В. Магаева Болезнь Паркинсона (этиология, патогенез, клиника, диагностика, лечение, профилактика) и др. – Москва : Медицина, 2002. – 334 с.

13. Браверман Э. М., Мучник И. Б. Структурные методы обработки эмпирических данных / Москва : Наука, 1983. – 464 с.

14. Каменецкий В. К. Паркинсонизм / Санкт-Петербург: Питер, 2001. – 414 с.

15. Антонен Е. Г. Немоторные и дополнительные моторные расстройства при паркинсонизме / Петрозаводск: Издательство ПетрГУ, 2015. – 79 с.

16. Дик О. Е. Энергетические и фрактальные характеристики физиологического и патологического тремора руки человека / Физиология человека. – 2010. – Т. 36, № 2. – С. 92–100.

17. Дик О.Е. Нелинейная динамика непроизвольных колебаний руки человека при двигательной патологии / Физиология человека. – 2015. – Т. 41, № 2. – С. 53–59.

18. Иванова-Смоленцева И. А. Современные инструментальные методы регистрации тремора / Бюллетень национального общества по изучению болезни Паркинсона. – 2011. – № 2. – С. 17– 23.

19. Кобенко В. Ю. Фрактальная идентификационная шкала / Омский научный вестник. – 2009. – № 3.– С. 205–213.

20. Программно-аппаратный комплекс оценки состояния центральной нервной системы ПАК-ЦНС-01.

http://iairas.ru/pakcns01.php

ВИСНОВКИ

У дипломній роботі з допомогою мови програмування Python та відповідних бібліотек аналізу даних було спроектовано інформаційну систему для діагностування захворювання порушень координації рухів в окремих людей за різними факторами ризику (ознаками захворювання).

Спроектувавши за вхідними даними модель машинного навчання та провівши її навчання за допомогою класифікатора XGBClassifier з бібліотеки Scikit-Learn, за вхідними факторами можна визначати здорових та хворих людей. З допомогою даної моделі отримали точність класифікації 92 %, що є досить непоганим результатом. Було спроектовано веб-додаток для цієї інформаційної системи з допомогою бібліотеки Streamlit. У ньому, ввівши вхідні параметри (ознаки захворювання), отримують діагноз для пацієнтів.

Ця інформаційна медична система може бути використана лікарями-невропатологами для вивлення на ранній стадії захворювань порушень координації рухів окремих груп людей, пришвидшить можливість діагностики цього небезпечного захворювання.

ДОДАТКИ

ДОДАТОК А

1.ipynb

```
[1] import numpy as np
    import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt
```

```
[2] from xgboost import XGBClassifier
```

```
[3] from sklearn.metrics import accuracy_score, confusion_matrix, plot_confusion_matrix, classification_report, roc_auc_score, plot_roc_curve
```

```
[4] from sklearn.preprocessing import MinMaxScaler
```

```
[5] from sklearn.model_selection import train_test_split
```

```
[7] data=pd.read_csv("/content/data1.csv")
```

```
[8] data.columns
```

```
Index(['name', 'MDVP:Fo(Hz)', 'MDVP:Fhi(Hz)', 'MDVP:Flo(Hz)', 'MDVP:Jitter(%)',
       'MDVP:Jitter(Abs)', 'MDVP:RAP', 'MDVP:PPQ', 'Jitter:DDP',
       'MDVP:Shimmer', 'MDVP:Shimmer(dB)', 'Shimmer:APQ3', 'Shimmer:APQ5',
       'MDVP:APQ', 'Shimmer:DDA', 'NHR', 'HNR', 'status', 'RPDE', 'DFA',
       'spread1', 'spread2', 'D2', 'PPE'],
      dtype='object')
```

```
[9] data.shape
```

```
(195, 24)
```

```
[10] data.head()
```

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DD
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.011
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.013
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.016
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.00698	0.015
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.00908	0.019

