Харківський національний університет імені В.Н. Каразіна

Факультет математики і інформатики

Кафедра прикладної математики

Кваліфікаційна робота

бакалавра

на тему «Задача прогнозування захворювань за допомогою методів машинного навчання»

Виконала: студентка групи МП 41
4 курсу
спеціальність 113 – прикладна математика
освітньо-професійна програма «Прикладна математика»

Ковтуненко М. О.

Науковий керівник: кандидат фіз.-мат. наук, доцент, Стєпанова К. В.

Рецензент: кандидат техн. наук, доцент кафедри вищої математики та економіко-математичних методів, Харківський національний економічний університет імені Семена Кузнеця, Денисова Т. В.

Харків - 2024 рік

АНОТАЦІЇ

Ковтуненко М.О. Задача прогнозування захворювань за допомогою методів машинного навчання. У роботі розглядаються методи машинного навчання, що застосовуються у сфері медицини. Детально розглядається задача передбачення ішемічної хвороби серця на даних про пацієнтів, зібраних з декількох лікарень світу. Для вирішення даної задачі класифікації використовується логістична регресія, метод k-найближчих сусідів, метод дерева рішень і метод випадкового лісу. Ефективність кожного з методів оцінюється за рядом метрик, за ними ж проводиться порівняння алгоритмів і визначення найкращого з них. Порівняння алгоритмів проілюстровано за допомогою діаграм, наведених у роботі.

Kovtunenko M.O. The problem of predicting diseases using methods of machine learning. This paper deals with the methods of machine learning used in the field of medicine. The problem of predicting coronary heart disease is studied in detail, using the data about patients collected from several hospitals in the world. For solving this classification problem various methods are used, such as logistic regression, k-nearest neighbours, decision tree and random forest. The performance of each method is assessed by a number of metrics, which are also used for comparing algorithms and determining the best performing one. The comparison of algorithms is illustrated with diagrams provided in the paper.

Зміст

ВСТУП

Надшвидкий розвиток інформаційних технологій в останні декілька десятиліть кардинально змінив ледь не всі сфери життя людей, зробивши повсякденні справи легшими і надавши засоби для розв’язання нетривіальних задач у різноманітних наукових галузях. Автоматизація процесів, діджиталізація баз даних, перехід до безконтактних способів оплати – все це приклади змін, які зробили можливими інформаційні технології. На особливу увагу заслуговує галузь штучного інтелекту, задача досліджень якої полягає у створенні комп’ютерних моделей, здатних виконувати завдання, які зазвичай передбачають втручання людини та її інтелектуальних здібностей. Штучний інтелект широко застосовується у соціальних мережах для покращення досвіду користувачів, для створення віртуальних асистентів на сайтах для онлайн шопінгу, для розпізнавання шахрайських банківських транзакцій тощо. Незважаючи на швидкість, з якою змінюються всі сфери людського життя, охорона здоров’я завжди має великий пріоритет як цілих держав, так і окремих організацій при розподілі ресурсів на різноманітні дослідження. Таким чином, дослідження штучного інтелекту також зробили важливий внесок у покращення ефективності праці медичних експертів. Однією з передових галузей штучного інтелекту є машинне навчання, яке є впливовим і в галузі медицини.

Машинне навчання – це галузь досліджень штучного інтелекту, зосереджена на побудові комп’ютерних систем, які використовують статистичні алгоритми з метою навчання з даних і узагальнювання на небачені дані. Цей зв'язок між математикою та комп'ютерними науками зумовлений унікальними обчислювальними проблемами, пов'язаними з побудовою статистичних моделей на основі великих масивів даних, які можуть включати в себе мільйони елементів. Діагностування хвороб є однією з найпоширеніших задач такої природи у сфері охорони здоров’я.

РОЗДІЛ 1: Аналіз задачі прогнозування захворювань

1.1 Огляд предметної області

Сфера медицини є однією з найпріоритетніших, коли мова заходить про розвиток методів, що дають можливість робити передбачення на основі доступних нам даних. Ідея вдосконалення медичної діагностики за допомогою комп'ютерних моделей є майже такою ж старою, як і самі комп'ютери. На початку 1960-х вчені застосували обчислювальні машини для аналізу медичних даних про пацієнтів, щоб робити передбачення про хвороби крові – це був один з найперших прикладів застосування комп'ютерних технологій у цій галузі. Розвиток штучного інтелекту грає велику роль у незупинному прогресі сучасної медицини, а конкретно розвиток машинного навчання дав можливість розвантажити лікарів, покращити точність прогнозів і якість допомоги, яка надається пацієнтам. Останні досягнення в цій сфері переважно виконують допоміжну роль у здатності лікарів та аналітиків виконувати свою роботу: ідентифікувати тенденції у сфері охорони здоров'я і розробляти моделі прогнозування захворювань. Під час пандемії COVID-19 застосування машинного навчання дозволило прискорити тестування та реагування лікарень у боротьбі з вірусом. Штучний інтелект також використовувався для ідентифікації генетичних послідовностей SARS-CoV-2 і створення вакцин. [1]

Методи машинного навчання вже показали себе як невід'ємну частину розвитку галузі медичних досліджень, а їх подальший розвиток призведе тільки до поліпшення швидкості, простоти і точності діагностування захворювань.

1.2 Огляд методів машинного навчання, що використовуються у медичній сфері

1.2.1 Кероване навчання

Типи машинного навчання зазвичай поділяються на дві основні категорії – кероване і некероване. При застосуванні керованого навчання (англ. *supervised learning*) комп'ютеру представляють зразки як вхідних, так і

бажаних вихідних даних, задля того щоб встановити загальне правило, яке відображує входи на виходи. Моделі керованого навчання також поділяються на методи регресії і класифікації. Регресійні моделі (лінійна регресія, метод випадкового лісу, метод опорних векторів) застосовуються для передбачень неперервних цільових змінних – ціна будинку, зарплатня співробітника компанії, передбачення ринкових тенденції. В свою чергу класифікаційні моделі (логістична регресія, наївний баєсів класифікатор, метод k-найближчих сусідів) є найбільш ефективними для задач класифікації, іншими словами передбачення дискретної цільової змінної або ж класу – спам/не спам, чоловік/жінка, оцінка за шкалою від 1 до 5.

Серед застосувань алгоритмів керованого машинного навчання у медицині можна виділити передбачення діагнозів за допомогою електронних медичних записів, які містять інформацію про історію хвороб пацієнтів у електронній формі, та автоматизовану інтерпретацію електрокардіографії (ЕКГ). У радіології автоматизоване виявлення солітарного вузла в легенях на рентгенівському знімку грудної клітки також може бути завданням керованого машинного навчання.[2]

1.2.2 Некероване навчання

Принциповою відмінністю некерованого навчання (англ. *unsupervised learning*) від керованого є відсутність бажаних виходів, які надаються алгоритму. Тобто при побудові моделі некерованого навчання задача полягає в пошуку прихованих закономірностей та угрупованні елементів датасету на їх основі. Природа некерованого навчання робить оцінку ефективності моделей більш складною, оскільки при їх побудові невідомо, які результати будуть отримані, і наскільки ці результати є інформативними для конкретного дослідження.

У медицині некероване навчання використовується для дослідження хвороб, які мають гетерогенну симптоматику і патогенез, що значно ускладнює виявлення закономірностей їх виникнення у пацієнтів. Таким чином, саме

методи некерованого навчання дали змогу лікарям виявити еозинфільний підтип астми.[2]

1.3 Огляд методів оцінки ефективності моделей

1.3.1 Матриця помилок

Матриця помилок – це інструмент для оцінки ефективності для задачі класифікації в машинному навчанні.[3]

Розглянемо випадок задачі бінарної класифікації – перед моделлю стоїть задача розподілу елементів датасету в дві групи. У цьому випадку матриця помилок представляє з себе таблицю з 4 різними комбінаціями прогнозованих і фактичних значень, як продемонстровано на рисунку 1.3.1.

Рисунок 1.3.1. Ілюстрація елементів матриці помилок.

Розглянемо кожний з елементів таблиці окремо:

- ІП – істинно позитивний (передбачили позитивний результат і він справді виявився позитивним);
- ХП – хибно позитивний (передбачили позитивний результат і він виявився негативним);
- ХН – хибно позитивний (передбачили негативний результат і він виявився позитивним);
- ІН – істинно негативний (передбачили негативний результат і він справді виявився негативним).

