**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**
**Інститут прикладного системного аналізу**

**Кафедра математичних методів системного аналізу**

«На правах рукопису»
УДК 51-76

«До захисту допущено»
Завідувач кафедри
_____ О.Л. Тимощук
«___»_____20__ р.

# Магістерська дисертація

**на здобуття ступеня магістра**

**зі спеціальності 124 Системний аналіз**

**на тему: «Методи інтелектуального аналізу даних для прийняття рішень щодо діагностування пацієнта»**

Виконав:
Студент(ка) ІІ курсу, групи КА-62м
Малашенко Дарина Вікторівна _____

Керівник:
к.т.н., доц.
Недашківська Н.І. _____

Рецензент:
к.ф.-м.н., доц.
Терещенко І.М. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ
2018

# РЕФЕРАТ

Магістерська дисертація: 111 с., 12 рис., 36 табл., 2 додатки, 46 джерела.

В роботі розглянуті і проаналізовані одні з найбільш вживаних з тих, що існують на даний момент, сучасних методів інтелектуального аналізу даних. Проведено дослідження відомих методів класифікації, а також ефективності використання ансамблів базових класифікаторів. Окрім цього, була запропонована дворівнева модель класифікації та доведена її ефективність на практичній задачі, а саме діагностиці пацієнта на предмет захворювання на Ішемічну хворобу серця та хронічну хворобу нирок.

Об’єктом дослідження є медичні показники (демографічні, симптоми, ЕКГ та результати обстежень) та їх значення для успішного діагностування захворювання.

Предметом дослідження є математичні моделі інтелектуального аналізу даних та їх ансамблів для проведення класифікації на основі статистичних даних.

КЛАСИФІКАЦІЯ, МАШИННЕ НАВЧАННЯ, ДІАГНОСТИКА, ПОПЕРЕДНЯ ОБРОБКА ДАНИХ, БЕГГІНГ, БУСТИНГ

# ABSTRACT

Master's thesis: 111 pages, 12 figures, 36 tables, 2 appendixes, 46 sources.

Theme: Data mining methods for diagnostic decision-making.

In this work one of the most widely used modern data mining methods were studied and analyzed. The research of known methods of classification, as well as the effectiveness of the use of ensembles of basic classifiers, has been carried out. In addition, a two-level model of classification was proposed and its effectiveness was proved on a practical task, namely diagnostics of the patient having heart and chronic kidney disease.

The subject of the study is medical indicators (demographic, symptoms, ECG and survey results) and their significance for successful diagnosis of the disease.

The subject of the study is the mathematical models of the intellectual analysis of data and their ensembles for the classification on the basis of statistical data.

CLASSIFICATION, MACHINE LEARNING, DIAGNOSTICS, DATA PRE-PROCESSING, BAGGING, BOOSTING

# ЗМІСТ

# ПЕРЕЛІК ПРИЙНЯТИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

SVM – Support Vector Machine

LDA – Linear Discriminant Analysis

DT – Decision Trees

NB – Naïve Bayes Classifier

KNN – K Nearest Neighbors

# ВСТУП

Медицина завжди була й залишається однією з найбільш важливих сфер діяльності для людства. А успіх або невдача лікування напряму залежать від вчасного та точного діагнозу.

За результатами дослідження Медичного інституту Джона Хопкінса [2], смертельні випадки, що сталися в результаті медичної помилки, стали третьою по частоті причиною смертності в США, обігнавши серцеві напади, хворобу Альцгеймера та діабет.

Важливий аспект, що відрізняє медичні дані від більшості інших, це те, що об’єктивність, точність, якість та своєчасність результатів є критично важливими і мають постійно ставитись під сумнів.

Але складнощі спричиняють не тільки самі дані, а також і процес їх збору. Усі дані звідкись беруться, але на жаль багатьох провайдерів послуг сфери охорони здоров’я, вони не завжди приходять з джерела, в якому інформацією керують бездоганно. Збір чистих, повних, точних і коректно відформатованих даних для використання в декількох системах – битва, що триває щодня для організацій, більшість яких програють.

Отже, основу медичного діагностування становить задача класифікації. Діагноз може бути зведено до проблеми з відображенням даних до одного з N різних результатів.

Перший розділ присвячено актуальності теми дослідження та аналізу проблем, що існують в медицині та шляхами їх вирішення, що можуть бути досягнуті завдяки використанню машинного навчання.

У другому розділі проводиться дослідження поширених методів попередньої обробки даних: нормування та очистки даних, заповнення пропусків і видалення дублікатів. Також в цьому розділі проводиться огляд моделей класифікації, переваг та недоліків кожної та підходів, що використовуються для оцінки їх роботи.

У третьому розділі наводяться результати застосування простих класифікаторів та їх ансамблів до обраної практичної задачі. Окрім цього, був проведений порівняльний аналіз отриманих результатів.