5 rows × 24 columns

```
[11] data.tail()
```

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP
190	phon_R01_S50_2	174.188	230.978	94.261	0.00459	0.00003	0.00263	0.00259	0.00790
191	phon_R01_S50_3	209.516	253.017	89.488	0.00564	0.00003	0.00331	0.00292	0.00994
192	phon_R01_S50_4	174.688	240.005	74.287	0.01360	0.00008	0.00624	0.00564	0.01873
193	phon_R01_S50_5	198.764	396.961	74.904	0.00740	0.00004	0.00370	0.00390	0.01109
194	phon_R01_S50_6	214.289	260.277	77.973	0.00567	0.00003	0.00295	0.00317	0.00885

5 rows × 24 columns

```
[12] data.info
```

```
<bound method DataFrame.info of              name  MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  MDVP:Jitter(%)  \
0    phon_R01_S01_1      119.992       157.302        74.997         0.00784
1    phon_R01_S01_2      122.400       148.650       113.819         0.00968
2    phon_R01_S01_3      116.682       131.111       111.555         0.01050
3    phon_R01_S01_4      116.676       137.871       111.366         0.00997
4    phon_R01_S01_5      116.014       141.781       110.655         0.01284
..              ...          ...           ...           ...             ...
190  phon_R01_S50_2      174.188       230.978        94.261         0.00459
191  phon_R01_S50_3      209.516       253.017        89.488         0.00564
192  phon_R01_S50_4      174.688       240.005        74.287         0.01360
193  phon_R01_S50_5      198.764       396.961        74.904         0.00740
194  phon_R01_S50_6      214.289       260.277        77.973         0.00567

     MDVP:Jitter(Abs)  MDVP:RAP  MDVP:PPQ  Jitter:DDP  MDVP:Shimmer  ...  \
0             0.00007   0.00370   0.00554     0.01109       0.04374  ...
1             0.00008   0.00465   0.00696     0.01394       0.06134  ...
2             0.00009   0.00544   0.00781     0.01633       0.05233  ...
3             0.00009   0.00502   0.00698     0.01505       0.05492  ...
4             0.00011   0.00655   0.00908     0.01966       0.06425  ...
..                ...       ...       ...         ...           ...  ...
190           0.00003   0.00263   0.00259     0.00790       0.04087  ...
191           0.00003   0.00331   0.00292     0.00994       0.02751  ...
192           0.00008   0.00624   0.00564     0.01873       0.02308  ...
193           0.00004   0.00370   0.00390     0.01109       0.02296  ...
194           0.00003   0.00295   0.00317     0.00885       0.01884  ...

     Shimmer:DDA      NHR     HNR  status      RPDE       DFA   spread1  \
0        0.06545  0.02211  21.033       1  0.414783  0.815285 -4.813031
1        0.09403  0.01929  19.085       1  0.458359  0.819521 -4.075192
2        0.08270  0.01309  20.651       1  0.429895  0.825288 -4.443179
3        0.08771  0.01353  20.644       1  0.434969  0.819235 -4.117501
4        0.10470  0.01767  19.649       1  0.417356  0.823484 -3.747787
..           ...      ...     ...     ...       ...       ...       ...
190      0.07008  0.02764  19.517       0  0.448439  0.657899 -6.538586
191      0.04812  0.01810  19.147       0  0.431674  0.683244 -6.195325
192      0.03804  0.10715  17.883       0  0.407567  0.655683 -6.787197
193      0.03794  0.07223  19.020       0  0.451221  0.643956 -6.744577
194      0.03078  0.04398  21.209       0  0.462803  0.664357 -5.724056

      spread2        D2       PPE
0    0.266482  2.301442  0.284654
1    0.335590  2.486855  0.368674
2    0.311173  2.342259  0.332634
3    0.334147  2.405554  0.368975
4    0.234513  2.332180  0.410335
..        ...       ...       ...
190  0.121952  2.657476  0.133050
191  0.129303  2.784312  0.168895
192  0.158453  2.679772  0.131728
193  0.207454  2.138608  0.123306
194  0.190667  2.555477  0.148569

[195 rows x 24 columns]>
```