Отримані результати оцінюються за наступними параметрами:

1) **Чутливість** (істинно-позитивний рівень, ІПР):

$$\text{ІПР} = \frac{\text{ІП}}{\text{ІП} + \text{ХН}}$$

Наведене вище рівняння показує, скільки з усіх позитивних класів ми передбачили правильно.[3]

2) **Влучність** (прогностична значущість позитивного результату):

$$\text{ПЗ+} = \frac{\text{ІП}}{\text{ІП} + \text{ХП}}$$

Дане рівняння показує, скільки з усіх класів, які ми передбачили як позитивні, насправді є позитивними.[3]

3) **Точність** (діагностична ефективність):

$$\text{ДЕ} = \frac{\text{ІП} + \text{ІН}}{\text{ІП} + \text{ІН} + \text{ХП} + \text{ХН}}$$

Визначаємо, скільки з усіх класів (позитивних і негативних), ми передбачили правильно.[3]

4) **F1-міра**:

$$F = \frac{2 * \text{Чутливість} * \text{Влучність}}{\text{Чутливість} + \text{Влучність}}$$

Важко порівнювати дві моделі з низькою влучністю та високим рівнем чутливості або навпаки, тому, щоб зробити це порівняння можливим, використовується F1-міра, яка використовує середнє гармонійне значень замість середнього арифметичного.[3]

1.3.2 Кросс-валідація

Однією з найбільш поширених проблем машинного навчання є перенавчання моделі (англ. *overfitting*). Перенавчання відбувається тоді, коли статистична модель або алгоритм машинного навчання занадто добре припасовується до тренувальних даних, що в результаті робить неможливими точні прогнози на будь-яких інших даних. Це суперечить меті машинного навчання – узагальнення моделі за допомогою тренувальних даних для того,

щоб вона була здатна прогнозувати і класифікувати подальші дані з високою точністю.

Кросс-валідація – це техніка, яка допомагає у вирішенні проблеми перенавчання. Існують різні види кросс-валідації, але їх всі поєднують один принцип:

- розділити датасет на кілька підвибірок;
- виключати по одній підвибірці і тренувати модель на решті даних;
- протестувати модель на виключеній підвибірці.[4]

Розглянемо два види кросс-валідації:

***k*-кратна кросс-валідація** (англ. *k-fold cross-validation*) – суть техніки полягає в розділенні датасету на k підвибірок і проведенні k ітерацій процедури, описаної нижче, на всіх даних:

1) Виключити одну підвибірку в якості тестового датасету;
2) Використати решту груп як тренувальний датасет;
3) Побудувати модель на тренувальному датасеті та оцінити її на тестовому датасеті;
4) Зберегти оцінку та відкинути модель.[4]

В кінці перевірки результати можна усереднити, щоб отримати фінальну оцінку. Значення k обирається таким чином, щоб кожна тренувальна або тестова підвибірка була достатньо великою, щоб бути статистично репрезентативною для ширшого датасету.[4]

Рисунок 1.3.2. Ілюстрація k-кратної кросс-валідації.

Кросс-валідація з виключенням по одному (англ. l*eave-one-out cross-validation*) – цей підхід виключає одну точку з навчального датасету, тобто, якщо вхідна вибірка містить *n* елементів, то для навчання моделі використовується *n-1* елемент.[4] Цей процес повторюється для всіх комбінацій, в яких вхідна вибірка може бути розділена таким чином, а потім помилка усереднюється для всіх випробувань, щоб отримати загальну ефективність.[4] Кількість можливих комбінацій дорівнює кількості елементів у вхідній вибірці, тобто *n*.

1.3.3 Root Mean Square Error

Root Mean Square Error (RMSE) – це стандартне відхилення залишків (відстаней точок до лінії регресії), тобто це значення, яке вказує на те, наскільки дані сконцентровані навколо лінії найкращої відповідності.[5]

Формула для обчислення RMSE виглядає наступним чином:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}, \text{ де}$$

y_i – залежна змінна регресії, $\hat{y}_i$ – передбачені дані залежної змінної, n – розмір вибірки.[5]

Коли в якості вхідних даних для RMSE використовуються стандартизовані спостереження та прогнози, існує прямий зв'язок з коефіцієнтом кореляції, тобто, наприклад, якщо коефіцієнт кореляції дорівнює 1, то RMSE буде дорівнювати 0, оскільки всі точки лежать на лінії регресії і помилок немає.[5]

1.3.4 Критерій хі-квадрат

Формула статистики хі-квадрат, що використовується при обчисленні критерію хі-квадрат, має наступний вигляд:

$$\mathrm{X}_c^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}, \text{ де}$$

c – ступені свободи, O_i – спостережувані значення, E_i – очікувані значення.[7]

У статистиці розрізняють два типи змінних: числові та нечислові (категорійні) змінні, статистика хі-квадрат є одним зі способів показати зв'язок між двома категорійними змінними.[7]

Існує кілька варіантів статистики хі-квадрат, вибір однієї з них залежить від того, як були зібрані дані і яка гіпотеза перевіряється (незалежність, узгодженість тощо). Незважаючи на відмінності, всі варіації цієї статистики використовують одну ідею, яка полягає в тому, що ми порівнюємо очікувані значення зі значеннями, які фактично були отримані.[7]

1.3.5 ROC-крива

ROC-крива або ж *Receiver Operating Characteristic curve* – це метод оцінки ефективності бінарної класифікації, здійсненої моделлю машинного навчання. Сутність методу полягає в тому, щоб побудувати співвідношення між істинно-позитивним і хибно-позитивним рівнями моделі при різних порогах класифікації.

Поріг класифікації – це значення, яке розділяє негативні і позитивні результати при бінарній класифікації. Частіше за все воно дорівнює 0.5 за замовчуванням, тобто, якщо передбачене значення залежної змінної менше 0.5, то результат визначається як негативний, а якщо більше, то як позитивний.

Цей метод стає в нагоді, наприклад, тоді, коли в датасеті наявна дуже маленька кількість записів одного з класів. Якщо з 10000 листів лише 100 є спамом, то навіть при помилковій класифікації цих 100 листів як не спаму ми отримаємо значення точності передбачень 99%. Точність, або ж діагностична ефективність, яка вже згадувалася в розділі про матрицю помилок, є легкою для розуміння оцінкою ефективності моделі, але у випадку небалансованого датасету вона може вводити в оману.

Величина, за допомогою якої оцінюється графік ROC-кривої, називається AUC – *Area Under the Curve*, іншими словами площа під кривою. Значення AUC завжди лежить між 0 і 1 і чим воно більше, тим кращою є модель класифікації. Як зазначено в статті [6], якщо AUC = 1, то класифікатор правильно розподіляє всі позитивні і негативні елементи класів; якщо AUC = 0, то всі негативні класи були передбачені як позитивні і навпаки; якщо AUC = 0.5, то модель непридатна для правильної бінарної класифікації – всі передбачені класи будуть або сталими, або випадковими.

Отже, про здатність моделі бінарної класифікації ефективно розподіляти елементи у два класи свідчить значення AUC між 0.5 і 1.

1.4 Постановка задачі

Серцево-судинні захворювання – клас захворювань, які пов'язані з патологією серця або кровоносних судин. Серцево-судинні захворювання є найбільш частою причиною смерті у світі. За оцінками ВООЗ 17,9 мільйона людей загинули від серцево-судинних захворювань у 2016 році, що становить 31% від усіх смертей у світі. З них 85% смертей — через серцевий напад та інсульт, а ішемічна хвороба серця є основною причиною летальності від серцево-судинних захворювань.[8]

Через велику поширеність ССЗ добре досліджені основні фактори виникнення хвороб цього класу:

- генетика;
- вік і стать;
- нездорове харчування;
- шкідливі звички (вживання алкоголю і тютюну);
- низький рівень фізичної активності.[8]

Хоча проблема серцево-судинних захворювань не є новою, саме вони досі є відповідальними за велику частку передчасних смертей (до 70 років) у

світі – за даними ВООЗ серед 17 мільйонів передчасних смертей від неінфекційних захворювань в 2019 році 38% були спричинені ССЗ.

Передчасні смерті від серцево-судинних захворювань можуть бути попереджені вчасним діагностуванням і наданням необхідної медичної допомоги пацієнтам. На ранніх стадіях діагностування може бути можливим завдяки виявленню таких факторів, як підвищений артеріальний тиск, підвищений рівень цукру і жирів у крові, ожиріння тощо.

Методи машинного навчання вже показали себе як ефективний інструмент у прискоренні діагностування хвороб, їх застосування є поширеним і для допомоги у виявленні серцево-судинних захворювань. Доступ лікарів до великих масивів медичних даних пацієнтів у електронному вигляді є незаперечною перевагою сучасних досліджень ССЗ, а статистичні методи і моделі машинного навчання дають змогу виявляти в них закономірності, які не бачить людське око.

Для дослідження проблеми передбачення хвороб серця було обрано датасет, що містить записи про фактори, що можуть свідчити про наявність ССЗ в пацієнтів, а також записи про наявність чи відсутність ішемічної хвороби серця в цих людей. Ці дані були зібрані з чотирьох лікарень світу (Клівленд, Огайо, США; Будапешт, Угорщина; Цюрих, Швейцарія; Лонг-Біч, Каліфорнія, США). Даний датасет був опублікований Університетом Каліфорнії в Ірвайні з дозволом на використання за ліцензією Creative Commons Attribution 4.0 International (CC BY 4.0).

