Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет __Комп'ютерної інженерії та управління_____

(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки_____

(повна назв

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти ____другий (магістерський)_____

(рівень вищої освіти)

Метод автоматизованого детектування серцево-судинних захворювань на основі даних пульсометру

(тема)

Виконав: студент 2 курсу, групи СКСм-21-1

_____Клєщов В.М._____

(прізвище, ініціали)

Спеціальність 123 Комп'ютерна інженерія та управління

(код і повна назва спеціальності)

Тип програми __освітньо-професійна_____

(освітньо-професійна або освітньо-наукова)

Освітня програма _Спеціалізовані комп'ютерні системи_____

(повна назва освітньої програми)

Керівник _____Філіппенко І.В_____

(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____- __ ____Чумаченко С.В._____

(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Автоматизації проектування обчислювальної техніки

Рівень вищої освіти другий (магістерський)

Спеціальність 123 Комп'ютерна інженерія
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Спеціалізовані комп'ютерні системи
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____ 20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Клєщову Василю Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Метод автоматизованого детектування серцево-судинних захворювань на основі даних пульсометру

затверджена наказом університету від 14 листопада 2022 р. № 1478Ст

2. Термін подання студентом роботи до екзаменаційної комісії ____ _____ 2022 р.

3. Вихідні дані до роботи
нейронні мережі
пульсометри

4. Перелік питань, що потрібно опрацювати в роботі

Аналіз предметної області та постановка задачі.

Розробка моделі

Програмна реалізація

Аналіз отриманих даних.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) _____
_____
_____
_____
_____
_____
_____
_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання	04.11.2022 - 05.11.2022	
2	Аналіз літератури	12.11.2022 - 16.11.2022	
3	Розробка моделі	17.11.2022 - 26.11.2022	
4	Реалізація алгоритму оптимізації	29.11.2022 - 05.12.2022	
5	Тестування отриманих даних	06.12.2022 - 10.12.2022	
6	Оформлення пояснювальної записки	13.12.2022 - 17.12.2022	

Студент _____
(підпис)

Керівник роботи _____ _____ ___доц. Філіппенко І.В.___
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 51 сторінку, 18 рисунків, 31 джерело за переліком посилань.

СЕРЦЕВО-СУДИННІ ЗАХВОРЮВАННЯ, ЕЛЕКТРОКАРДІОГРАМА, ВАРІАБЕЛЬНІСТЬ СЕРЦЕВОГО РИТМУ, ШТУЧНІ НЕЙРОННІ МЕРЕЖІ, НЕЙРОМОРФНІ КОМП'ЮТЕРИ, МІКРОКОНТРОЛЕРИ, ІМПУЛЬСНІ НЕЙРОННІ МЕРЕЖІ, МОДЕЛІ НЕЙРОНІВ, МУЛЬТИСТАБІЛЬНІ НЕЙРОНИ

Метою роботи є дослідження методів автоматизованого аналізу показників пульсометра для виявлення аномалій в роботі серцево-судинної системи. В якості основного напрямку дослідження взяті методи машинного навчання та нейронні мережі для аналізу сигналів.

Для виконання поставленої задачі було проведено дослідження різних методів прийняття рішень, оцінено їх точність та продуктивність, а також обрано вид для подальшого дослідження та розробки моделі нейронної мережі, що детектує аномальні події роботи серцево-судинної системи.

ABSTRACT

Master’s thesis contains 51 pages, 18 figures, 31 sources according to the list of links.

CARDIOVASCULAR DISEASES, ELECTROCARDIOGRAM, HEART RATE VARIABILITY, ARTIFICIAL NEURAL NETWORKS, NEUROMORPHIC COMPUTERS, MICROCONTROLLERS, SPIKING NEURAL NETWORKS, NEURON MODELS, MULTISTABLE NEURONS

The purpose of this project is to research methods of the automated analysis of the pulsometer readings to detect cardiovascular system anomalies. Machine learning algorithms and neural networks were chosen as a main course for the work.

The research of the different decision making methods and neural networks was conducted, as well as their accuracy and performance was evaluated. A network type was chosen for the implementation of the system that will detect cardiovascular system anomalies.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ДСР - датчик серцевого ритму

ВСР - варіабельність серцевого ритму

РНН - рекурентна нейронна мережа

ШНМ - штучна нейронна мережа

СПЗШ - синдром передчасного збудження шлуночків серця

МПІ - міжпіковий інтервал

СЗПР - середньоквадратичне значення послідовних різниць

СВНН - стандартне відхилення від норми до норми

ВРЧ - виявлення у реальному часі

КПНМ - каскадна пряма нейронна мережа

ССЗ – Серцево-Судинні Захворювання

ЕКГ – Електрокардіографія

ШІ – Штучний Інтелект

IoT – Internet of Things

ANN – Artificial Neural Network

ШНМ – Штучна Нейронна Мережа

CNN – Convolutional Neural Network

ЗНМ – Згорткова Нейронна Мережа

LR – Linear Regression

SVR – Support Vector Regression

Machine learning – міждисципліонарна сфера знань, що використовує статистичні підходи для надання комп’ютеру можливості «вчитися» на наборах даних без програмування якимось специфічним алгоритмом

Deep learning – тип машинного навчання, що використовує алгоритми в багатошарових нейронних мережах для обробки великих об’ємів даних

ВСТУП

Штучний інтелект та додатки, що його використовують, різноманітно використовуються в медицині, науці та інших технологіях. Можливість використання “розумних” машин в медицині вивчалася з 1960-х років. Завдяки багатьом відкриттям у сфері обчислювання, розробці алгоритмів машинного навчання, особливо мереж глибокого навчання що імітують роботу людського мозку, інтерес до використання їх в клінічній медицині поновився. В медицині серцево-судинної системи додатки з використанням штучного інтелекту знайшли застосування для розробки нових ліків, передбачення ризиків хвороб, тощо.

Водночас, розвиток портативних пристроїв, що допомагають слідкувати за різноманітними показниками людського організму, призвів до того, що сучасні пульсометри, оксиметри та навіть одноканальні кардіомонітори можуть бути вбудовані в пристрій формату звичайного годинника. А розвиток Internet of Things (IoT) призвів до того, що пристрої мають стандартизовані протоколи передачі та обробки різноманітних даних. Найчастіше в якості “бази”, з якою можна синхронізувати різноманітні пульсометри та кардіомонітори стали сучасні смартфони. Їх потужності та спеціалізованих процесорних ядер навіть достатньо для повноцінної роботи нейронної мережі.

Таким чином, рівень технологій на сьогодні дозволяє розробляти системи, що збирають показники роботи серцево-судинної системи, пересилають їх до смартфону, на якому дані будуть проаналізовані в реальному часі. На основі цього аналізу можуть бути надані не тільки рекомендації щодо фізичної активності та споживання їжі, а й також аналізувати аномалії роботи ССЗ та попереджувати користувача про небезпеку виникнення тієї або іншої хвороби або стану.

1 АКТУАЛЬНІСТЬ ВИКОРИСТАННЯ АВТОМАТИЗОВАННОГО ДЕТЕКТУВАННЯ ПРОБЛЕМ ЗІ ЗДОРОВ'ЯМ ЛЮДИНИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Серцево-судинні захворювання в житті людини

Згідно з даними Всесвітньої організації охорони здоров'я, серцево-судинні захворювання є одним з найбільших факторів смертності в світі. Згідно з даними за 2019 рік, від ССЗ вмерло 17.9 мільйонів людей, що становить 32% від всієї смертності в світі. Серцеві захворювання – це низка станів та відхилень в роботі серцево-судинної системи, що заважає серцю створювати правильне переміщення крові тілом людини. Втім, без слідкування за серцевим ритмом дуже важко виявити ознаки серцевих захворювань. У свою чергу, серцевий ритм не є якоюсь константною величиною та залежить від багатьох факторів: віку, фізичної активності, звичок, тощо. Це робить дані про серцевий ритм непередбачуваними та важко передбачуваними. Але чим скоріше серцеве захворювання буде помічене, тим простіше надати допомогу пацієнту.

Також останнім часом в світі стало більше уваги приділятися власному здоров'ю, моніторингу власного стану, особливо за допомогою «розумних» пристроїв та «Інтернету речей». Розвиток технологій дозволяє робити пристрої компактними, а розвиток методів діагностики захворювань та активне використання так званих технологій «штучного інтелекту» для аналізу вхідних даних дозволяє створювати моніторингові системи, що можуть надавати поради одночасно пацієнту, що знаходиться під моніторингом, та його лікарю задля максимально швидкої реакції на зміни в стані здоров'я. Кількість пристроїв, що зараз використовується в світі, дозволяє збирати великі датасети та аналізувати їх новітніми методами для знаходження закономірностей, що важко помітити

одному конкретному лікарю. Все це підводить медичну сферу до використання систем на базі нейронних мереж та інших AI-схожих технологій.

1.2 Дослідження щодо використання автоматизованих методів в медицині

Одне з перших досліджень щодо ролі комп'ютерних програм/алгоритмів в медицині серцево-судинної системи було зроблене в 1963 році доктором Варнером, та продовжене Горрі та Барнетом у 1968. Вони вивчали роль математичних програм для діагностики вроджених вад серця. Варнер описав математичну модель клінічної діагностики цієї хвороби, що побудована на базі теореми ймовірності Байєса. Використовуючи цей підхід, вроджена вада серця може бути виявлена з точністю не менше людської, а також може бути покращена за допомогою матриці даних про хворобу: симптоми та характерні фізичні ознаки. Втім, використання додаткових систем було обмежене.