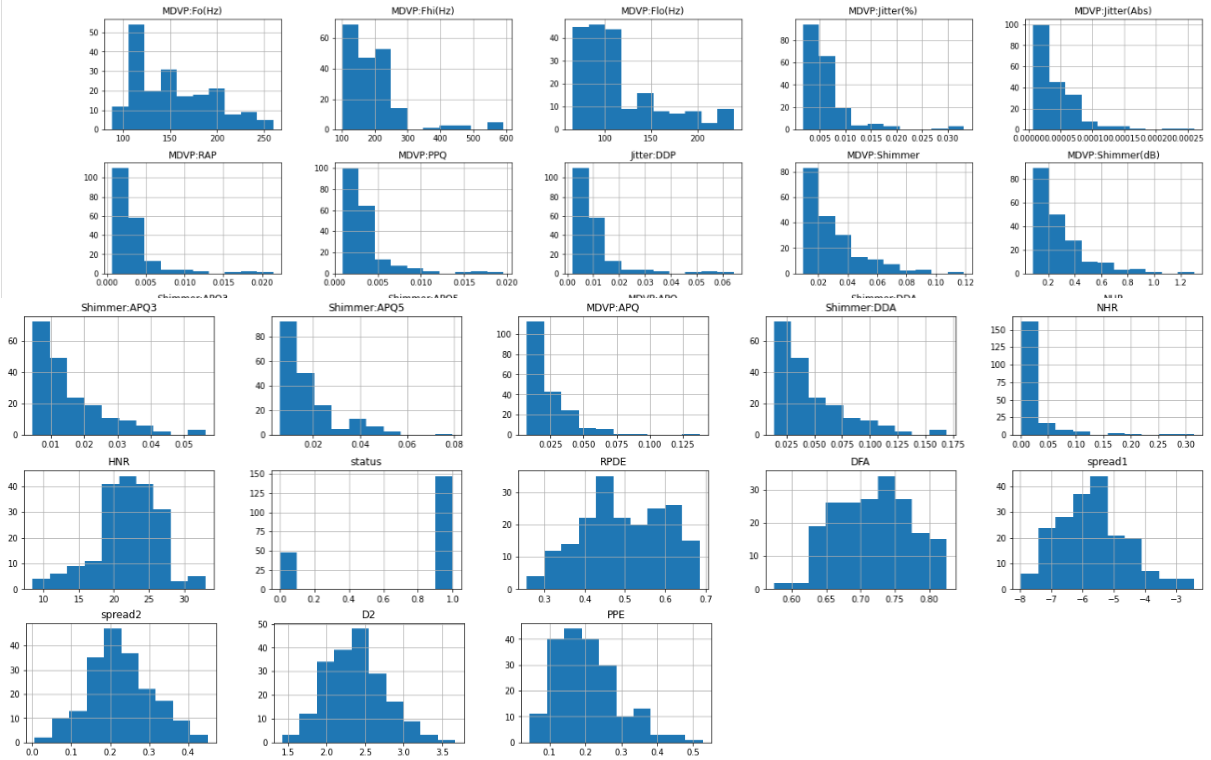
```
[13] data.dtypes
```

```
name                object
MDVP:Fo(Hz)        float64
MDVP:Fhi(Hz)       float64
MDVP:Flo(Hz)       float64
MDVP:Jitter(%)     float64
MDVP:Jitter(Abs)   float64
MDVP:RAP           float64
MDVP:PPQ           float64
Jitter:DDP         float64
MDVP:Shimmer       float64
MDVP:Shimmer(dB)   float64
Shimmer:APQ3       float64
Shimmer:APQ5       float64
MDVP:APQ           float64
Shimmer:DDA        float64
NHR                float64
HNR                float64
status               int64
RPDE               float64
DFA                float64
```