Датасет містить записи про 918 пацієнтів, а саме наступні 11 характеристик про них:

1. Age: вік пацієнта;
2. Sex: стать пацієнта (M – чоловік; F – жінка);
3. ChestPainType: вид болю в грудній клітці (TA – типова стенокардія, ATA – атипова стенокардія, NAP – інший вид болю, не стенокардія, ASY – біль відсутній);

4. RestingBP: артеріальний тиск в стані спокою (мм рт. ст.);
5. Cholesterol: рівень холестерину в крові (мг/dl);
6. FastingBS: рівень цукру в крові натщесерце (1 – якщо FastingBS > 120 мг/dl, 0 – в іншому випадку);
7. RestingECG: результат електрокардіограми в стані спокою (Normal – нормальний; ST – присутні аномалії сегмента ST; LVH – присутні ознаки можливої або наявної гіпертрофії лівого шлуночка серця за критерієм Естеса);
8. MaxHR: найвища досягнена частота пульсу (числове значення між 60 і 202);
9. ExerciseAngina: стенокардія спричинена фізичним навантаженням (Y – присутня; N – відсутня);
10. Oldpeak: депресія ST-сегменту ЕКГ, спричинена фізичним навантаженням, відносно стану спокою (числове значення);
11. ST_Slope: нахил ST-сегменту ЕКГ під час піку фізичного навантаження (Up – нахил вгору; Flat – плоский; Down – нахил вниз)

Останній стовпчик датасету «HeartDisease» містить в собі цільову змінну – наявність ішемічної хвороби серця. Якщо її значення дорівнює 1, то в пацієнта було виявлено ІХС, якщо 0 – ІХС немає.

Отже, перед нами стоїть задача бінарної класифікації, для вирішення якої будуть застосовані методи керованого машинного навчання, які будуть детально оглянуті в наступному розділі. За допомогою бібліотек мови програмування Python буде проведено попередню обробку даних, поділ датасету на тренувальний і тестовий, імплементацію методів керованого навчання і оцінку ефективності кожного з методів. Після цього можна буде визначити, який із розглянутих алгоритмів показав найкращі результати у прогнозуванні наявності ішемічної хвороби серця в пацієнтів.

РОЗДІЛ 2: Вибір методів для прогнозування і оцінки результатів

Як було зазначено в першому розділі, для передбачення наявності ішемічної хвороби в пацієнтів буде вирішуватися задача бінарної класифікації. Це означає, що будуть розглядатися лише методи, які є придатними для задач класифікації дискретних значень.

Серед регресійних моделей найпоширенішою є лінійна регресія, яка прогнозує неперервні значення залежної змінної, що не підходить у випадку даного датасету. З іншого боку, логістична регресія є поширеним алгоритмом бінарної та мультикласової класифікації дискретної залежної змінної. Також розглянемо метод *k*-найближчих сусідів, який для класифікації оцінює відстані від елементу до заздалегідь визначеної кількості його сусідів серед тренувальних даних. Серед інших підходів машинного навчання до бінарної класифікації будуть розглянуті методи основані на моделюванні дерев рішень – класифікацію за допомогою побудови дерева рішень і ансамблеву модель випадковий ліс.

2.1 Обрані методи машинного навчання для прогнозування

2.1.1 Логістична регресія

Логістична регресія отримала свою назву від функції, яка лежить в основі методу – логістичної або ж сигмоїдної функції. Логістична функція з’явилася в результаті дослідження динаміки зміни населення популяції і має наступний вигляд:

$$y = \frac{1}{1 + e^{-x}}$$

Ця функція переводить будь-яке число $x \in \mathbb{R}$ в інтервал (0; 1). [9]

Логістична регресія використовується тоді, коли результат (залежна змінна) це взаємовиключні події – наприклад, визначення статі людини (чоловік/жінка) за зростом, чи є лист спамом чи ні; у медицині – для діагностування хвороб або виявлення злоякісних пухлин.

Наведемо приклад рівняння логістичної регресії:

$$y = \frac{1}{1+e^{-z}},$$

$$\boldsymbol{z} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n, \text{ де}$$

y – передбачений результат,

$x_1, x_2, \ldots, x_n$ – значення незалежних змінних (атрибутів),

$\beta_0, \beta_1, \ldots, \beta_n$ – коефіцієнти логістичної регресії, які зазвичай визначаються за допомогою методу максимальної правдоподібності – це поширений алгоритм у побудові моделей машинного навчання.

Найкращі коефіцієнти призведуть до моделі, яка передбачить значення дуже близьке до 1 для класу за замовчуванням (наприклад, правда) і значення дуже близьке до 0 для іншого класу (наприклад, брехня). [9]

2.1.2 Метод k-найближчих сусідів

В основі методу *k*-найближчих сусідів полягає обчислення відстаней від елементів тестового датасету до вибірки з k найближчих елементів тренувального датасету. Алгоритм може використовуватися як для задач регресії, так і задач класифікації.

Для класифікації за допомогою методу *k*-найближчих сусідів будується модель, яка здійснює класифікацію наступним чином:

1. для кожного елементу тестового датасету знаходить задану кількість найближчих сусідів серед тренувальних даних, обчислюючи відстань до них;
2. підраховує, скільки з *k* сусідів належить до кожного з можливих класів;
3. відносить даний елемент до класу, якому належить більшість із його сусідів.

Для пошуку відстані між елементом $y = (y_1, y_2, \ldots, y_n)$, який ми хочемо класифікувати, і елементом $x = (x_1, x_2, \ldots, x_n)$ з тренувального датасету можуть бути використані різні метрики, наприклад:

- евклідова метрика:

$$\sqrt{\sum_{i=1}^{k}(y_i - x_i)^2}$$

- мангеттенська метрика:

$$\sum_{i=1}^{k}|y_i - x_i|$$

Оскільки алгоритм *k*-найближчих сусідів покладається на пошук відстаней до елементів, ефективність моделі може значно підвищити масштабування вхідних даних, яке буде розглянуте детальніше в розділі 2.2.

2.1.3 Дерево рішень

Дерево рішень – це непараметричний алгоритм керованого машинного навчання для задач класифікації та регресії.[11] Самі по собі дерева рішень є ієрархічними моделями, які складаються з вузлів і голок (ребер).

Серед вузлів розрізняють корінь, з якого починається класифікація всіх елементів; вузли прийняття рішень, які є результатом поділу кореневого вузлу на декілька випадків; листя – вузли, досягнення яких означає неможливість подальшого поділу дерева, тобто на цьому етапі визначається фінальний клас елементу.[11] Шлях, утворений послідовністю вузлів прийняття рішень, називається гілкою дерева рішень.

Алгоритм дерева рішень для задачі класифікації працює наступним чином:

- алгоритм починається з вершини, тобто кореневого вузла, що представляє весь датасет;
- алгоритм шукає найважливішу характеристику або питання, які розбивають дані на найбільш чіткі групи;

- залежно від відповіді на це питання, дані поділяються на менші підмножини, створюючи нові гілки, кожна з них представляє можливий маршрут по дереву;
- алгоритм продовжує ставити запитання і розділяти дані на кожній гілці, поки не досягає листя, тобто вузли, що представляють результати класифікації.[11]

Алгоритм дерева прийняття рішень базується на оцінці невизначеності моделі на кожному з вузлів, або ж ентропії, яка визначається наступною формулою:

$$E(M) = \sum_{m \in M} p(m) \log p(m), \text{ де}$$

M – підмножина тренувальних даних; *m* – класи, на які поділена підмножина; *p(m)* – ймовірність, що буде обрано цей клас.

Ентропія використовується для оцінки приросту інформації (англ. *information gain*) – величини, яка є вирішальною у виборі, яка ознака має становитися кореневим вузлом або вузлом прийняття рішень. Приріст інформації визначається наступним чином:

$$IG(Y|X) = E(Y) - E(Y|X), \text{ де}$$

$E(Y|X)$ – умовна ентропія:

$$E(Y|X) = -\sum_{x \in \mathcal{X},\ y \in \mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(x)}, \text{ де}$$

$\mathcal{X}, \mathcal{Y}$ – носії множин *X* і *Y*.