Бернер та його команда вивчали діагностичні можливості 4 внутрішніх медичних систем діагностики, що називалися Dxplain, Iliad, Meditel та QMR відповідно. Вони припустили, що ці системи повинні використовуватися терапевтами для допомоги в діагностиці хвороб. Це припущення було викликано занепокоєністю через те, що інколи важливі діагнози можуть бути незрозумілими під час діагностики та витрачається дуже багато часу на саму діагностику, особливо коли лікар не дуже досвідчений. Крім допомоги з діагностикою перші системи з використанням «штучного інтелекту» допомагали з інтерпретацією результатів аналізів. Однією з перших таких систем був PUFF, що був розроблений дослідниками зі Стенфордського університету та Тихоокеанського Пресвітеріанського медичного центру в Сан-Франциско, для обробки пульмонологічних функціональних тестів пацієнтів з хворобами легенів. Ця система використовувала комп'ютерні алгоритми для

виявлення наявності та тяжкості хвороби легенів та для формування автоматичних звітів на основі електронних даних пацієнтів.

Використання комп'ютерних систем в рішеннях щодо діагнозів було обмежене декількома факторами. По-перше, більшість з них була побудована по принципу «якщо сталося це – робіть те», а потім використання математики для обчислення ймовірності різніх результатів. Але в реальному світі клінічні проблеми комплексні, мають багато факторів, випадкові та втім пов'язані. Наприклад, висока температура та сильний біль в животі у 40-річного афроамериканця та 25-річної жінки з Європи можуть мати дуже різні причини. Інші фактори, що впливають на складність діагностування:

– соціально-економічні;
– попередні хірургічні втручання;
– чи подорожувала людина;
– ліки, що приймаються;
– персональні звички;
– тощо.

Таким чином, в медичній сфері є потреба в алгоритмах та системах, що не використовують жорстку алгоритмічну логіку при діагностиці, можуть робити аналіз великої кількості даних, знаходити нові закономірності того чи іншого стану або симптому та надавати швидкий результат як пацієнту, так і лікарям. Для всього цього більш за все підходять системи на базі машинного навчання та нейронних мереж.

1.3 Пристрої для слідкування за серцевим ритмом

Сучасні портативні пристрої з категорії «розумні речі», що можуть збирати дані про серцевий ритм, поділяються на декілька категорій:

– пульсометри – пристрої, що мають тільки функцію вимірювання пульсу;

– пульсоксиметри – вимірюють не тільки пульс, а й насиченість крові киснем;

– електрокардіомонітори – фактично зменшений варіант медичного ЕКГ апарату, інколи навіть з декількома каналами даних. Втім, найчастіше зустрічаються портативні одноканальні кардіомонітори;

– кардіоодежа – спеціальний одяг, що має велику кількість вбудованих сенсорів, які дозволяють слідкувати за проходженням крові по багатьох частинах організму та знаходити проблеми більш локалізовано;

– «розумні» годинники – наручні пристрої, найчастіше під'єднані до мобільного телефону, можуть об'єднувати в собі декілька пристроїв одночасно.

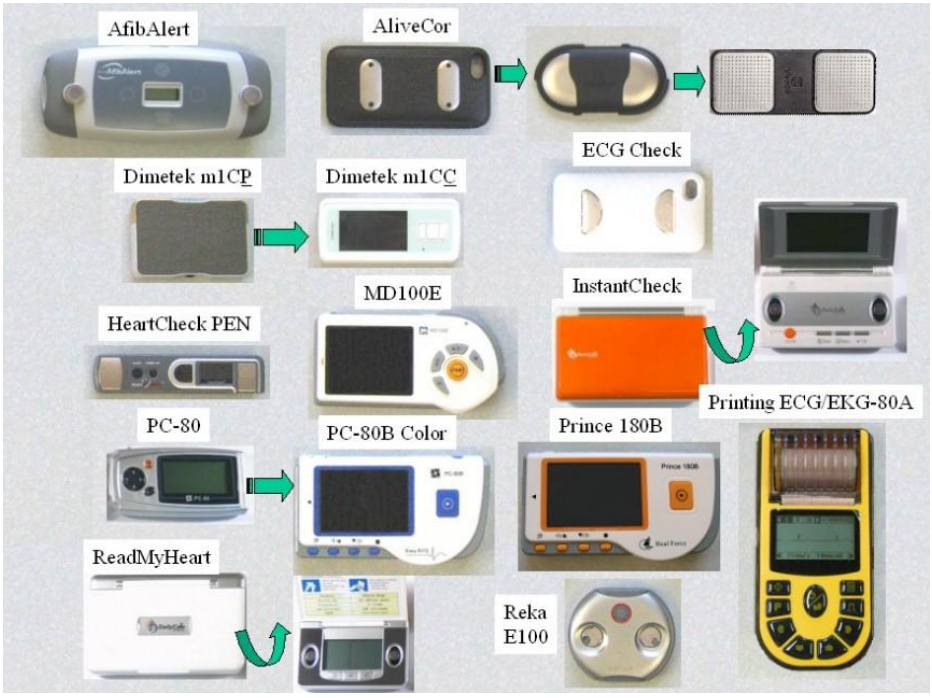

Рисунок 1.1 – Сучасні портативні пристрої для реєстрації ЕКГ-сигналів

Всі ці пристрої дозволяють збирати дані про серцевий ритм людини. Щодо збереження даних вони поділяються на наступні категорії:

– пристрої, що тільки збирають дані та одразу передають їх на парний пристрій (ПК, телефон, тощо);

– пристрої, що мають власну пам’ять та можуть зберігати деяку кількість даних локально.

Пристрої можуть бути спеціалізовані для використання суто в медичній сфері чи для більш широкого використання. Кожна з категорій має свої переваги та недоліки. Переваги спеціалізованих систем:

– більш глибока інтеграція з іншими медичними системами, використання стандартизованих протоколів зберігання та обміну даними;

– більша точність вимірювань.

Недоліками є:

– пропрієтарність використаних протоколів;

– «vendor lock», тобто неможливість працювати з системами інших брендів.

В свою чергу загальнодоступні пристрої можуть мати як пропрієтарні, так і відкриті протоколи обміну даними, але найчастіше мають меншу точність вимірювання та відсутність швидкої інтеграції з медичними системами.

1.4 Медичні калькулятори

Медичний калькулятор або клінічний калькулятор являє собою програмне забезпечення для розрахунку різних клінічних показників і індексів, таких як індекс маси тіла (IMT), площа поверхні тіла (BSA), коронарний ризик серцево-судинних захворювань, індивідуальна доза препарату і так далі. Зазвичай розрахунок клінічних показників або індексів включає в себе складні формули з використанням декількох вхідних параметрів. Медичні калькулятори, як

правило, надають користувальницький інтерфейс для введення параметрів і обчислення оцінки, використовуючи стандартну формулу. Користувачам не потрібно використовувати або навіть знати фактичну формулу для розрахунку клінічної оцінки або індексу. Наприклад, індекс маси тіла або ІМТ є найбільш часто використовуваним показником ожиріння на міжнародному рівні [19].

Спочатку комп'ютерні програми медичного калькулятора були доступні на персональних комп'ютерах. Пізніше, онлайн-версії деяких калькуляторів були доступні через Інтернет. Проте, лікарі часто не в змозі використати це програмне забезпечення на момент виїзду із-за відсутності доступу до комп'ютера. Тепер медичні калькулятори доступні для декількох платформ для смартфонів. Найбільш часто використовувані медичні калькулятори MedMath і MedCalc. Розрахунок доз препаратів для педіатричних пацієнтів дуже важлива під час невідкладної медичної допомоги, а також застосування «Paeds ED» який розраховує дози препарату для дітей в залежності від їх віку. Дози препарату, для пацієнтів з нирковою недостатністю можуть бути обчислені за допомогою Softforce. Додаток uBurn допомагає хірургам обчислити відсоток від загальної площі поверхні тіла постраждалих при опіках [20].

1.5 Аналіз проблеми та постановка задачі

Якщо подивитися на сучасну діагностику серцево-судинних захворювань, то можна побачити, що коректність діагнозу дуже залежить від таких факторів, як:

– регулярність візитів людини до лікарні для спостереження;

– досвідченість лікарів, що спостерігають за людиною (в тому числі операторів ЕКГ, лаборантів, тощо);

– повсякденний стиль життя людини;

– звички;

– стреси;

– хронічні захворювання інших систем організму;

– вік людини.

Також є багато інших факторів. Це робить діагностику непередбачуваною та відкритою до інтепрпретації, що може призвести до виявлення захворювання не на початковій стадії розвитку та ускладнити лікування.

Покращити діагностування захворювань можливо за допомогою постійного спостереження за показниками організму з подальшою передачею даних до медичного закладу для аналізу. Це дозволить виявляти аномалії роботи наприклад серцево-судинної системи швидше та попереджати як саму людину, так і її лікаря про необхідність додаткових досліджень, спостережень, тощо.

Тренд використання фітнес-браслетів з функцією слідкування за серцевим ритмом та швидкий розвиток ідеї Internet of Things, а також розвиток методів аналізу великих об’ємів даних є перспективним напрямком для покращення діагностики захворювань.

Нерівномірність та варіабельність серцевих ритмів від одної людини до іншої вимагає використання систем, що можуть виконувати аналіз даних та навчатися на нових вхідних послідовностях без використання жорстко прописаних алгоритмів. Такими системами є нейронні мережі.

Таким чином, ціллю цієї роботи є розробка автоматизованого методу діагностування серцево-судинних захворювань за допомогою нейронних мереж.

2 МАШИННЕ НАВЧАННЯ ТА НЕЙРОННІ МЕРЕЖІ

2.1 Машинне навчання

Розвиток комп'ютерних технологій, особливо швидкості обробки даних, суперком'ютери та новітні розробки в сфері ШІ все більше знаходять використання в медичній сфері. Термін «машинне навчання» був вперше використаний американським піонером в сфері комп'ютерних ігор та штучного інтелекту Л. А. Семьюелом в 1959 році. Машинне навчання – це підхід до вирішення задач за допомогою програм/систем, що «навчаються» на вхідному наборі даних, але не є жорстко запрограмованими на будь-який алгоритм.