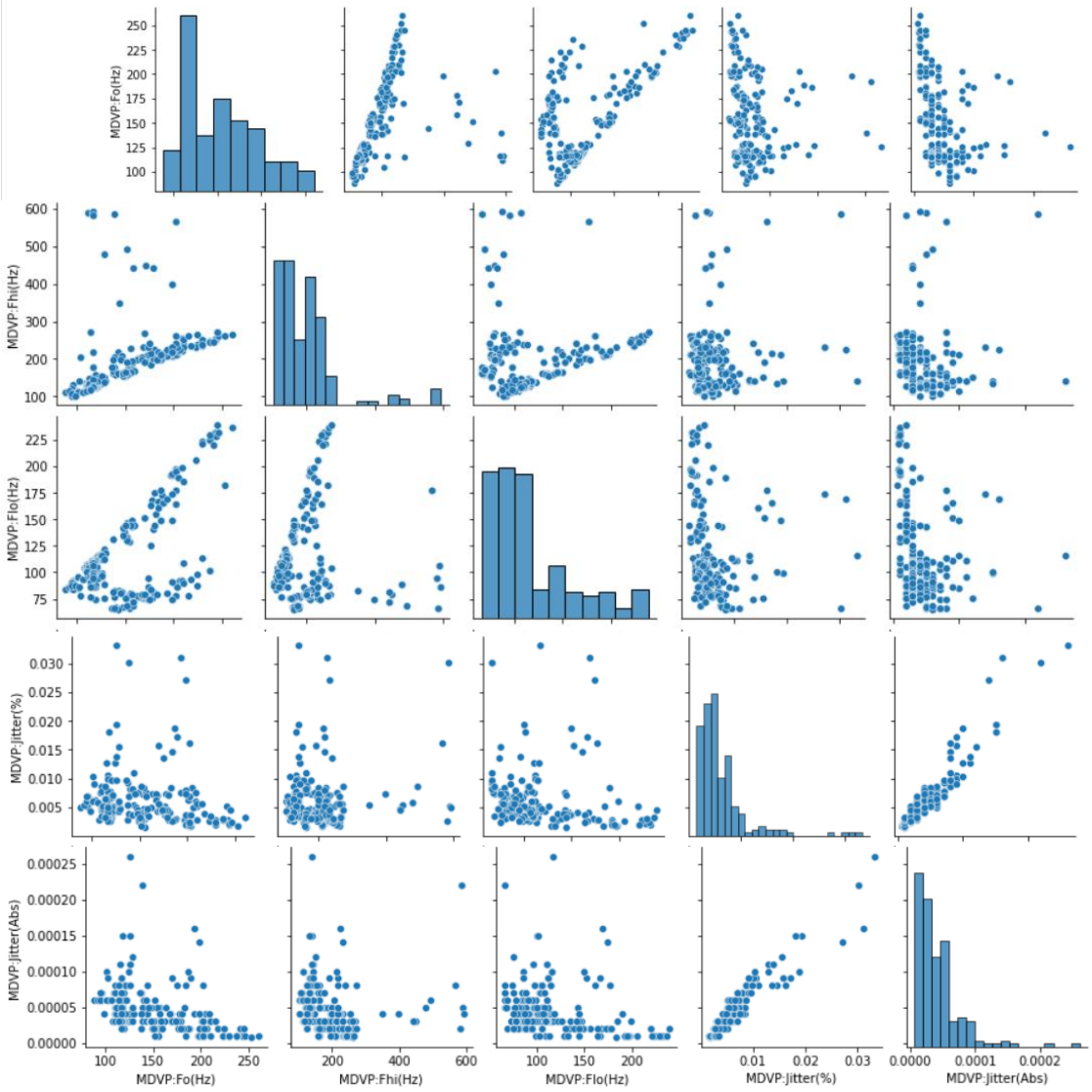
```
[14] data.isnull().sum()
```

```
name               0
MDVP:Fo(Hz)        0
MDVP:Fhi(Hz)       0
MDVP:Flo(Hz)       0
MDVP:Jitter(%)     0
MDVP:Jitter(Abs)   0
MDVP:RAP           0
MDVP:PPQ           0
Jitter:DDP         0
MDVP:Shimmer       0
MDVP:Shimmer(dB)   0
Shimmer:APQ3       0
Shimmer:APQ5       0
MDVP:APQ           0
Shimmer:DDA        0
```

```
[15] data.hist(figsize=(25,16))
     plt.show()
```