2.1.4 Випадковий ліс

Випадковий ліс є ще одним алгоритмом машинного навчання, який може використовуватися для задач класифікації і регресії. Це ансамблева модель, в основі якої лежить раніше розглянутий метод дерев рішень. Іншими словами, "ліс", який будує ця модель – це ансамбль дерев рішень, які зазвичай навчаються за допомогою методу беггінгу. [12]

Сутність беггінгу або ж бутстрепової агрегації полягає в тому, щоб об’єднувати «слабкі» моделі (дерева) з високим значенням дисперсії з метою отримання однієї «сильної» моделі (ліс) з нижчою дисперсією, ніж в індивідуальних дерев. Беггінг для ансамблевої моделі виконується за наступними кроками:

1. є початковий тренувальний датасет, що містить *n* екземплярів;
2. створюється *m* підмножин даних з тренувального датасету, для кожної з них береться підмножина з *N* елементів з початкового датасету, при цьому конкретний елемент може бути використаний більше ніж один раз;
3. для кожної підмножини даних відповідні слабкі моделі тренуються незалежно, ці моделі є однотипними (гомогенними);
4. кожна з моделей робить передбачення, які потім агрегуються в одне передбачення. [13]

2.2 Вибір методів підготовки даних

Нами були оглянуті алгоритми керованого навчання, які будуть застосовані для класифікації пацієнтів за наявністю або відсутністю в них ішемічної хвороби серця. Окрім вибору методів, які будуть застосовані для даної задачі класифікації, велику роль також може зіграти попередня обробка даних. Для деяких моделей керованого машинного навчання доцільна підготовка даних може кардинально покращити результати її передбачень.

Розглянемо два методи попередньої обробки даних – нормалізація даних і виявлення мультиколінеарності.

Масштабування ознак - це метод попередньої обробки даних, який використовується для трансформації значень змінних у датасеті до однакового масштабу.[14]

Масштабування ознак є важливим, коли в датасеті наявні дані у різних діапазонах, що може змусити модель віддавати перевагу тим ознакам, значення яких мають більший розмах. Одним з підходів до масштабування даних є

нормалізація, також відома як *Min-Max scaling*. При нормалізації кожне значення атрибуту датасету отримує нове значення, яке лежить в інтервалі [0, 1]. Нормалізація ознак здійснюється наступним чином:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}, \text{де}$$

x' – нове значення ознаки для конкретного елементу;

x – початкове значення ознаки для конкретного елементу;

x_{min} – мінімальне значення ознаки серед всіх елементів;

x_{max} – максимальне значення ознаки серед всіх елементів.

Мультиколінеарність – це статистичне явище, яке виникає, коли дві або більше незалежних змінних у регресійній моделі сильно корелюють між собою, тобто мультиколінеарність вказує на сильний лінійний зв'язок між незалежними змінними (ознаками). [15]

Мультиколінеарність може виникати на етапі створення датасету, наприклад, якщо в нього включають ознаки, які залежать від інших ознак, що не додає нової інформації в датасет і може погіршити якість передбачень моделі.

Одним зі способів виявлення мультиколінеарності є обчилення фактору інфляції дисперсії – VIF (*Variance inflation factor*):

$$VIF_i = \frac{1}{1 - R_i^2}, \text{де}$$

R_i^2 – коефіцієнт детермінації рівняння регресії, яке має наступний вигляд для кожної з *i* характеристик, для якої обчислюється VIF (при *i*=1):

$$X_i = \alpha_0 + \alpha_2 X_2 + \alpha_3 X_3 + \cdots + +\alpha_k X_k + \varepsilon, \text{де}$$

α_0 – константа; ε – коефіцієнт похибки.

Як зазначено в статті [15], фактор інфляції дисперсії, що перевищує 10 (іноді цим порогом є 5), вказує на високу мультиколінеарність між даною незалежною змінною та іншими.

2.3 Вибір методів оцінки ефективності моделей

Наступним кроком є вибір методів, за допомогою яких ми будемо оцінювати ефективність індивідуальних моделей, а також порівнювати їх між собою. У випадку задачі бінарної класифікації найбільше інформації про якість моделі нам надає **матриця помилок**, яка була детально розглянута в розділі 1.3.1. Всі моделі будуть порівнюватися за наступними характеристиками:

- точність;
- чутливість;
- F-міра;
- значення площі під ROC-кривою (AUC).

Також ефективність передбачень кожної з моделей буде оцінена за допомогою 9-кратної **кросс-валідації**, детальніше про цей метод мова йшла у розділі 1.3.2. Значення k було обрано таким, оскільки число елементів у нашому датасеті (918) націло ділиться на 9 і такий поділ можна вважати репрезентативним для загальної оцінки ефективності моделі на різноманітних поділах датасету на тренувальний і тестовий.

2.4 Використані бібліотеки мови програмування Python:

Найпоширенішим інструментом для імплементації оглянутих раніше моделей керованого машинного навчання є інтерпретована мова програмування Python. Саме ця мова програмування надає доступ до таких потужних бібліотек, як Pandas і Scikit-learn, що стали фундаментальними у програмній реалізації проєктів аналізу даних та машинного навчання.

2.4.1 Pandas

Бібліотека Pandas була створена для полегшення роботи з даними, а саме їх аналізу та маніпуляцій над ними. Структура даних, якою користується ця бібліотека, називається датафрейм. Завантаження даних з, наприклад, файлу формату CSV у датафрейм Pandas дає змогу використовувати різноманітні

операції з цієї бібліотеки для полегшення роботи з даними – пошук порожніх клітинок, видалення і додавання стовпчиків та строчок, зміна типів даних тощо.

2.4.2 Scikit-learn

Бібліотека Scikit-learn містить в собі алгоритми для тренування різноманітних моделей як керованого, так і некерованого машинного навчання. Використання цієї бібліотеки значно полегшує задачу побудови моделей машинного навчання, оскільки вона містить методи для поділу датасету на тренувальний і тестовий, створення моделі, її навчання на тренувальних даних та застосування моделі на тестових даних. Також Scikit-learn дає доступ до різноманітних метрик, що дозволяють оцінити якість побудованої моделі, як для задач регресії, так і для задач класифікації.

2.4.3 Matplotlib

Matplotlib – це всеосяжна бібліотека, яка містить інструменти для побудови статичних, анімованих та інтерактивних візуалізацій у мові Python. Ця бібліотека є потужним інструментом у задачах аналізу даних в поєднанні з бібліотекою Pandas. Одним з багатьох прикладів використання Matplotlib у машинному навчанні є побудова ROC-кривих.

РОЗДІЛ 3: Програмна реалізація та аналіз результатів

3.1 Попередня обробка і огляд даних

Для роботи з даними про пацієнтів у форматі CSV вони були завантажені у датафрейм, створений за допомогою методу *read_csv* з бібліотеки Pandas. Наступним кроком є попередній огляд даних, а саме перевірка на наявність порожніх клітинок або невідповідних типів даних у стовпчиках, яка здійснюється за допомогою методу *info* з тієї ж бібліотеки. Отримана інформація наведена на рисунку 3.1.1.

Рисунок 3.1.1. Результат застосування методу info().

```
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Age             918 non-null    int64  
 1   Sex             918 non-null    object 
 2   ChestPainType   918 non-null    object 
 3   RestingBP       918 non-null    int64  
 4   Cholesterol     918 non-null    int64  
 5   FastingBS       918 non-null    int64  
 6   RestingECG      918 non-null    object 
 7   MaxHR           918 non-null    int64  
 8   ExerciseAngina  918 non-null    object 
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object 
 11  HeartDisease    918 non-null    int64  
dtypes: float64(1), int64(6), object(5)
```

Отже, в датасеті немає порожніх клітинок, а також дані в кожній колонці належать одному типу. Можемо побачити, що стовпчики «Sex», «ChestPainType», «RestingECG», «ExerciseAngina», «ST_Slope» містять інформацію типу *object*, тобто текстові поля – в цих колонках містяться категорійні змінні, зміст яких був розібраний в розділі 1.4. Для подальшої роботи з цими даними використовуємо метод *LabelEncoder* з бібліотеки Scikit-learn, який закодує ці змінні як числа 0, 1 або 0, 1, 2 (в залежності від кількості категорій в стовпчику).

Для тренування і тестування кожної з моделей машинного навчання датасет буде розділено в співвідношенні 25:75. Кожна з моделей буде застосована як на необроблених даних, так і на нормалізованих. Нормалізація, розглянута в розділі 2.2, буде виконуватися методом *MinMaxScaler* з бібліотеки Scikit-learn і буде застосована тільки для ознак (незалежних змінних), оскільки значення залежної змінної вже лежать на відрізку [0, 1].

Метою дослідження є порівняння ефективності різних моделей керованого машинного навчання у передбаченні наявності ішемічної хвороби серця в пацієнтів, тому кожна з моделей буде тренуватися на 688 елементах датасету і тестуватися на 230 елементах. Порівняння буде здійснюватися за характеристиками, визначеними в розділі 2.3.

3.2 Застосування методів для передбачення ішемічної хвороби серця

3.2.1 Класифікація за допомогою логістичної регресії

Першою розглядаємо регресійну модель для задачі класифікації – логістичну регресію. У цьому випадку перед побудовою моделі буде також підраховано фактор інфляції дисперсії (VIF) за допомогою відповідного методу з бібліотеки Statsmodels мови Python, щоб виявити можливу лінійну залежність між даними в датасеті.