Машинне навчання засноване на алгоритмах. Алгоритми можна розділити на три основні категорії.

Контрольовані алгоритми являють собою набір навчальних даних, що має вхідні дані, а також бажаний результат. Під час навчання модель буде корегувати свої змінні для зіставлення вхідних даних з відповідним виходом.

Неконтрольовані алгоритми: в цій категорії немає бажаного результату. Алгоритми групуватимуть набір даних на різні групи.

Алгоритми з підкріпленням: ці алгоритми навчаються на готових рішеннях. На основі цих рішень алгоритм буде навчатися на основі успіху / помилки результату. В кінцевому рахунку алгоритм зможе давати хороші прогнози. На рисунку 2.1 приведені категорії машинного навчання.

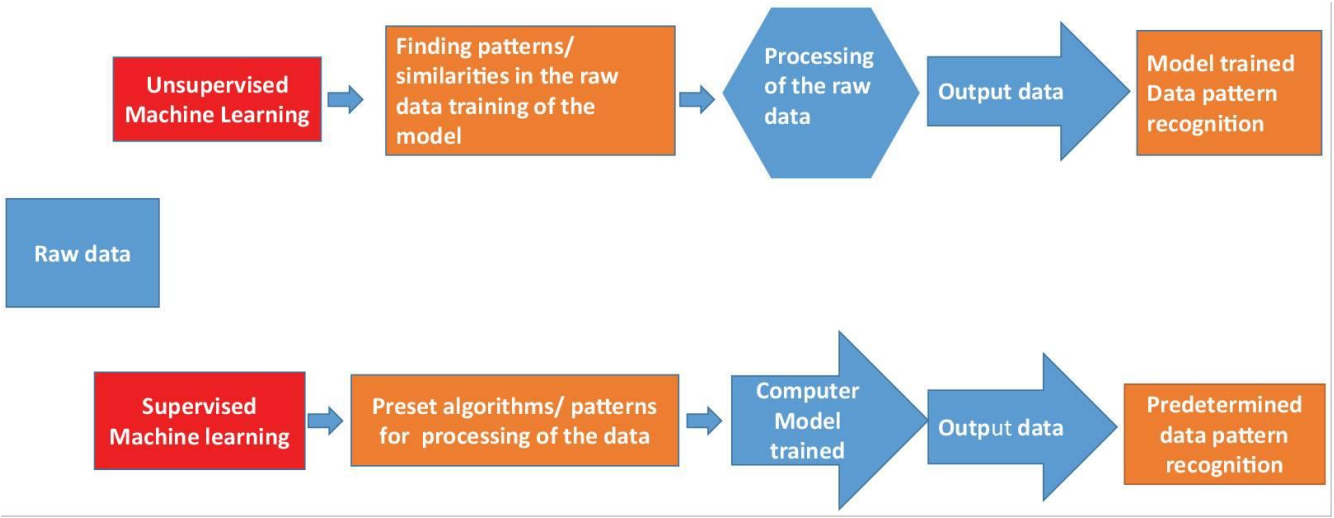

Рисунок 2.1 – Категорії машинного навчання.

Так, машинне навчання може бути з зовнішнім наглядом та без нього. При навчанні з зовнішнім наглядом основна ціль – співвіднести вхідний набір даних з вихідним. Такий тип навчання найчастіше використовується в системах машинного зору, розпізнавання рукописного тексту, інтерпретації електрокардіограм та рентгенівських зображень. Під час тренування системи з зовнішнім наглядом користувач бере базу вхідних спостережень та їх наслідків і використовує ці дані для формування моделі прогнозів, щоб класифікувати наслідки для набору спостережень. Найчастіше використовувані математичні алгоритми в цій сфері включають в себе статистичні методи такі як лінійна регресія, логістична регресія, аналіз виживання та дерева рішень.

Навчання без зовнішнього нагляду – тип машинного навчання, коли ціллю є розпізнання відношень всередині самих даних. Прикладами навчання без нагляду є кластеризація, аналіз принципових компонент та самоорганізовані карти.

Глибоке навчання – відносно новий принцип використання штучного інтелекту, що також називають нейронними мережами. Ідея цього методу – обробка інформації способом, що схожий на обробку даних в мозку людини. Ця

концепція обробки інформації була вперше запропонована двома вченими з университету Чикаго в 1944 році, Волтером Піттсом та Ворреном МакКало. Їх стаття «A Logical Calculus of the Ideas Immanent in Nervous Activity» була опублікована в періодичному виданні Математична біофізика та пояснювала, що мозок працює як «процесор інформації», а нейрони мозку можуть видавати комплексні паттерни з необхідною інформацією за допомогою з'єднання один з одним та використовуючи логічні концепції «І», «АЛЕ», «НЕ». Вчені прийшли до висновку, що нейрони є логічними елементами, що можуть обробляти декілька вхідних потоків даних та видавати єдиний вихідний потік на основі них.

2.2 Нейронні мережі

Нейронні мережі увійшли в практику всюди, де потрібно вирішувати завдання прогнозування, класифікації або управління. Такий вражаючий успіх визначається декількома наступними причинами.

Багаті можливості. Нейронні мережі − виключно потужний метод моделювання, що дозволяє відтворювати надзвичайно складні залежності. Зокрема, нейронні мережі нелінійні по свій природі. Протягом багатьох років лінійне моделювання було основним методом моделювання в більшості областей, оскільки для нього добре розроблені процедури оптимізації. У завданнях, де лінійна апроксимація незадовільна (а таких досить багато), лінійні моделі працюють погано. Крім того, нейронні мережі справляються з «прокляттям розмірності», яке не дозволяє моделювати лінійні залежності в разі великого числа змінних.

Простота у використанні. Нейронні мережі навчаються на прикладах. Користувач нейронної мережі підбирає представницькі дані, а потім запускає алгоритм навчання, який автоматично сприймає структуру даних. При цьому

від користувача, звичайно, потрібно якийсь набір евристичних знань про те, як слід відбирати і готувати дані, вибирати потрібну архітектуру мережі та інтерпретувати результати, проте рівень знань, необхідний для успішного застосування нейронних мереж, набагато скромніше, ніж, наприклад, при використанні традиційних методів статистики.

Ще один фактор, який робить використання нейронних мереж привабливим, є схожість їх роботи з мозком людини. Нейронні мережі вважаються одним з кроків до повноцінного штучного інтелекту.

Штучна нейронна мережа (ШНМ) складається з об'єднаних «нод», що імітують нейрони в мозку людини таким чином, що мають вхід, процесор для даних та вихід. Будь-яка мережа має в своєму складі:

– вхідний шар нейронів;
– прихований шар;
– вихідний шар.

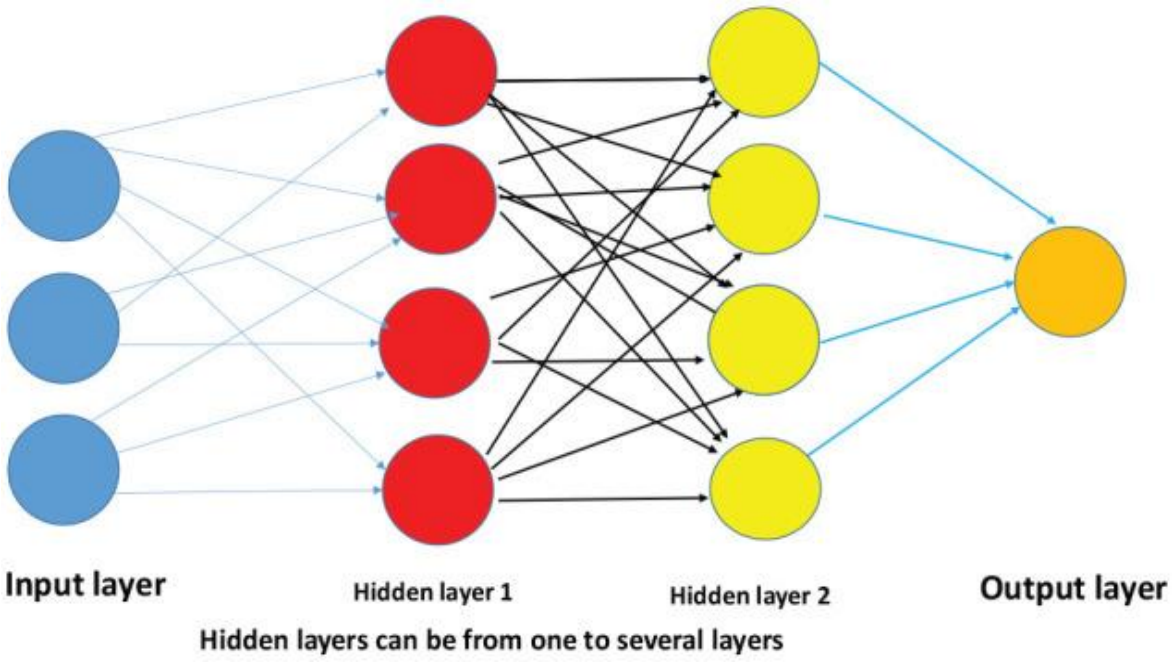

Рисунок 2.2 – Структура нейронної мережі

Вхідний шар виконує функцію сенсорних нейронів в нервовій системі та приносить дані на обробку. Прихований шар виконує основний процес обробки інформації, а вихідний інтерпретує дані та видає їх у фінальній формі.

Згорточна нейронна мережа – один з типів штучної нейронної мережі, що навчається за допомогою таких методів, як зворотне поширення. Іншими ознаками такої мережі є використання спільного доступу до параметрів та об’єднання. Ці ознаки допомагають знизити вимоги до кількості обчислень при роботі системи та покращує обробку зображень, звуків та відеозаписів.