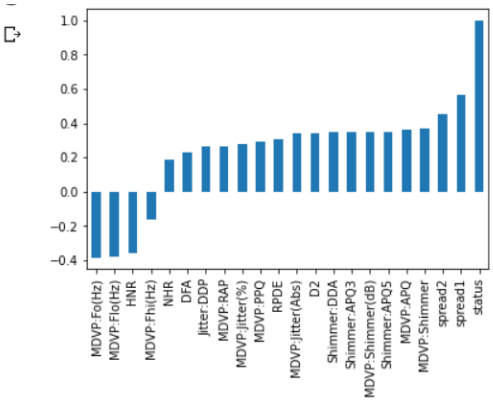
```
[16] sns.pairplot(data.iloc[: ,0:6])
     plt.show()
```



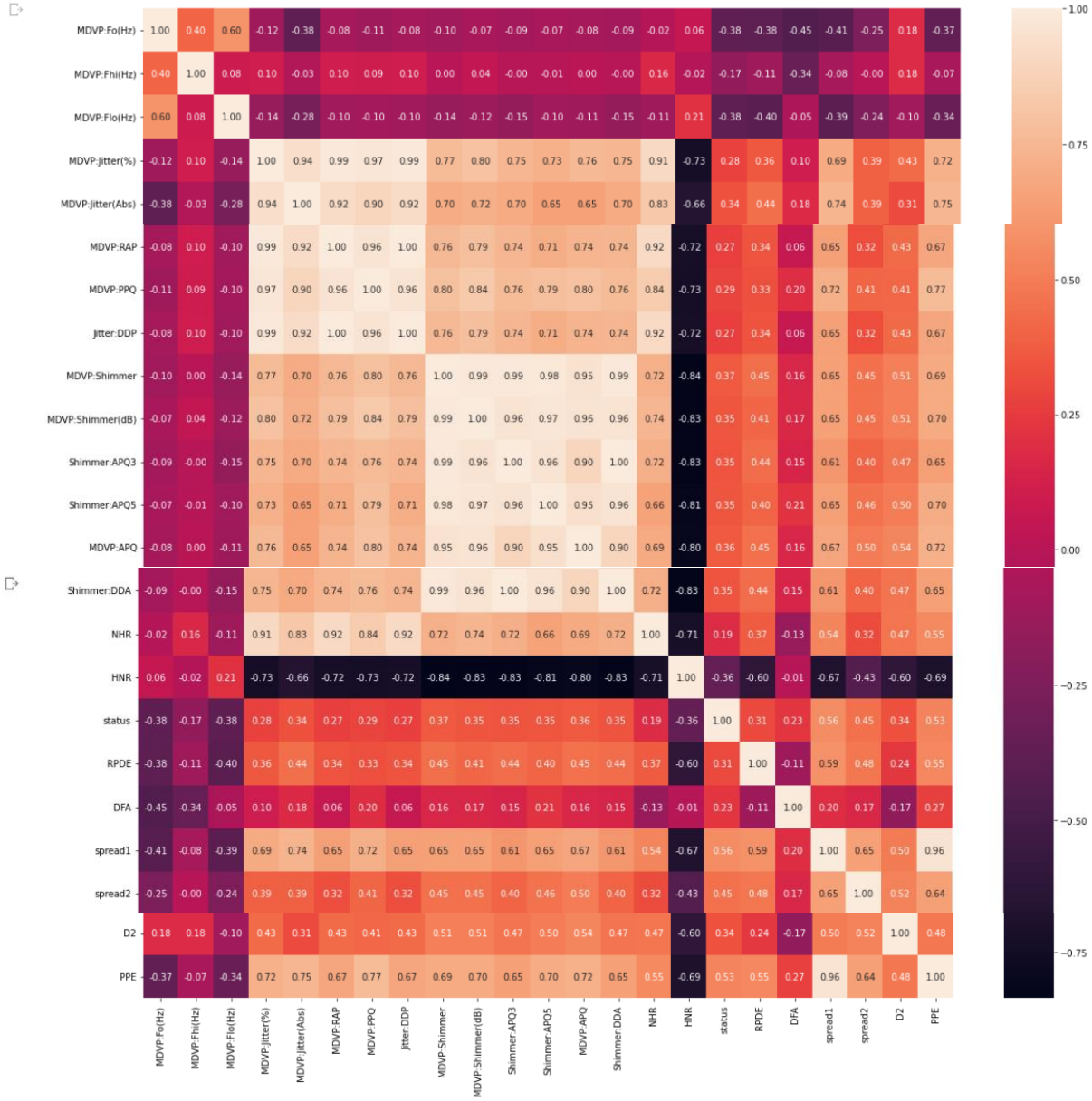

```
[17] data.corr()
```

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP
MDVP:Fo(Hz)	1.000000	0.400985	0.596546	-0.118003	-0.382027	-0.076194	-0.112165	-0.076213
MDVP:Fhi(Hz)	0.400985	1.000000	0.084951	0.102086	-0.029198	0.097177	0.091126	0.097150
MDVP:Flo(Hz)	0.596546	0.084951	1.000000	-0.139919	-0.277815	-0.100519	-0.095828	-0.100488
MDVP:Jitter(%)	-0.118003	0.102086	-0.139919	1.000000	0.935714	0.990276	0.974256	0.990276
MDVP:Jitter(Abs)	-0.382027	-0.029198	-0.277815	0.935714	1.000000	0.922911	0.897778	0.922913
MDVP:RAP	-0.076194	0.097177	-0.100519	0.990276	0.922911	1.000000	0.957317	1.000000
MDVP:PPQ	-0.112165	0.091126	-0.095828	0.974256	0.897778	0.957317	1.000000	0.957319

```
[18] data.corr()['status'][:-1].sort_values().plot(kind='bar')
     plt.show()
```




```
[19] f,ax = plt.subplots(figsize=(20, 20))
     sns.heatmap(data.corr(), annot = True, fmt= '.2f')
     plt.show()
```



```
[20] features=data.loc[:,data.columns!='status'].values[:,1:]