Рисунок 3.2.1.1. Результат застосування методу variance_inflation_factor.

	variables	VIF
0	Age	30.261977
1	Sex	4.912441
2	ChestPainType	1.991361
3	RestingBP	46.835556
4	Cholesterol	5.227740
5	FastingBS	1.482939
6	RestingECG	3.535198
7	MaxHR	26.142683
8	ExerciseAngina	2.521587
9	Oldpeak	2.543331
10	ST_Slope	9.390390

	variables	VIF
0	Age	19.032324
1	Sex	4.848800
2	ChestPainType	1.990709
3	Cholesterol	5.076344
4	FastingBS	1.478692
5	RestingECG	3.444089
6	MaxHR	22.310280
7	ExerciseAngina	2.451644
8	Oldpeak	2.531016
9	ST_Slope	9.190735

На рисунку 3.2.1.1 зображені значення фактору інфляції дисперсії для кожної з ознак в датасеті (ліворуч) і значення VIF для всіх ознак, якщо

прибрати найбільш корельовану з іншими – «RestingBP», тобто артеріальний тиск пацієнта (праворуч). Експериментальним шляхом було виявлено, що видалення цієї ознаки покращує ефективність моделі логістичної регресії, що буде показано детальніше в цьому розділі, але видалення інших ознак з високим значенням VIF (більше 10) призводить до погіршення ефективності моделі. Це явище можна пояснити тим, що ми працюємо з доволі невеликою вибіркою даних, тому втрата навіть двох з одинадцяти атрибутів не призводить до покращення передбачень, а навпаки до погіршення.

Отже, будуємо модель логістичної регресії для повної вибірки – для початкових та нормалізованих даних. Для цього застосовуємо алгоритм *LogisticRegression* з Scikit-learn з параметрами за замовчуванням, оскільки було виявлено, що їх зміна не призводить до підвищення ефективності на нашому датасеті. Код програми для моделей з цього розділу наведено в Додатку 1.

Таблиця 3.2.1.1. Результати застосування логістичної регресії на повній вибірці.

	Початкові дані	Нормалізовані дані
Істинно позитивна класифікація	77	78
Істинно негативна класифікація	125	125
Хибно позитивна класифікація	12	11
Хибно негативна класифікація	16	16
Точність	87.83%	88.26%
Чутливість	82.80%	82.98%
F1-міра	89.93%	90.25%
AUC	0.914	0.917
Середня точність кросс-валідації	83.66%	83.66%

Як можна побачити в таблиці 3.2.1.1, нормалізація даних призвела до покращення ефективності моделі за всіма метриками окрім кросс-валідації, оцінка за допомогою якої не змінилася. Тепер подивимося на результати

застосування логістичної регресії для датасету без стовпчику з інформацією про артеріальний тиск пацієнтів.

Таблиця 3.2.1.2. Результати застосування логістичної регресії на вибірці з 10 ознаками.

	Початкові дані	Нормалізовані дані
Істинно позитивна класифікація	78	78
Істинно негативна класифікація	127	126
Хибно позитивна класифікація	11	11
Хибно негативна класифікація	14	15
Точність	89.13%	88.70%
Чутливість	84.78%	83.87%
F1-міра	91.04%	90.65%
AUC	0.913	0.917
Середня точність кросс-валідації	83.55%	83.77%

З таблиці 3.2.1.2 можна побачити, що у випадку вибірки з 10 ознаками нормалізація даних призвела до погіршення точності, чутливості та F1-міри класифікацій, а також незначного покращення площі під ROC-кривою та середньої точності кросс-валідації. Для подальшого порівняння моделей залишимо результати з кращою точністю і чутливістю передбачень, тобто для нормалізованих даних на повній вибірці і ненормалізованих на вибірці з 10 ознаками.

3.2.2 Класифікація за допомогою методу k-найближчих сусідів

Наступним застосуємо класифікацію методом k-найближчих сусідів, який реалізується за допомогою алгоритму *KNeighborsClassifier* бібліотеки Scikit-learn. Код програми для побудови цієї та подальших моделей наведено в Додатку 2. Головним параметром при побудові цієї моделі є кількість *k* сусідів, яка зазвичай є доволі невеликою, залишаємо стандартне значення 5. Параметр *p* встановлюємо 1, що відповідає використанню мангеттенської метрики для

розрахунку відстаней замість евклідової – це призводить до побудови більш ефективної моделі на нашому датасеті.

Таблиця 3.2.2. Результати застосування методу k-найближчих сусідів

	Початкові дані	Нормалізовані дані
Істинно позитивна класифікація	60	78
Істинно негативна класифікація	104	123
Хибно позитивна класифікація	29	11
Хибно негативна класифікація	37	18
Точність	71.30%	87.39%
Чутливість	61.86%	81.25%
F1-міра	75.91%	89.45%
AUC	0.76	0.929
Середня точність кросс-валідації	73.42%	86.38%

Отримані результати підтверджують, що нормалізація даних є принципово важливим кроком у побудові моделі на основі алгоритму k-найближчих сусідів, оскільки цей метод заснований на обчисленні відстаней до елементів датасету. Була отримана дещо нижча точність, ніж при застосуванні логістичної регресії, але значення AUC та середня точність кросс-валідації виявилися поки що найвищими.

3.2.3 Класифікація за допомогою дерева рішень

Тепер розглядаємо метод дерева рішень, імплементація якого здійснюється алгоритмом *DecisionTreeClassifier* з бібліотеки Scikit-learn. При побудові моделі встановлюємо обмеження на максимальну глибину дерева (довжину однієї гілки від кореня до листа) параметром max_depth=5, що зумовлено невеликим обсягом датасету і допомагає запобігти перенавчанню моделі. Також через параметр criterion встановлюємо використання ентропії для оцінки невизначеності моделі на кожному з вузлів, ентропія і приріст інформації були детально розглянуті в розділі 2.1.3.

Таблиця 3.2.3. Результати застосування методу дерева рішень

	Початкові дані	Нормалізовані дані
Істинно позитивна класифікація	79	79
Істинно негативна класифікація	125	125
Хибно позитивна класифікація	10	10
Хибно негативна класифікація	16	16
Точність	88.70%	88.70%
Чутливість	83.16%	83.16%
F1-міра	90.58%	90.58%
AUC	0.909	0.909
Середня точність кросс-валідації	84.53%	84.53%

Як можна побачити в таблиці 3.2.3, нормалізація даних ніяк не вплинула на результати класифікації методом дерева рішень, на відміну від розглянутого раніше метода k-найближчих сусідів. Отримана модель, знову таки, показала доволі високі результати за всіма метриками, що розглядаються для порівняння алгоритмів.

3.2.4 Класифікація за допомогою випадкового лісу

Останньою розглядаємо метод випадкового лісу – ансамблеву модель, в основі якої лежить побудова дерев рішень. Ця модель будується за допомогою алгоритму *RandomForestClassifier* з бібліотеки Scikit-learn. Знову таки, використовуємо ентропію для оцінки невизначеності моделі на вузлах дерев і обмежуємо максимальну глибину дерев параметром max_depth=3 – ця модель будує багато дерев, а не одне, як попередній метод, тому менше обмеження дає кращий результат.

Таблиця 3.2.3 Результати застосування методу випадкового лісу

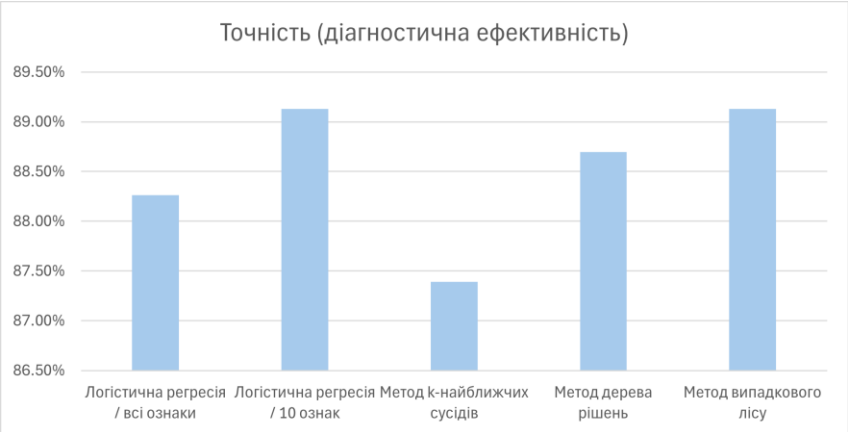
	Початкові дані	Нормалізовані дані
Істинно позитивна класифікація	78	78
Істинно негативна класифікація	127	127
Хибно позитивна класифікація	11	11
Хибно негативна класифікація	14	14
Точність	89.13%	89.13%
Чутливість	84.78%	84.78%
F1-міра	91.04%	91.04%
AUC	0.939	0.94
Середня точність кросс-валідації	84.31%	84.53%

Як і очікувалося, нормалізація не мала суттєвого впливу і на результати отримані методом випадкового лісу, оскільки модель лісу складається з багатьох дерев рішень. При застосуванні методу випадкового лісу було отримано найвище значення AUC серед усіх розглянутих моделей. У порівняння візьмемо кращий результат, тобто отриманий на нормалізованих даних.