2.3 Архітектури нейронних мереж

2.4.1 Багатошаровий перцептрон

Багатошаровий перцептрон складається з 3 або більше шарів. Він використовує нелінійну функцію активації, часто тангенціальну або логістичну, яка дозволяє класифікувати лінійно нероздільні дані. Кожен вузол в шарі з'єднаний з кожен вузлом в наступному шарі, що робить мережу повністю пов'язаної. Така архітектура знаходить застосування в задачах розпізнавання мови і машинному перекладі. Архітектура багатошарового перцептрона наведена на рисунку 2.3.

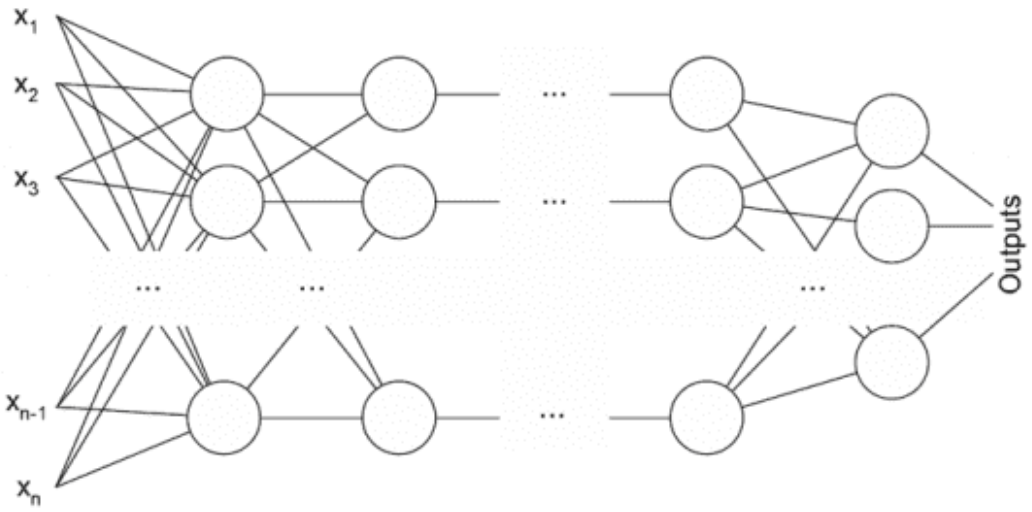

Рисунок 2.3 – Багатошаровий перцептрон

2.4.2 Згорткова нейронна мережа

Згорткова нейронна мережа (Convolutional neural network, CNN) містить один або більше об'єднаних або поєднаних згортальних шарів. CNN використовує варіацію багатошарового перцептрона, розглянутого вище. Згорткові шари використовують операцію згортки для вхідних даних і передають результат в наступний шар. Ця операція дозволяє мережі бути глибше з меншою кількістю параметрів.

2.4.3 Рекурсивна нейронна мережа

Це тип глибокої нейронної мережі, сформований при застосуванні одних і тих же наборів ваг рекурсивно над структурою, щоб зробити скалярне або структуроване пророкування над вхідною структурою змінного розміру через активацію структури в топологічному порядку. У простій архітектурі нелінійність, така як тангенціальна функція активації, і матриця ваг, колективна всією мережею, використовуються для об'єднання вузлів в батьківські об'єкти.

2.4.4 Рекурентна нейронна мережа

Рекурентна нейронна мережа, на відміну від прямої нейронної мережі, є варіантом рекурсивної ІНС, в якій зв'язки між нейронами − спрямовані цикли. Останнє означає, що вихідна інформація залежить не тільки від поточного входу, але також від станів нейрона на попередньому кроці. Така пам'ять дозволяє користувачам вирішувати завдання NLP: розпізнання рукописного тексту або мови.

2.4.5 LSTM

Мережа довгої короткостроковій пам'яті (Long Short-Term Memory, LSTM) − різновид архітектури рекуррентной нейромережі, створена для більш точного моделювання часових послідовностей і їх довгострокових залежностей, ніж традиційна рекуррентная мережу. LSTM-мережа не використовує функцію активації в рекурентних компонентах, збережені значення не модифікуються, а градієнт не прагне зникнути під час тренування. Часто LSTM застосовується в блоках по кілька елементів. Ці блоки складаються з 3 або 4 затворів (наприклад, вхідного, вихідного і гейта забування), які контролюють побудову інформаційного потоку з логістичної функції.

2.4.6 Sequence-to-sequence модель

Часто Sequence-to-sequence моделі складаються з двох рекурентних мереж: кодувальника, який обробляє вхідні дані, і декодера, який здійснює висновок.Sequence-to-Sequence моделі часто використовуються в питально-відповідних системах, чат-ботах і машинному перекладі.

2.4.7 Поверхневі (shallow) нейронні мережі

Неглибокі моделі, як і глибокі нейронні мережі, теж популярні і корисні інструменти. Наприклад, word2vec − група неглибоких двошарових моделей, яка використовується для створення векторних уявлень слів (word embeddings). Word2vec приймає на вході великий корпус тексту і створює векторний простір. Кожному слову в цьому корпусі приписується відповідний вектор в цьому просторі. Відмітна властивість − слова із загальних текстів в корпусі розташовані близько один до одного в векторному просторі.

3 МЕТОДИ ПРИЙНЯТТЯ РІШЕНЬ

Машинне навчання можуть використовувати різні математичні алгоритми для класифікації об’єктів та прийняття рішень. В залежності від задачі та вхідних даних, різні алгоритми можуть давати різну точність рішень та можуть потребувати різну кількість обчислювальних ресурсів для роботи. Нижче описані деякі з найпоширеніших алгоритмів прийняття рішень, що використовуються в сучасному аналізі даних.

3.1 Лінійна регресія (linear regression)

Алгоритм лінійної регресії використовує точки даних для пошуку оптимальної лінії для створення моделі. Лінію можна представити рівнянням y = m * x + c, де y − залежна змінна, а x − незалежна змінна. Базові теорії обчислення застосовуються для визначення значень для m і c з використанням заданого набору даних.

Існує два типи лінійної регресії: проста лінійна регресія із однієї незалежною змінною і множинна лінійна регресія, де використовується кілька незалежних змінних.

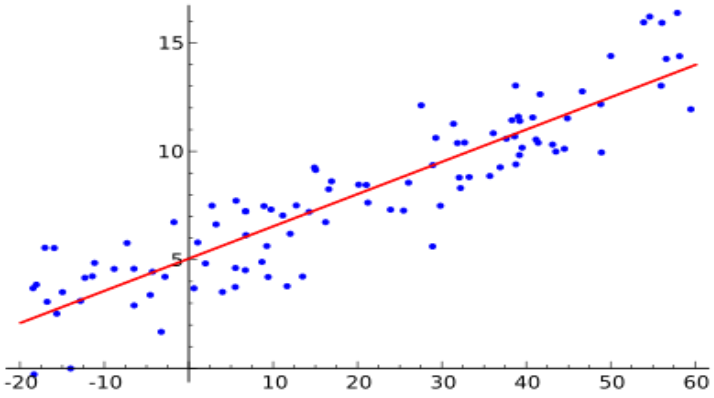

Рисунок 3.1 – Графік лінійної регресії

3.2 Метод опорних векторів (SVM − Support Vector Machine)

Метод опорних векторів належить до алгоритму класифікаційного типу. Алгоритм буде розділяти точки даних, використовуючи лінію. Ця лінія повинна бути максимально віддаленої від найближчих точок даних в кожній з двох категорій.

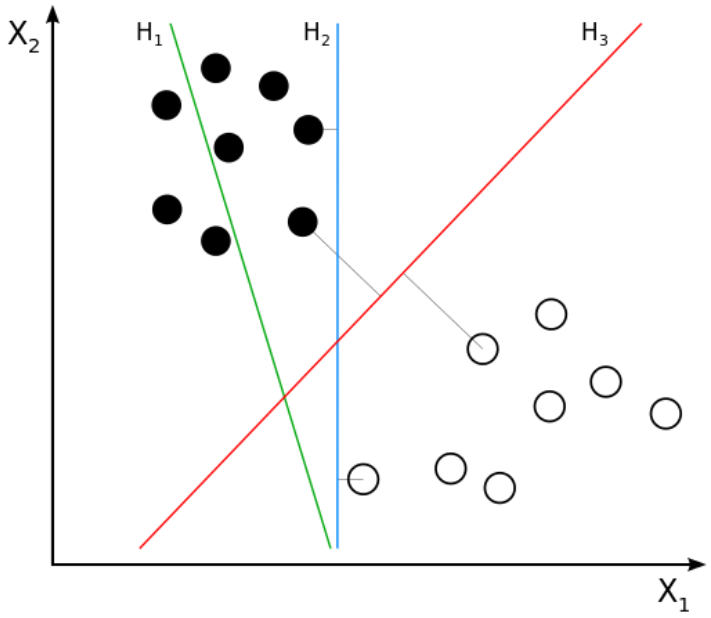

Рисунок 3.2 – Графік алгоритму опорних векторів

На рис.3.2 червона лінія підходить найкраще, так як вона найбільше віддалена від усіх точок. На основі цієї лінії дані діляться на дві групи.

3.3 Метод k-найближчих сусідів (KNN − K-nearest neighbors)

Метод k-найближчих сусідів є простим алгоритм, який передбачає невідому точку даних на основі її k найближчих сусідів. Значення k тут критично важливий фактор, який визначає точність передбачення. Найближча точка визначається виходячи з базових функцій відстані, на кшталт Евклідовій.