     labels=data.loc[:,'status'].values
```

```
[21] print(labels[labels==1].shape[0], labels[labels==0].shape[0])
```

```
147 48
```

```
[22] scaler=MinMaxScaler((-1,1))
     x=scaler.fit_transform(features)
     y=labels
```

```
[23] x_train,x_test,y_train,y_test=train_test_split(x, y, test_size=0.2, random_state=7)
```

```
[24] model = XGBClassifier(learning_rate=0.1, max_depth=10,
                           scale_pos_weight=1.5, eval_metric='mlogloss')
```

```
[25] model.fit(x_train, y_train)
```

```
XGBClassifier(eval_metric='mlogloss', max_depth=10, scale_pos_weight=1.5)
```

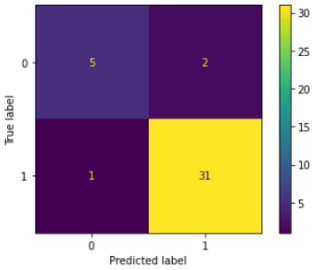
```
[26] y_pred=model.predict(x_test)
```

```
[27] print(accuracy_score(y_test, y_pred)*100)
```

```
92.3076923076923
```

```
[28] print(confusion_matrix(y_test, y_pred))
     plot_confusion_matrix(model, x_test, y_test)
     plt.show()
```

```
[[ 5  2]
 [ 1 31]]
/usr/local/lib/python3.8/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function
  warnings.warn(msg, category=FutureWarning)
```




```
[29] print(classification_report(y_test, y_pred))
```