3.3 Порівняння отриманих результатів

Почнемо порівняння отриманих результатів з точності класифікації моделей.

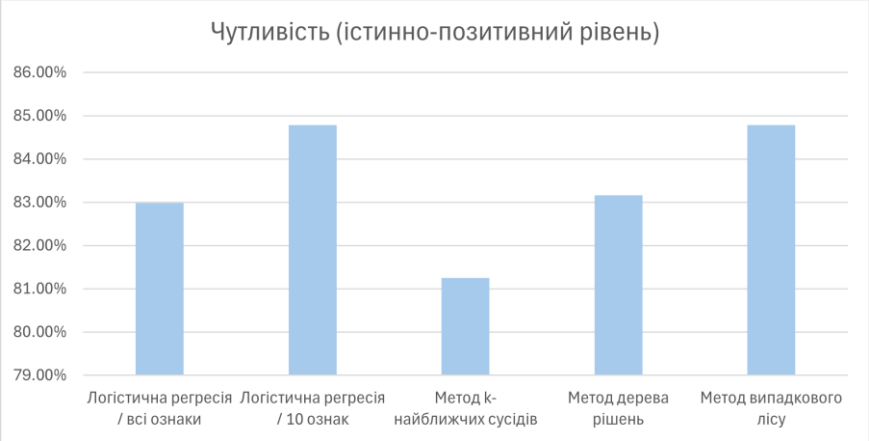
Рисунок 3.3.1. Діаграма точності застосованих моделей.



Точність вказує на те, скільки всього класифікацій було зроблено правильно, тобто як позитивних, так і негативних. Як можна побачити на рисунку 3.3.1, в нашому випадку точність виявилася найвищою в моделі логістичної регресії після видалення однієї з ознак і в моделі випадкового лісу, третьою за точністю виявився алгоритм дерева рішень.

Хоча точність і є легкою для розуміння метрикою, ми працюємо з прикладною задачею діагностики ішемічної хвороби серця. Тому наступною розглянемо чутливість – здатність моделі правильно класифікувати позитивні класи, які вказують на наявність хвороби у нашому випадку.

Рисунок 3.3.2. Діаграма чутливості застосованих моделей.



З рисунку 3.3.2 бачимо, що за чутливістю першими знову таки виявилися моделі випадкового лісу та логістичної регресії для 10 ознак з методом дерева рішень на третьому місці. Це вказує на те, що цим моделям вдалося зробити якомога більше істинно позитивних класифікацій і, що не менш важливо, якомога менше хибно негативних класифікацій.

Низький показник хибно негативних класифікацій є однією з найважливіших оцінок ефективності моделі машинного навчання у багатьох прикладних медичних задачах, оскільки ішемічна хвороба серця – це лише один приклад захворювання, діагностування якого на ранніх стадіях може значно полегшити лікування.

Рисунок 3.3.3. Діаграма F1-міри застосованих моделей.

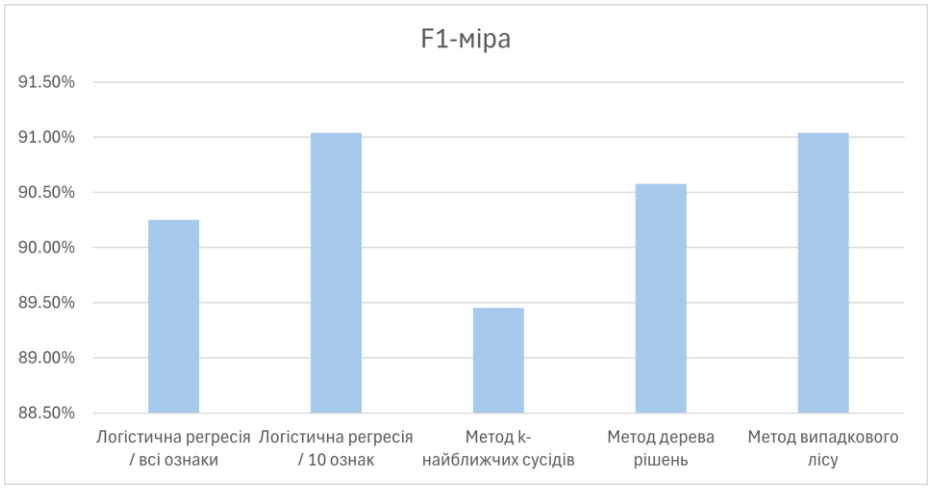
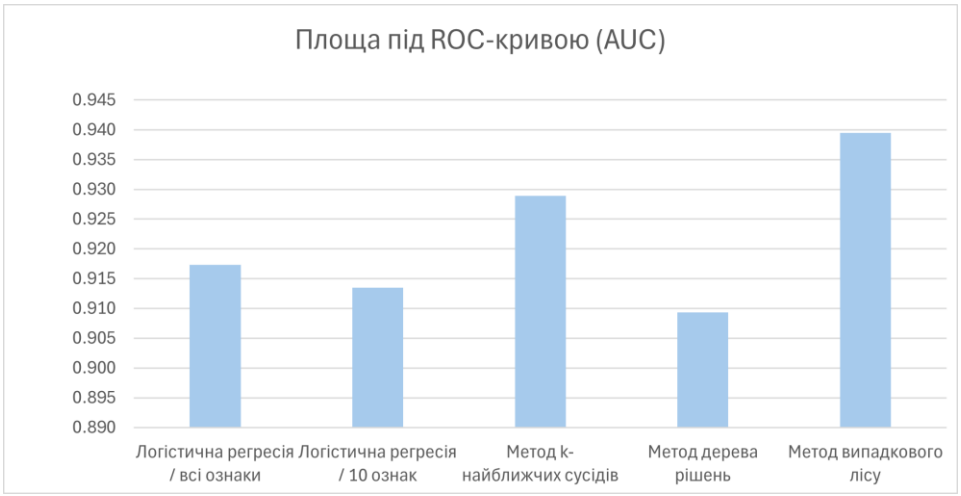


Рисунок 3.3.4. Діаграма AUC застосованих моделей.



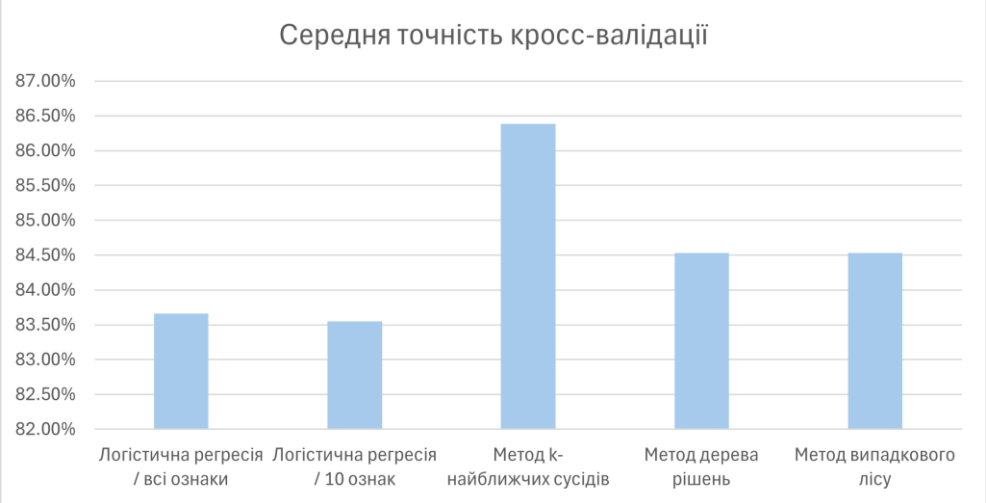
На рисунках 3.3.3 і 3.3.4 зображені діаграми F1-міри та значення площі під ROC-кривою для розглянутих нами моделей відповідно.

F1-міра – це середнє гармонійне значень влучності і чутливості моделі, тобто це оцінка, яка надає однакову вагу цим метрикам, і розглядає істинно позитивні та істинно негативні класифікації як однаково важливі. Можна побачити, що найкращі показники, тобто найближчі до 1, знову були показані моделями випадкового лісу та логістичної регресії (побудованої на неповному датасеті) – в обох випадках 91.04%.