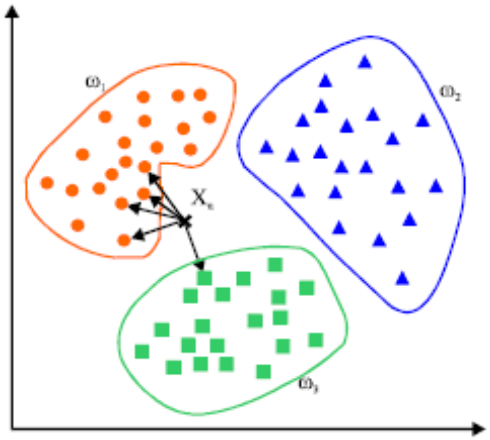

Рисунок 3.3 – Зображення методу k-найближчих сусідів

Однак цей алгоритм вимагає високої обчислювальної потужності, і необхідно спочатку нормалізувати дані, щоб кожна точка даних була в тому ж діапазоні.

3.4 Логістична регресія (logistic regression)

Логістична регресія використовується, коли очікується дискретний результат, наприклад, виникнення якої-небудь події, наприклад, піде дощ чи ні. Зазвичай логістична регресія використовує функцію, щоб помістити значення в певний діапазон.

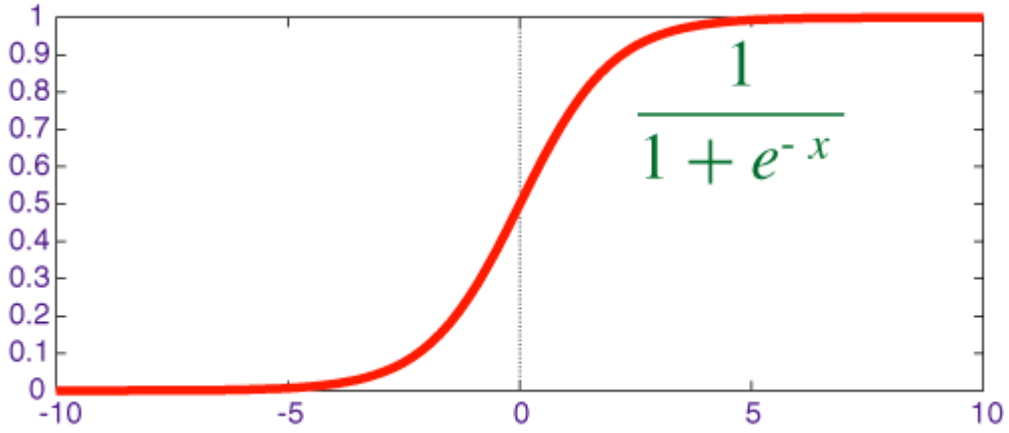

Рисунок 3.4 – Графік логістичної регресії

«Sigmoid» − це одна з таких функцій у формі літери S, яка використовується для бінарної класифікації. Вона конвертує значення в діапазон від 0 до 1, що є ймовірністю виникнення події. Просте рівняння логістичної регресії має вид:

$$y = e \wedge (b0 + b1 * x) / (1 + e \wedge (b0 + b1 * x)), \quad (1)$$

де b0 і b1 − це постійні змінні. Під час навчання значення для них будуть обчислюватися таким чином, щоб помилка між прогнозом і фактичним значенням ставала мінімальною.

3.5 Дерево рішень (decision tree)

Дерево рішень є основними одиницями випадкового лісового класифікатора (random forest).

На високому рівні дерева рішень можуть розглядатися як конструкція машинного навчання, яка використовується для класифікації або регресії деяких даних в ієрархічній структурі.

Дерева рішень використовують машинне навчання для виявлення основних диференціюючих факторів між різними класами наших даних. Роблячи це, дерева рішень можуть приймати деякі вхідні дані та прогнозувати клас, запускаючи дані за допомогою набору диференціюючих питань, які він формує за допомогою машинного навчання.

Сформульовані питання − це питання «ні», і після кожного запитання є два шляхи − шлях «так» і шлях «ні». Кожен із цих шляхів призведе до наступного питання або до остаточного результату.

Існує деяка ключова термінологія, з якою треба ознайомитися, перш ніж дізнатися про дерева рішень та випадкові класифікатори лісу:

– класи: коли проводиться класифікація, список різних груп, до яких можуть належати дані, називають класами;

– вузли: різні питання, утворені деревом рішень, називаються вузлами. Їх можна розглядати як точки на дереві рішень, де формулюється розкол (або гілки). Кожен із них − це питання «так» або «ні», і як тільки на нього відповімо, переходимо на крок ближче до визначення класу, до якого належать дані;

– листя: через наявність вузлів існує багато альтернативних шляхів, які створюються в будь-якому дереві рішень, навіть з кількома шарами вузлів. Кінцеві точки кожного з цих шляхів відомі як «лист». Ці листи представляють кінцеве значення або клас, який буде прогнозовано для заданих вхідних даних.

На рисунку 3.5 зображено сильно спрощений вигляд того, як може виглядати частина дерева рішень, яка використовується для класифікації тварини як птаха, собаки чи риби. На цій діаграмі білі поля − це вузли, а зелені − це листя.

Алгоритми машинного навчання використовуватимуться для аналізу трьох різних класів та для формулювання питань, необхідних для розмежування різних класів, у цьому випадку: «Чи плаває?» і «У нього чотири ноги?».

Після відповіді на ці запитання «так» або «ні» можна взнати, до якого класу належатимуть дані, тобто до якої тварини.

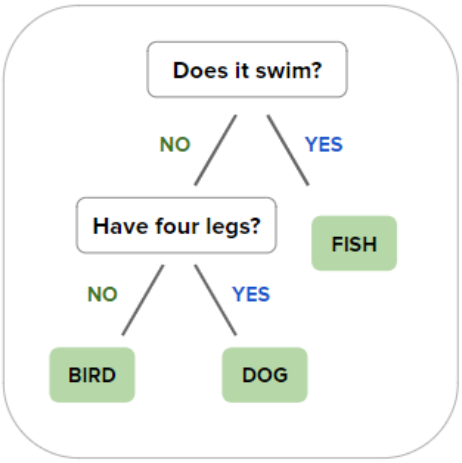

Рисунок 3.5 – Частина дерева рішень

Отже, дерева рішень − це досить проста концепція, яку можна зрозуміти, оскільки вона є надзвичайно інтуїтивно зрозуміла.

3.6 Випадковий ліс (Random Forest)

Простий спосіб осягнути цю модель − інтерпретувати її назву в дуже буквальному сенсі. Ліс можна розглядати як колекцію дерев. Це говорить про те, що модель Random Forest буде складатися з різних дерев рішень. Причина, яку вони називають «Випадковою», полягає в тому, що кожне дерево рішень у лісі навчається за допомогою випадкового підмножини даних про навчання. Ця методика навчання кожного дерева в лісі з використанням іншої, випадкової вибірки даних відома як Bagging (також відома як агрегація Bootstrap). У типових моделях випадкових лісів випадкова вибіркова підмножина даних, щодо яких слід здійснити класифікацію, передається через кожне дерево рішень у лісі з метою підготовки цього дерева. Пізніше, кожне дерево, що становить ліс, дає передбачуваний клас (мітку). Остаточне передбачення визначається шляхом вимірювання того, яке прогнозування було зроблено найбільшою кількістю дерев у лісі.

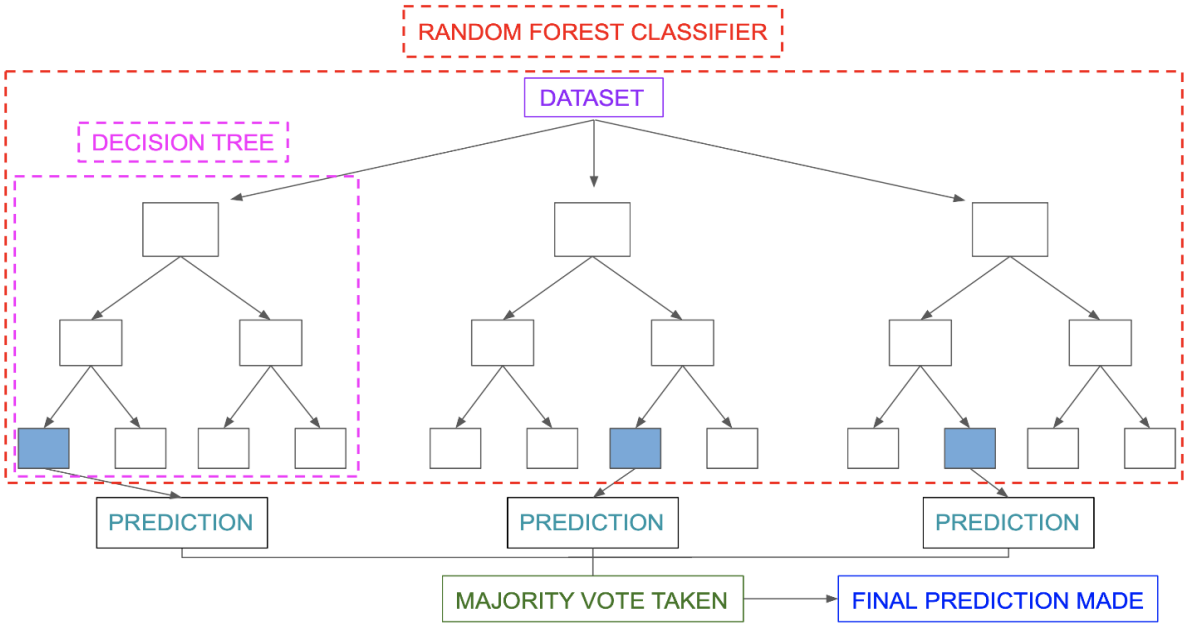

Рисунок 3.6 – Спрощений вигляд випадкового класифікатора лісу

Однією з головних причин випадкові класифікатори лісу є настільки ефективними, тому що кожне дерево в лісі значною мірою не пов'язане з іншими деревами в лісі, це умова, що досягається завдяки випадковому відбору даних для навчання кожного окремого дерева (пакетування). Як результат, деякі дерева можуть дати неправильні прогнози для остаточного класу випуску, але оскільки береться мітка, що передбачає найбільшою кількістю дерев, можна отримати досить високу точність, оскільки більшість дерев, ймовірно, дають правдиве передбачення. Таким чином, використання низки непов'язаних дерев рішень допомагає зменшити відсоток помилок, оскільки помилки, зроблені одним деревом, зазвичай не роблять для інших дерев. Таким чином, точність збільшується на значну кількість.