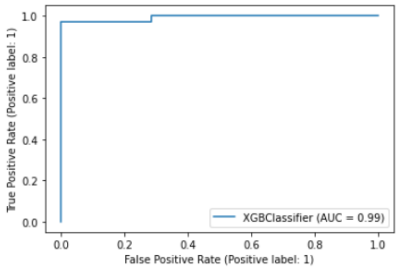
```
                precision    recall  f1-score   support

           0       0.83      0.71      0.77         7
           1       0.94      0.97      0.95        32

    accuracy                           0.92        39
   macro avg       0.89      0.84      0.86        39
weighted avg       0.92      0.92      0.92        39
```

```
[30] plot_roc_curve(model, x_test, y_test)
     plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_roc_curve is deprecated; Function :func:`
  warnings.warn(msg, category=FutureWarning)
```




```
[31] newinput=[[122.40000,148.65000,113.81900,0.00968,0.00008,0.00465,0.00696,0.01394,0.06134,0.62600,0.03134,0.04518,0.04368,0.09403,0.01929,19.0
```

```
[32] newinput=[[203.18400,211.52600,196.16000,0.00178,0.000009,0.00094,0.00106,0.00283,
     0.00958,0.08500,0.00468,0.00610,0.00726,0.01403,0.00065,33.04700,0.340068,
     0.741899, -7.964984,0.163519,1.423287,0.044539]]
```

```
[33] output=model.predict(newinput)
     output
     if output == 1:
         print(True)
     else:
         print(False)
```

```
True
```

ДОДАТОК Б

2.py

```
import streamlit as st
from streamlit_option_menu import option_menu

# Бічна панель
with st.sidebar:
    selected = option_menu('',
                           ['Ознаки хвороби'],
                           icons=['activity'],
                           default_index=0)

# Головна панель
if (selected == "Ознаки хвороби"):
    st.title("Інформаційна система аналізу порушень
координації рухів людини")
    col1, col2, col3, col4, col5 = st.columns(5)
    with col1:
        fo = st.text_input('MDVP:Fo(Hz)')
    with col2:
        fhi = st.text_input('MDVP:Fhi(Hz)')
    with col3:
        flo = st.text_input('MDVP:Flo(Hz)')
    with col4:
        Jitter_percent = st.text_input('MDVP:Jitter(%)')
    with col5:
        Jitter_Abs = st.text_input('MDVP:Jitter(Abs)')
    with col1:
        RAP = st.text_input('MDVP:RAP')
    with col2:
        PPQ = st.text_input('MDVP:PPQ')
    with col3:
        DDP = st.text_input('Jitter:DDP')
    with col4:
```

```
        Shimmer = st.text_input('MDVP:Shimmer')
    with col5:
        Shimmer_dB = st.text_input('MDVP:Shimmer(dB)')
    with col1:
        APQ3 = st.text_input('Shimmer:APQ3')
    with col2:
        APQ5 = st.text_input('Shimmer:APQ5')
    with col3:
        APQ = st.text_input('MDVP:APQ')
    with col4:
        DDA = st.text_input('Shimmer:DDA')
    with col5:
        NHR = st.text_input('NHR')
    with col1:
        HNR = st.text_input('HNR')
    with col2:
        RPDE = st.text_input('RPDE')
    with col3:
        DFA = st.text_input('DFA')
    with col4:
        spread1 = st.text_input('spread1')
    with col5:
        spread2 = st.text_input('spread2')
    with col1:
        D2 = st.text_input('D2')
    with col2:
        PPE = st.text_input('PPE')

    diagnosis = ''

    if st.button("Результат аналізу"):
        prediction = ab_model.predict([[fo, fhi, flo,
Jitter_percent, Jitter_Abs, RAP, PPQ, DDP, Shimmer,
Shimmer_dB, APQ3, APQ5, APQ, DDA, NHR, HNR, RPDE, DFA,
spread1, spread2, D2, PPE]])
```

```
    if (prediction[0] == 1):
      diagnosis = "Людина має порушення координації рухів"
    else:
      diagnosis = "Людина не має даного захворювання"

st.success(diagnosis)
```