У свою чергу значення AUC визначає здатність моделі розрізняти позитивні класи від негативних в більш загальному випадку – ROC-крива показує відношення між істинно-позитивними та хибно-позитивними рівнями для різних порогів класифікації від 0 до 1. Іншими словами, значення AUC

(площі під ROC-кривою) дає оцінку того, з якою ймовірністю моделі вдасться правильно класифікувати елемент. З рисунку 3.3.4 бачимо, що найкращі значення AUC були показані методами випадкового лісу (0.94) та k-найближчих сусідів (0.929).

Рисунок 3.3.5. Діаграма середньої точності кросс-валідації застосованих моделей.



Останньою було розглянуто оцінку ефективності моделей за допомогою кросс-валідації, тобто усереднену оцінку точності передбачень кожної з моделей у 9 різних поділах датасету на тренувальний і тестовий у співвідношенні 1:8. Як можна побачити на рисунку 3.3.5, на відміну від попередніх результатів, за цією метрикою на першому місці виявився метод k-найближчих сусідів з середньою точністю 86.38%, а наступними йдуть методи випадкового лісу і дерева рішень з однаковими значеннями 84.53%. Незважаючи на високі результати за попередніми оцінками, логістична регресія (в обох випадках) не виявилася такою ж ефективною на всьому датасеті, як на окремо розглянутій тестовій вибірці.

ВИСНОВКИ

У дослідженні були оглянуті підходи машинного навчання, які широко використовуються для вирішення прикладних задач в різноманітних сферах людського життя, а особливо пріоритетними є такі задачі у сфері медицини. Застосування комп'ютерних моделей і статистичних підходів до діагностування хвороб, звичайно, не здатне замінити експертний аналіз справжніх лікарів, але перспектива пришвидшення встановлення діагнозу є достатньою причиною звернутися до подібних методів, що і було зроблено в даній роботі.

Було розглянуто наступні методи машинного навчання: логістична регресія, метод k-найближчих сусідів, моделі дерева рішень та випадкового лісу. Перераховані моделі були застосовані для задачі діагностування ішемічної хвороби серця, що робить високу точність прогнозів, а особливо високу частку істинно позитивних і низьку частку хибно негативних класифікацій надзвичайно важливими факторами оцінки ефективності моделей.

Найпоширенішою з моделей, що застосовуються для бінарної класифікації, є логістична регресія – це, зазвичай, перший метод, до якого звертаються у вирішенні подібних задач. Але популярність використання одного підходу не означає, що він є універсально найкращим для будь-якого датасету, що і було визначено в результаті даного дослідження.

За першими трьома розглянутими метриками (точність, чутливість та F1-міра) логістична регресія показала одні з найкращих результатів, разом з класифікацією за допомогою випадкового лісу, а третьою за ефективністю стабільно виявилася модель дерева рішень. Водночас порівняння таких метрик, як AUC та середнє значення точності кросс-валідації, що краще демонструють ефективність моделей на повному обсязі інформації, розглянутої в дослідженні, показало дещо інший результат. За цими метриками було виявлено, що найкращими у точності передбачення наявності ішемічної хвороби серця в пацієнтів є методи випадкового лісу та *k*-найближчих сусідів.

З результатів дослідження можна скласти наступний рейтинг ефективності розглянутих моделей:

1. Випадковий ліс (найкращий результат за всіма метриками, другий найкращий результат кросс-валідації);
2. Дерево рішень (одні з найкращих результатів за всіма метриками, однакова середня точність кросс-валідації з випадковим лісом);
3. Логістична регресія та метод *k*-найближчих сусідів – обидва методи мали найкращі результати за окремими метриками, але недостатньо стабільні, щоб опинитися вище в цьому списку.

Отже, можемо зробити висновок, що, незважаючи на простоту в побудові та розумінні моделі логістичної регресії, вона не завжди є оптимальним вибором для вирішення задач бінарної класифікації, як було показано на прикладі задачі діагностування ішемічної хвороби серця на основі різноманітних даних про стан здоров'я пацієнтів. Проведений аналіз виявив, що у задачі діагностування медичних захворювань найбільш ефективною є ансамблева модель випадковий ліс, а другою за ефективністю виявився метод дерева рішень. Віддання переваги цим моделям машинного навчання при діагностуванні серцево-судинних захворювань може суттєво підвищити швидкість та точність їх виявлення, що грає важливу роль у наданні вчасної допомоги пацієнтам і зменшенні кількості передчасних смертей через хвороби цього класу.

ДЖЕРЕЛА

1. Hafsa Habehh, Suril Gohel1. Machine Learning in Healthcare. *Current Genomics.* 2021. XII. 22 (4). P. 291-300.
2. Rahul C. Deo. Machine Learning in Medicine. *Circulation.* 2015. XI. 132 (20). P. 1920-1930.
3. Sarang Narkhede. Understanding Confusion Matrix. *Towards Data Science*. 2018. V. URL: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62
4. Racheel Shaikh. Cross Validation Explained: Evaluating estimator performance. *Towards Data Science*. 2018. XI. URL: https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85
5. Kenney, J. F., Keeping, E. S. Root Mean Square. *Mathematics of Statistics.* 1962. P. 59-60.
6. Aniruddha Bhandari. Guide to AUC ROC Curve in Machine Learning: What Is Specificity? *Analytics Vidhya.* 2020. VI. URL: https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/
7. Chi-Square Statistic: How to Calculate It / Distribution. *Statistics How To.* URL: https://www.statisticshowto.com/probability-and-statistics/chi-square/
8. World Health Organization. Cardiovascular diseases. URL: https://www.who.int/health-topics/cardiovascular-diseases#tab=tab_1
9. Jason Brownlee. Logistic Regression for Machine Learning. *Machine Learning Mystery.* 2023. XII. URL: https://machinelearningmastery.com/logistic-regression-for-machine-learning/
10. Antony Christopher. K-Nearest Neighbor. *The Startup*. 2021. II. URL: https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4

11. Anshul Saini. What is Decision Tree? [A Step-by-Step Guide]. *Analytics Vidhya.* 2021. VIII. URL: https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/
12. Niklas Donges, Brennan Whitfield. Random Forest: A Complete Guide for Machine Learning. *Built In.* 2024. III. URL: https://builtin.com/data-science/random-forest-algorithm#what
13. Mbali Kalirane. Ensemble Learning in Machine Learning: Stacking, Bagging and Boosting. *Analytics Vidhya.* 2023. I. URL: https://www.analyticsvidhya.com/blog/2023/01/ensemble-learning-methods-bagging-boosting-and-stacking/
14. Aniruddha Bhandari. Feature Scaling: Engineering, Normalization, and Standardization. *Analytics Vidhya.* 2020. IV. URL: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/
15. Aniruddha Bhandari. Multicollinearity | Causes, Effects and Detection Using VIF. *Analytics Vidhya.* 2020. III. URL: https://www.analyticsvidhya.com/blog/2020/03/what-is-multicollinearity/
16. R. Detrano, A. Jánosi, W. Steinbrunn, M. Pfisterer, J. Schmid, S. Sandhu, K. Guppy, Stella Lee, V. Froelicher. International application of a new probability algorithm for the diagnosis of coronary artery disease. *American Journal of Cardiology.* 1989. VIII. 64 (5). P. 304-310.
17. Federico Soriano. Heart Failure Prediction Dataset. *Kaggle.* URL: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction/data

ДОДАТКИ

Додаток 1:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn import preprocessing
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, f1_score, roc_auc_score, roc_curve
from statsmodels.stats.outliers_influence import variance_inflation_factor

def sens(cnf):
    return cnf[0][0]/(cnf[0][0]+cnf[1][0])

def results(test, pred):
    conf_m = confusion_matrix(test, pred)
    print('Матриця помилок: \n', conf_m)
    print('Точність:', accuracy_score(test, pred))
    print('Влучність:', precision_score(test, pred))
    print('Чутливість', sens(conf_m))
    print('F1-міра:', f1_score(test, pred))

def roc_auc (model, testx, testy): # розрахунок AUC
    prob = model.predict_proba(testx)
    prob = prob[:, 1]
    return roc_auc_score(testy, prob)

def cross_val(model, X, Y): # кросс-валідація
    folds = KFold(n_splits=9)
    scores = cross_val_score(model, X, Y, cv=folds)
    print("Результати кросс-валідації: ", scores)
    print("Середнє значення результатів: ", scores.mean())

def calc_vif(X): # розрахунок VIF
    vif = pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i) for i in
range(X.shape[1])]
    return vif


def roc_c(model, model_name, test_x, test_y, file_name): # побудова ROC
кривої
    model_prob = model.predict_proba(test_x)
    model_prob = model_prob[:, 1]
    noskill_prob = [0 for _ in range(len(y_test))]

    noskill_fpr, noskill_tpr, _ = roc_curve(test_y, noskill_prob)
    model_fpr, model_tpr, _ = roc_curve(test_y, model_prob)

    plt.plot(noskill_fpr, noskill_tpr)
    plt.plot(model_fpr, model_tpr, marker='.', label=model_name)