Такий стан, в основному, незалежних дерев рішень, який досягається завдяки випадковій вибірці даних, є тим, що допомагає гарантувати, що модель в цілому продовжує рухатися в правильному напрямку та підтримує досить високу точність.

Випадкові лісові класифікатори виступають як ансамбль. Оскільки всі дерева рішень, які складають «випадковий ліс», значною мірою не залежать одне від одного через випадкову вибірку точок даних (перенесення), то це забезпечує ті прогнози, які робляться на основі більшості голосів, отриманих від дерев, мають більш високу точність, ніж один прогноз, отриманий з будь-якого одного дерева. Це також одна з основних причин того, що випадкові лісові класифікатори мають набагато більшу точність, ніж інші класифікаційні моделі, що працюють на тому ж наборі даних.

3.7 Наївний байесовський класифікатор (Naive Bayes)

Цей алгоритм заснований на теоремі Байеса. Завдяки цьому наївний байесовский класифікатор можна застосувати, тільки якщо функції незалежні один від одного. Якщо спробувати передбачити вид квітки на основі довжини і ширини його пелюстки, зможемо використовувати цей алгоритм, так як функції не залежать один від одного.

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Наївний байесовский алгоритм також є класифікаційним. Він використовується, коли проблема містить кілька класів.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Огляд програмних компонентів розробки

Реалізація нейронної мережі була виконана за допомогою мови програмування Python та спеціалізованих бібліотек для аналізу даних, таких як NumPy, Pandas. В якості фреймворку нейронної мережі було використано scikit-learn. Для візуалізації результатів було використано фреймворк matplotlib, що дозволяє генерувати графіки на основі вхідних масивів даних та має інтеграцію з NumPy та Pandas.

Python – мова програмування високого рівня, що дозволяє виконувати програми без їх компіляції, тобто методом інтерпретації. Мова використовує динамічну строгу типізацію даних, має легкий синтаксис, багату стандартну бібліотеку, що дозволяє розробнику ПЗ вирішувати широкий клас задач. Окрім стандартної бібліотеки, Python має репозиторій зовнішніх бібліотек, що розробляються та підтримуються суспільством розробників. Ця мова є одною з найпопулярніших в сучасній веб-розробці, розробці системних утиліт та активно використовується для аналізу даних. За допомогою CFFI (C Foreign Function Interface) Python дозволяє створення бібліотек на мовах C/C++, які надають інтерфейс для використання поряд зі стандартним кодом. Це дозволяє проводити оптимізації та прискорювати обчислення великих масивів даних порівняно з реалізацією на «чистому» Python в декілька разів. Простий синтаксис та велика кількість написаних для мови Python бібліотек стала причиною його популярності в науковому середовищі, що підштовхнуло розробників до створення декількох потужних фреймворків для роботи з масивами даних та створення нейронних мереж.

У роботі використовувались наступні бібліотеки.

scikit-learn – це Python-бібліотека, що дозволяє створювати нейронні мережі в декілька строк коду. В ній зібрані реалізації алгоритмів для лінійної та нелінійної регресії, кластеризації даних та багато іншого. Також бібліотека має готові реалізації моделей машинного навчання під наглядом, без нагляду та глибокого навчання. Водночас scikit-learn не надає інструментів для роботи з нейронними мережами, тому була використана додаткова бібліотека keras.

Keras – це бібліотека для реалізації задач глибокого машинного навчання та роботи з нейронними мережами. Вона дозволяє швидко реалізовувати моделі, тренувати їх на підготованих наборах даних, оцінювати точність, коригувати гіперпараметри мережі та багато іншого. Keras також має в своєму складі специфічні оптимізації під широкий спектр процесорів та відеокарт, що дозволяє прискорювати виконання навчання нейронних мереж в декілька разів.

4.1 Підготовка тестового набору даних

Найпоширенішим способом роботи з даними для будь-якого аналізу є використання об’єкту Dataframe із бібліотеки pandas. Цей об’єкт поводиться як таблиця, дозволяє зберігати в колонках довільні типи даних та трансформувати дані таким чином, що найбільше підходить під конкретні задачі. Реалізації машинного навчання, що написані на Python, також вміють працювати з об’єктом Dataframe та часто використовують його як вхідний набір.

Для формування тестового набору даних для різних моделей аналізу спочатку імпортуємо базовий код та набір даних для аналізу, як на лістингу 5.1.

Лістинг 4.1 – Підключення необхідних бібліотек

```
import numpy as np
```

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

%matplotlib inline

import os
print(os.listdir())

import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split

dataset = pd.read_csv("heart.csv")
```

Таким чином, було отримано об’єкт Pandas Dataframe, який може бути використаний в якості вихідної сутності для подальшого аналізу. Приклад використаних даних наведений на рисунку 5.1.

```
In [6]: dataset.sample(5)
```

Out[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
131	49	0	1	134	271	0	1	162	0	0.0	1	0	2	1
256	58	1	0	128	259	0	0	130	1	3.0	1	2	3	0
123	54	0	2	108	267	0	0	167	0	0.0	2	0	2	1
262	53	1	0	123	282	0	1	95	1	2.0	1	2	3	0
210	57	1	2	128	229	0	0	150	0	0.4	1	1	3	0

Рисунок 4.1 – Приклад даних, використаних в дослідженні

Тепер сформуємо набори для тренування та перевірки коректності аналізу, як показано на лістингу 5.2

Лістинг 4.2 – Формування тренувальних та тестових наборів даних

```
predictors = dataset.drop("target",axis=1)
target = dataset["target"]

X_train,X_test,Y_train,Y_test = train_test_split(predictors,target,test_size=0.20,random_state=0)
```

4.2 Алгоритм аналізу точності роботи моделей

Фреймворк scikit-learn має вбудовану функцію перевірки точності обраних методів прийняття рішень, що називається `accuracy_score`. Завдяки цьому формула аналізу точності для всіх моделей однакова:

```
from sklearn.metrics import accuracy_score
score = round(accuracy_score(Y_pred_meth, Y_test)*100,2)
```

Функція `accuracy_score` порівнює значення, що отримані при роботі моделі, зі значеннями, що отримані при формуванні тестових та тренувальних наборів.

4.3 Аналіз точності моделей обробки даних

Методи, що були перевірені:

– логістична регресія;

– наївний байєсовский класифікатор;
– метод опорних векторів;
– метод к-найближчих сусідів;
– метод дерева рішень;
– метод рандомного лісу;
– нейронна мережа.

Кожен з методів було протестовано наступним чином:
– імпорт та ініціалізація реалізації методу з бібліотеки;
– навчання на тренувальних наборах даних;
– обробка тестового набору даних для детектування аномалій;
– перевірка точності детектування за допомогою еталонних даних.

Результати аналізу кожної з моделей приведені на рисунках 4.2 - 4.8.

Реалізація логістичної регресії в scikit-learn не потребує додаткового налаштування, тому її тестування має один з найпростіших виглядів:

```
In [44]: from sklearn.linear_model import LogisticRegression

         lr = LogisticRegression()

         lr.fit(X_train,Y_train)

         Y_pred_lr = lr.predict(X_test)

In [45]: Y_pred_lr.shape

Out[45]: (61,)

In [96]: score_lr = round(accuracy_score(Y_pred_lr,Y_test)*100,2)

         print("Точність роботи при використанні методу логістичної регресії: "+str(score_lr)+" %")

         Точність роботи при використанні методу логістичної регресії: 85.25 %
```

Рисунок 4.2 – Точність роботи логістичної регресії

Модель логістичної регресії не підтримує навчання на нових даних після виконання навчання на тренувальному наборі, тобто неможлива реалізація

постійного перенавчання та адаптації. Замість цього потрібно згенерувати новий тренувальний набір та перевчати модель з нуля.

Для тестування наївного байєсовського класифікатора було обрано реалізацію з використанням розподілення Гауса. Ця реалізація має хорошу продуктивність, проста в налаштуванні, але швидко деградує при збільшенні тренувального набору даних. Також ця реалізація підтримує online-оновлення ядра рішень, тобто може постійно навчатися на нових вхідних даних та бути більш адаптивною.

Результат оцінки точності методу можна побачити на рисунку 4.3.

```
In [47]: from sklearn.naive_bayes import GaussianNB

         nb = GaussianNB()

         nb.fit(X_train,Y_train)

         Y_pred_nb = nb.predict(X_test)

In [48]: Y_pred_nb.shape

Out[48]: (61,)

In [97]: score_nb = round(accuracy_score(Y_pred_nb,Y_test)*100,2)

         print("Точність роботи при використанні наївного байєсовського класифікатора: "+str(score_nb)+" %")

         Точність роботи при використанні наївного байєсовського класифікатора: 85.25 %
```

Рисунок 4.3 – Точність роботи наївного байєсовського класифікатора

Логістична регресія та наївний байєсовський класифікатор мають однакову точність детектування проблем, але останній потребує менше часу на оцінку. Час виконання детектування за допомогою наївного байєсовського класифікатора можна подивитися в таблиці 5.2.

Тестування методу опорних векторів проводилося з використанням linear kernel, що більше підходить для обраних тренувального та тестового набору даних. Ця модель, як і логістична регресія, не підтримує навчання на льоту, тому для оновлення ядра прийняття рішень потрібно перетренування з нуля.

Результат оцінки точності методу опорних векторів можна побачити на рисунку 4.4.