    plt.xlabel('Хибно-позитивний рівень')
    plt.ylabel('Істинно-позитивний рівень')

    plt.legend(loc='lower right')

    plt.savefig(file_name)
    plt.show()
```

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

df.info()

features = ['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol',
'FastingBS', 'RestingECG', 'MaxHR',
            'ExerciseAngina', 'Oldpeak', 'ST_Slope']

categ_features = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina',
'ST_Slope']

label_enc = preprocessing.LabelEncoder()

for i in range(len(categ_features)):
    df[categ_features[i]] = label_enc.fit_transform(df[categ_features[i]])

x = df[features]  # ознаки, незалежні змінні
y = df.HeartDisease  # цільова, залежна змінна

print(calc_vif(x))

# дані для нормалізації
x_norm = df[features]


x = x.drop('RestingBP', axis=1)
x_norm = x_norm.drop('RestingBP', axis=1)

print(calc_vif(x))

scaler = preprocessing.MinMaxScaler()
scaler.fit(x_norm)
x_norm = scaler.transform(x_norm)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25,
random_state=1)
x_train_norm, x_test_norm, y_train_norm, y_test_norm =
train_test_split(x_norm, y, test_size=0.25, random_state=1)

log_reg = LogisticRegression(random_state=1, max_iter=1000)
log_reg_norm = LogisticRegression(random_state=1, max_iter=1000)

log_reg.fit(x_train, y_train)
y_pred = log_reg.predict(x_test)

log_reg_norm.fit(x_train_norm, y_train_norm)
y_pred_norm = log_reg_norm.predict(x_test_norm)


print('   Логістична регресія: ')
results(y_test, y_pred)
print('Площа під ROC-кривою (AUC):', roc_auc(log_reg, x_test, y_test))
print('\n------------\n')
cross_val(log_reg, x, y)
print('\n------------\n')


print('   Логістична регресія, нормалізовані дані: ')
results(y_test_norm, y_pred_norm)
print('Площа під ROC-кривою (AUC):', roc_auc(log_reg_norm, x_test_norm,
y_test_norm))
print('\n------------\n')
cross_val(log_reg_norm, x_norm, y)
```

```
print('\n------------\n')

# roc_c(log_reg, 'Логістична регресія', x_test, y_test, 'log_reg_roc.png')
# roc_c(log_reg, 'Логістична регресія, нормалізовані дані', x_test_norm,
y_test_norm, 'log_reg_roc_norm.png')
```

Додаток 2:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn import preprocessing
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, f1_score, roc_auc_score, roc_curve

def sens(cnf):
    return cnf[0][0]/(cnf[0][0]+cnf[1][0])

def results(test, pred):
    conf_m = confusion_matrix(test, pred)
    print('Матриця помилок: \n', conf_m)
    print('Точність:', accuracy_score(test, pred))
    print('Влучність:', precision_score(test, pred))
    print('Чутливість', sens(conf_m))
    print('F1-міра:', f1_score(test, pred))


def roc_auc (model, testx, testy): # розрахунок AUC
    prob = model.predict_proba(testx)
    prob = prob[:, 1]
    return roc_auc_score(testy, prob)

def cross_val(model, X, Y): # кросс-валідація
    folds = KFold(n_splits=9)
    scores = cross_val_score(model, X, Y, cv=folds)
    print("Результати кросс-валідації: ", scores)
    print("Середнє значення результатів: ", scores.mean())


def roc_c(model, model_name, test_x, test_y, file_name): # побудова ROC
кривої
    model_prob = model.predict_proba(test_x)
    model_prob = model_prob[:, 1]
    noskill_prob = [0 for _ in range(len(y_test))]

    noskill_fpr, noskill_tpr, _ = roc_curve(test_y, noskill_prob)
    model_fpr, model_tpr, _ = roc_curve(test_y, model_prob)

    plt.plot(noskill_fpr, noskill_tpr)
    plt.plot(model_fpr, model_tpr, marker='.', label=model_name)

    plt.xlabel('Хибно-позитивний рівень')
    plt.ylabel('Істинно-позитивний рівень')

    plt.legend(loc='lower right')

    plt.savefig(file_name)
    plt.show()


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

```
features = ['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol',
'FastingBS', 'RestingECG', 'MaxHR',
            'ExerciseAngina', 'Oldpeak', 'ST_Slope']

categ_features = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina',
'ST_Slope']

label_enc = preprocessing.LabelEncoder()

for i in range(len(categ_features)):
    df[categ_features[i]] = label_enc.fit_transform(df[categ_features[i]])


x = df[features]  # ознаки, незалежні змінні
y = df.HeartDisease  # цільова, залежна змінна


# дані для нормалізації
x_norm = df[features]


scaler = preprocessing.MinMaxScaler()
scaler.fit(x_norm)
x_norm = scaler.transform(x_norm)


x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25,
random_state=1)
x_train_norm, x_test_norm, y_train_norm, y_test_norm =
train_test_split(x_norm, y, test_size=0.25, random_state=1)


# ---------------- Метод k-найближчих сусідів ----------------

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=5, p=1)
knn_norm = KNeighborsClassifier(n_neighbors=5, p=1)

knn.fit(x_train, y_train)
y_pred = knn.predict(x_test)

knn_norm.fit(x_train_norm, y_train_norm)
y_pred_norm = knn_norm.predict(x_test_norm)



print('   Метод k-найближчих сусідів: ')
results(y_test, y_pred)
print('Площа під ROC-кривою (AUC):', roc_auc(knn, x_test, y_test))
print('\n------------\n')
cross_val(knn, x, y)
print('\n------------\n')


print('   Метод k-найближчих сусідів, нормалізовані дані: ')
results(y_test_norm, y_pred_norm)
print('Площа під ROC-кривою (AUC):', roc_auc(knn_norm, x_test_norm,
y_test_norm))
print('\n------------\n')
cross_val(knn_norm, x_norm, y)
print('\n------------\n')
```

```
# roc_c(knn, 'Метод k-найближчих сусідів', x_test, y_test, 'knn_roc.png')
# roc_c(knn_norm, 'Метод k-найближчих сусідів, нормалізовані дані',
x_test_norm, y_test_norm, 'knn_roc_norm.png')


# ---------------- Метод дерева рішень ----------------
from sklearn.tree import DecisionTreeClassifier

DT = DecisionTreeClassifier(max_depth=5, criterion="entropy", random_state=1,
splitter="random")
DT_norm = DecisionTreeClassifier(max_depth=5, criterion="entropy",
random_state=1, splitter="random")

DT.fit(x_train, y_train)
y_pred = DT.predict(x_test)

DT_norm.fit(x_train_norm, y_train_norm)
y_pred_norm = DT_norm.predict(x_test_norm)

print('   Метод дерева рішень: ')
results(y_test, y_pred)
print('Площа під ROC-кривою (AUC):', roc_auc(DT, x_test, y_test))
print('\n------------\n')
cross_val(DT, x, y)
print('\n------------\n')


print('   Метод дерева рішень, нормалізовані дані: ')
results(y_test_norm, y_pred_norm)
print('Площа під ROC-кривою (AUC):', roc_auc(DT_norm, x_test_norm,
y_test_norm))
print('\n------------\n')
cross_val(DT_norm, x_norm, y)
print('\n------------\n')

# roc_c(DT, 'Метод дерева рішень', x_test, y_test, 'DT_roc.png')
# roc_c(DT_norm, 'Метод дерева рішень, нормалізовані дані', x_test_norm,
y_test_norm, 'DT_roc_norm.png')

# ---------------- Метод випадкового лісу ----------------
from sklearn.ensemble import RandomForestClassifier


RF = RandomForestClassifier(random_state=1, max_depth=3, criterion="entropy")
RF_norm = RandomForestClassifier(random_state=1, max_depth=3,
criterion="entropy")

RF.fit(x_train, y_train)
y_pred = RF.predict(x_test)

RF_norm.fit(x_train_norm, y_train_norm)
y_pred_norm = RF_norm.predict(x_test_norm)

print('   Метод випадкового лісу: ')
results(y_test, y_pred)
print('Площа під ROC-кривою (AUC):', roc_auc(RF, x_test, y_test))
print('\n------------\n')
cross_val(RF, x, y)
print('\n------------\n')


print('   Метод випадкового лісу, нормалізовані дані: ')
results(y_test_norm, y_pred_norm)
```

```
print('Площа під ROC-кривою (AUC):', roc_auc(RF_norm, x_test_norm,
y_test_norm))
print('\n------------\n')
cross_val(RF_norm, x_norm, y)
print('\n------------\n')


# roc_c(RF, 'Метод випадкового лісу', x_test, y_test, 'RF_roc.png')
# roc_c(RF_norm, 'Метод випадкового лісу, нормалізовані дані', x_test_norm,
y_test_norm, 'RF_roc_norm.png')
```