```
In [50]: from sklearn import svm

         sv = svm.SVC(kernel='linear')

         sv.fit(X_train, Y_train)

         Y_pred_svm = sv.predict(X_test)

In [51]: Y_pred_svm.shape

Out[51]: (61,)

In [98]: score_svm = round(accuracy_score(Y_pred_svm,Y_test)*100,2)

         print("Точність роботи при використанні методу опорних векторів: "+str(score_svm)+" %")

         Точність роботи при використанні методу опорних векторів: 81.97 %
```

Рисунок 4.4 – Точність роботи методу опорних векторів

При тестуванні методу к-найближчих сусідів одним з головних параметрів є число сусідів, що потрібно шукати для кожної точки. Експериментальним шляхом було встановлено, що найкращу точність модель має при кількості сусідів від 7 до 9: менша кількість погіршує точність нижче 50%, більша кількість збільшує час тренування без надання суттєвих переваг по точності.

Метод к-найближчих сусідів також не підтримує навчання на льоту.

Результат точності методу можна побачити на рисунку 4.5.

```
In [53]: from sklearn.neighbors import KNeighborsClassifier

         knn = KNeighborsClassifier(n_neighbors=7)
         knn.fit(X_train,Y_train)
         Y_pred_knn=knn.predict(X_test)

In [54]: Y_pred_knn.shape

Out[54]: (61,)

In [99]: score_knn = round(accuracy_score(Y_pred_knn,Y_test)*100,2)

         print("Точність роботи при використанні методу К найближчих сусідів: "+str(score_knn)+" %")

         Точність роботи при використанні методу К найближчих сусідів: 67.21 %
```

Рисунок 4.5 – Точність роботи методу К найближчих сусідів

Для оцінки точності методу дерева рішень потрібно спочатку обчислити параметр випадковості класифікатора. Це потрібно для отримання максимальної точності класифікації. Методика обчислення параметру випадковості наступна:

– беремо 200 ітерацій на прийняття рішення;

– на кожній ітерації виконуємо навчання моделі та оцінюємо точність;

– в якості параметру випадковості використовується індекс поточної ітерації;

– якщо поточна точність більше максимальної, то запам’ятовуємо індекс ітерації.

Після находження параметру випадковості виконуємо фінальне тренування та оцінку моделі.

Оцінку точності методу дерева рішень можна побачити на рисунку 4.6.

```
In [56]: from sklearn.tree import DecisionTreeClassifier

         max_accuracy = 0


         for x in range(200):
             dt = DecisionTreeClassifier(random_state=x)
             dt.fit(X_train,Y_train)
             Y_pred_dt = dt.predict(X_test)
             current_accuracy = round(accuracy_score(Y_pred_dt,Y_test)*100,2)
             if(current_accuracy>max_accuracy):
                 max_accuracy = current_accuracy
                 best_x = x

         #print(max_accuracy)
         #print(best_x)


         dt = DecisionTreeClassifier(random_state=best_x)
         dt.fit(X_train,Y_train)
         Y_pred_dt = dt.predict(X_test)

In [57]: print(Y_pred_dt.shape)

         (61,)

In [100]: score_dt = round(accuracy_score(Y_pred_dt,Y_test)*100,2)

          print("Точність роботи при використанні методу дерева рішень: "+str(score_dt)+" %")

          Точність роботи при використанні методу дерева рішень: 81.97 %
```

Рисунок 4.6 – Точність роботи методу дерева рішень

Метод випадкового лісу для своєї оцінки потребує того ж самого підходу, що метод дерева рішень, тому повторимо алгоритм пошуку параметру випадковості. В процесі пошуку можна помітити, що метод випадкового лісу працює помітно повільніше за метод дерева рішень. Точну оцінку по часу можна подивитися в таблиці 5.2.

Оцінку точності методу дерева випадкового лісу можна побачити на рисунку 4.7.

```
In [59]: from sklearn.ensemble import RandomForestClassifier

         max_accuracy = 0


         for x in range(2000):
             rf = RandomForestClassifier(random_state=x)
             rf.fit(X_train,Y_train)
             Y_pred_rf = rf.predict(X_test)
             current_accuracy = round(accuracy_score(Y_pred_rf,Y_test)*100,2)
             if(current_accuracy>max_accuracy):
                 max_accuracy = current_accuracy
                 best_x = x

         #print(max_accuracy)
         #print(best_x)

         rf = RandomForestClassifier(random_state=best_x)
         rf.fit(X_train,Y_train)
         Y_pred_rf = rf.predict(X_test)

In [60]: Y_pred_rf.shape

Out[60]: (61,)

In [101]: score_rf = round(accuracy_score(Y_pred_rf,Y_test)*100,2)

          print("Точність роботи при використанні методу випадкового лісу: "+str(score_rf)+" %")

          Точність роботи при використанні методу випадкового лісу: 90.16 %
```

Рисунок 4.7 – Точність роботи методу «випадкового лісу»

Результати оцінювання методів прийняття рішень в машинному навчанні показують, що сучасні алгоритми мають можливість надавати точні прогнози та детектувати аномалії. Останній метод, що оцінений в данній роботі – це нейронна мережа.

Реалізацію нейронної мережі взято з бібліотеки keras. Сама бібліотека скомпільована зі специфічними для процесорів Intel оптимізаціями та використанням інструкцій AVX та AVX2. Ці оптимізації дозволяють виконувати тренування нейронної мережі за менший час та опрацьовувати більші тренувальні набори даних.

В якості моделі нейронної мережі було використано модель Sequential. Ця модель підходить до випадків, коли використовується лише один вхідний та один вихідний тензор.

Щоб запобігти перетренуванню моделі, потрібно обрати правильну кількість нейронів у прихованому шарі. Загальне правило створення нейронних

мереж полягає в тому, що для запобігання зайвої тренованості моделі слід обирати від 5 до 10 вхідних нейронів. Втім, експериментальним методом було встановлено, що найкращі результати по точності нейронна мережа отримує при 11 вхідних нейронах, тому в оцінці точності виберемо саме цю кількість нейронів.

Тривалість тренування моделі – 300 епох, після цього модель оцінюється на точність.

Результати оцінювання точності нейронної мережі можна побачити на рисунку 4.8.

```
In [65]: from keras.models import Sequential
         from keras.layers import Dense

In [66]: model = Sequential()
         model.add(Dense(11,activation='relu',input_dim=13))
         model.add(Dense(1,activation='sigmoid'))

         model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])

In [108]: train_history = model.fit(X_train,Y_train,epochs=300)
          8/8 [==============================] - 0s 1ms/step - loss: 0.3729 - accuracy: 0.8182
          Epoch 292/300
          8/8 [==============================] - 0s 1ms/step - loss: 0.3739 - accuracy: 0.8223
          Epoch 293/300
          8/8 [==============================] - 0s 2ms/step - loss: 0.3577 - accuracy: 0.8430
          Epoch 294/300
          8/8 [==============================] - 0s 2ms/step - loss: 0.3647 - accuracy: 0.8471
          Epoch 295/300
          8/8 [==============================] - 0s 776us/step - loss: 0.3739 - accuracy: 0.8223
          Epoch 296/300
          8/8 [==============================] - 0s 2ms/step - loss: 0.3607 - accuracy: 0.8430
          Epoch 297/300
          8/8 [==============================] - 0s 1ms/step - loss: 0.3640 - accuracy: 0.8264
          Epoch 298/300
          8/8 [==============================] - 0s 1ms/step - loss: 0.3766 - accuracy: 0.8099
          Epoch 299/300
          8/8 [==============================] - 0s 1ms/step - loss: 0.3756 - accuracy: 0.8347
          Epoch 300/300
          8/8 [==============================] - 0s 2ms/step - loss: 0.3675 - accuracy: 0.8471

In [109]: Y_pred_nn = model.predict(X_test)
          2/2 [==============================] - 0s 0s/step

In [110]: Y_pred_nn.shape
Out[110]: (61, 1)

In [111]: rounded = [round(x[0]) for x in Y_pred_nn]

          Y_pred_nn = rounded

In [112]: score_nn = round(accuracy_score(Y_pred_nn,Y_test)*100,2)

          print("Точність роботи при використанні нейронної мережі: "+str(score_nn)+" %")
          Точність роботи при використанні нейронної мережі: 83.61 %
```

Рисунок 4.8 – Точність роботи нейронної мережі

5 АНАЛІЗ РЕЗУЛЬТАТІВ

5.1 Загальний аналіз результатів роботи алгоритмів

Як можна побачити вище з результатів роботи алгоритмів прийняття рішень, вони дають різну точність детектування серцево-судинних захворювань. Окрім точності, є факт використання обчислювальних ресурсів. Проаналізуємо ефективність кожного алгоритму. Аналіз будемо робити як по точності, так і по часу роботи алгоритму.

Точність методів наведена в таблиці 5.1.

Таблиця 5.1

Алгоритм	Точність, %
Логістична регресія	85.25
Наївний байєсовський класифікатор	85.25
Метод опорних векторів	81.97
Метод к-найближчих сусідів	67.21
Дерево рішень	81.97
Випадковий ліс	90.16
Нейронна мережа	83.61

Як можна побачити з результатів, найкращий результат дає метод випадкового лісу, а найгірший – метод к-найближчих сусідів. Всі інші методи мають приблизно схожу точність, що коливається від 81.97 відсотків до 90.16 відсотків. Найкращим методом в даному випадку став метод випадкового лісу.

Тепер проаналізуємо, скільки часу потребує кожен з методів для видання результатів. Час роботи кожного з методів наведено в таблиці 5.2.

Таблиця 5.2

Алгоритм	Час роботи, с
Логістична регресія	0.03
Наївний байєсовський класифікатор	0.007
Метод опорних векторів	0.63
Метод к-найближчих сусідів	0.012
Дерево рішень	0.74
Випадковий ліс	249
Нейронна мережа	0.08

Таким чином, можна побачити, що найкращий час показує алгоритм наївного байєсовського класифікатора, а найгірший – метод випадкового лісу.

5.2 Покращення результатів детектування серцево-судинних захворювань

Результати, що отримані в ході дослідження, наглядно демонструють, що кожний з обраних алгоритмів має свої плюси та мінуси. Але, деякі результати можуть бути покращені. Наприклад, для методу к-найближчих сусідів є 2 потенційні оптимізації: додатковий препроцесінг даних, що дасть можливість більш ефективного пошуку, та змінення кількості сусідів.

В дослідженні була використана кількість сусідів, що дорівнює 7, змінення кількості сусідів до 8 дозволяє навіть без додаткової роботи над даними підвищити точність до 68.85%.

Покращити результати для нейронної мережі можна за допомогою підвищення кількості навчальних епох з 300 до 2000.

5.3 Вибір алгоритму автоматичного детектування серцево-судинних захворювань

Остаточний вибір алгоритму, який буде аналізувати дані та шукати ознаки серцево-судинних хвороб у людини дуже залежить від того, в якому середовищі буде працювати алгоритм, чи є обмеження по використанню обчислювальних ресурсів, часу роботи (наприклад, система буде працювати в реальному часі) та багато іншого. Розглянемо декілька випадків:

– мобільні аплікації. В цьому випадку час роботи та використання обчислювальних ресурсів мають максимальне значення, бо впливають на автономність пристрою. Тому тут краще використовувати алгоритми, що працюють швидко, наприклад логістична регресія або байєсовський класифікатор;

– стаціонарна система моніторингу в реальному часі. В цьому випадку найважливішим параметром стає час роботи алгоритму, тому краще використовувати логістичну регресію, байєсовський класифікатор чи нейронну мережу;

– дослідницька система для вивчення великих об’ємів даних. Тут час роботи має значення, але ще більше важлива точність системи. Так як отримання результатів в реальному часі не обов’язково, то найкращим варіантом будуть метод випадкового лісу чи нейронна мережа.

Таким чином, на вибір алгоритму впливає багато факторів, які потрібно враховувати при виборі найбільш відповідного задачам.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було проведено дослідження існуючих методів аналізу даних та пошуку схожих патернів, що стосується серцево-судинних захворювань. Проаналізовано предметну область, та виявлено потреби щодо аналізу даних.

Розглянуто сучасні пристрої для контроля серцево-судинної системи людини, такі як пульсометри, пульсоксиметри, електрокардіомонітори та розумні годинники.

Розглянуто сучасні методи аналізу великих об’ємів даних, використання машинного навчання та нейронних мереж для детектування патернів всередині даних.

Були досліджені декілька алгоритмів обробки даних, в тому числі нейронні мережі. Було порівняно точність роботи кожного з методів, їх час роботи та потреба в обчислювальних ресурсах.

Дослідження показало великий потенціал автоматизованого детектування серцево-судинних захворювань, можливість отримати високу точність передбачення наявності хвороби на основі даних від пульсометру.

Подальші дослідження можуть включати в себе детектування не тільки серцево-судинних захворювань, а також захворювань нервової системи, шлунково-кишкового тракту, опорно-рухового апарату, що, в свою чергу, допоможе покращити загальний рівень медицини та вдосконалити можливості лікування та надання допомоги.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kranjec, J., Beguš, S., Geršak, G., & Drnovšek, J. (2014). Non-contact heart rate and heart rate variability measurements: A review. Biomedical Signal Processing and Control, 13, 102-112.

2. Mark, R., & Moody, G. (1997). Mit-bih arrhythmia database 1997. URL http://ecg. mit. edu/dbinfo. html.

3. Shimmer. (2015). Shimmer ECG sensor Retrieved from http://www.shimmersensing.com

4. Allami, R., Stranieri, A., Balasubramanian, V., & Jelinek, H. F. (2017). A count data model for heart rate variability forecasting and premature ventricular contraction detection. Signal, Image and Video Processing, 11, 1427-1435.

5. Minarini, Giovanni. (2020). Root Mean Square of the Successive Differences as Marker of the Parasympathetic System and Difference in the Outcome after ANS Stimulation. 10.5772/intechopen.89827.

6. Ahonen, L., Cowley, B., Torniainen, J., Ukkonen, A., Vihavainen, A., & Puolamäki, K. (2016). Cognitive collaboration found in cardiac physiology: Study in classroom environment. PLOS ONE 11(7). doi:10.1371/journal.pone.0159178.

7. Bakhtiari, A. S., & Bouguila, N. (2014). A variational Bayes model for count data learning and classification. Engineering Applications of Artificial Intelligence, 35, 176-186.

8. Cameron, A. C., & Trivedi, P. K. (2013). Regression analysis of count data (Vol. 53): Cambridge University Press.

9. Wang, Jianhong & Lin, Xiaoyan. (2020). A Bayesian approach for semiparametric regression analysis of panel count data. Lifetime Data Analysis. 26. 10.1007/s10985-019-09471-3.

10. Ibrahim Musa, Saleh & Obini, Nweze. (2021). Model Selection for Time Series Count Data with Over-Dispersion. Asian Journal of Probability and Statistics. 60-73. 10.9734/ajpas/2021/v14i230326.

11. Montesinos-López, O. A., Montesinos-López, A., Crossa, J., Toledo, F. H., Montesinos-López, J. C., Singh, P., . . . Salinas-Ruiz, J. (2017). A Bayesian Poisson-lognormal model for count data for multiple-trait multiple-environment genomic-enabled prediction. G3: Genes, Genomes, Genetics, 7(5), 1595-1606.

12. Marchal, J., Cumming, S. G., & McIntire, E. J. (2017). Exploiting Poisson additivity to predict fire frequency from maps of fire weather and land cover in boreal forests of Québec, Canada. Ecography, 40(1), 200-209.

13. Wang, J., & Wang, J. (2016). Forecasting energy market indices with recurrent neural networks: Case study of crude oil price fluctuations. Energy, 102, 365-374.

14. Aloafi, Tahani & Elhag, Azhari & Jawa, Taghreed & Sayed-Ahmed, Neveen & Bayones, Fatimah & Bouslimi, Jamel & Marin, Marin. (2022). Predication and Photon Statistics of a Three-Level System in the Photon Added Negative Binomial Distribution. Symmetry. 14. 284. 10.3390/sym14020284.

15. Pan, J., & Tompkins, W. J. (1985). A real-time QRS detection algorithm. IEEE Transactions on Biomedical Engineering(3), 230-236.

16. Elman, J. L. (1990). Finding structure in time. Cognitive Science, 14(2), 179-211.

17. Wang, J., & Wang, J. (2016). Forecasting energy market indices with recurrent neural networks: Case study of crude oil price fluctuations. Energy, 102, 365-374.

18. Samarasinghe, S. (2016). Neural networks for applied sciences and engineering: From fundamentals to complex pattern recognition: CRC Press.

19. Eknoyan G. Adolphe Quetelet –the average man and indices of obesity. Nephrology, dialysis, transplantation: official publication of the European Dialysis and Transplant Association. / Eknoyan G. // European Renal Association. – 2008p.

20. Hunter T. The smart phone: An indispensable tool for the plastic surgeon? / Hunter T, Hardwicke J. // Journal of Plastic, Reconstructive & Aesthetic Surgery. - 2010p.

21. Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., . . . Stanley, H. E. (2000). Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals. Circulation, 101(23), 215-220.

22. Nabil, D., & Reguig, F. B. (2015). Ectopic beats detection and correction methods: A review. Biomedical Signal Processing and Control, 18, 228-244.

23. Bauer, A., Malik, M., Schmidt, G., Barthel, P., Bonnemeier, H., Cygankiewicz, I., Oto, A. (2008). Heart rate turbulence: Standards of measurement, physiological interpretation, and clinical use: International society for holter and noninvasive electrophysiology consensus. Journal of the American College of Cardiology, 52(17), 1353-1365.

24. Zhou, F., Jin, L., & Dong, J. (2017). Premature ventricular contraction detection combining deep neural networks and rules inference. Artificial Intelligence in Medicine, 79, 45-51.

25. Elhaj, F. A., Salim, N., Harris, A. R., Swee, T. T., & Ahmed, T. (2016). Arrhythmia recognition and classification using combined linear and nonlinear features of ECG signals. Computer Methods and Programs in Biomedicine, 127, 52-63.

26. Tarvainen, M. P., Niskanen, J., Lipponen, J. A., Ranta-Aho, P. O., & Karjalainen, P. A. (2014). Kubios HRV–heart rate variability analysis software. Computer Methods and Programs in Biomedicine, 113(1), 210-220.

27. Asl, B. M., Setarehdan, S. K., & Mohebbi, M. (2008). Support vector machine-based arrhythmia classification using reduced features of heart rate variability signal. Artificial Intelligence in Medicine, 44(1), 51-64.

28. Tsipouras, M. G., & Fotiadis, D. I. (2004). Automatic arrhythmia detection based on time and time–frequency analysis of heart rate variability. Computer Methods and Programs in Biomedicine, 74(2), 95-108.

29. Li, P., Liu, C., Wang, X., Zheng, D., Li, Y., & Liu, C. (2014). A low-complexity data-adaptive approach for premature ventricular contraction recognition. Signal, Image and Video Processing, 8(1), 111-120.

30. Cuesta, P., Lado, M. J., Vila, X. A., & Alonso, R. (2014). Detection of premature ventricular contractions using the RR-interval signal: A simple algorithm for mobile devices. Technology and Health Care, 22(4), 651-656.

31. Pierleoni, P., Pernini, L., Belli, A., & Palma, L. (2014). An Android-based heart monitoring system for the elderly and for patients with heart disease. International journal of telemedicine and applications. doi:10.1155/2014/625156.