У четвертому розділі проводиться опис запропонованої дворівневої моделі класифікації та приведені результати її застосування до вирішення практичної задачі.

У п’ятому розділі був проведений маркетинговий аналіз стартап проекту на базі запропонованого в дисертації науково-технічного рішення з ціллю визначення принципової можливості його ринкового впровадження та можливих напрямків реалізації цього впровадження.

# РОЗДІЛ 1. АКТУАЛЬНІСТЬ РОЗВ'ЯЗАННЯ ЗАДАЧІ ДІАГНОСТУВАННЯ ЗАХВОРЮВАННЯ

## 1.1 Загальна проблема медичного діагностування та її особливості

### 1.1.1 Поняття діагностики

Діагностика (грец. diagnostikos — здатний розпізнавати) — розділ клінічної медицини, що вивчає зміст, методи та послідовні ступені процесу розпізнавання хвороб або особливих фізіологічних станів. У вузькому розумінні діагностикою називають безпосередньо процес розпізнавання хвороби та оцінки індивідуальних біологічних і соціальних особливостей суб'єкта, який включає цілеспрямоване медичне обстеження, тлумачення отриманих результатів та їх узагальнення у вигляді встановленого діагнозу. Розпізнавання хвороби здійснюється за її симптомами, як очевидними, так і встановленими за допомогою спеціальних досліджень, та ґрунтується на певних методологічних принципах. [1]

Враховуючи високу ціну людського життя, особливо у розвинених країнах, коректне використання сучасних методів діагностування, а також вміння сформулювати та обґрунтувати діагноз є надзвичайно важливим. Адже своєчасність та безпомилковість діагнозу дозволяє підвищити ефективність рекомендованого лікування, а в певних ситуаціях – врятувати людині життя.

### 1.1.2 Медичні помилки

Більшість людей погодиться, що правильно визначений діагноз – це надзвичайно важливо. Але тільки невелика частка організацій, що займаються охороною здоров'я, вимірюють величину та частоту помилок діагностування пацієнтів.

Відповідно до приблизних оцінок, помилки в діагностуванні трапляються в 10%-15% випадків, але якщо б ми захотіли точно оцінити це число, це визвало б багато складнощів. Перш за все, необхідно точно визначити яка частка цих помилок причинила серйозну шкоду пацієнтам.

Для прикладу, в Австралії кількість помилок становить 140 000 випадків на рік, з них – 21 000 завдають серйозної школи і спричиняють від 2000 до 4000 смертей.

За результатами дослідження Медичного інституту Джона Хопкінса [2], смертельні випадки, що сталися в результаті медичної помилки, стали третьою по частоті причиною смертності в США, обігнавши серцеві напади, хворобу Альцгеймера та діабет. На додачу, кожний сьомий пацієнт, що звертається до клініки, стає жертвою медичних помилок.

Ці цифри самі по собі є вже досить вагомою причиною, щоб зайнятися більш глибоким вивченням проблеми. Деякі опитування показали, що саме помилка діагностування – найбільший страх пацієнтів при зверненні до системи охорони здоров'я. Варто замислитись над тим, чому ці помилки взагалі мають місце. Їх природа може бути різною.

Розглянемо питання діагностики пацієнта. Перш за все, необхідно визнати: діагностування – задача дуже складна.

Є фактори пацієнта, такі як симптоми проявлення хвороби, а також варто враховувати складність системи охорони здоров'я з бар'єрами спілкування та розладженою системою догляду. Потім, маємо лікарські фактори впливу, такі як експертиза, доступ до даних про стан пацієнта, стрес та відволікаючі фактори.

Зрештою, існує понад 12 000 хвороб, якщо вірити Світовій Організації Охорони Здоров'я. Ніколи не чули про синдром жовтого нігтя або синдром чужородної руки? Велика кількість лікарів також. Занадто багато існує захворювань і їх кількість зростає з кожним роком.

Ми знаємо, що помилки виникають як внаслідок проблем з системою, так і з помилок в когнітивних міркуваннях. І ми знаємо, що головним фактором, який спричиняє понад 80% когнітивних помилок, є помилковий синтез інформації.

Не варто забувати, що лікарі – також люди. Вони так само зазнають впливу втоми, стресу, психологічних факторів тощо. А людям властиво помилятися. Хоча помилка у судженнях лікаря має, звичайно, набагато вищу ціну ніж, скажемо, прибиральниці. Змінити природу людини неможливо, але можна зменшити вплив людського фактору на процес прийняття рішення щодо діагнозу пацієнта, використовуючи сучасні технології штучного інтелекту.

### 1.1.3 Робота з медичними даними

Одну з найбільших складнощів в процесі роботи становлять дані: їх об’єм, структура та форма представлення.

З початку вісімдесятих років минулого сторіччя, швидко почали поширюватися так звані Медичні Інформаційні Системи (MIC) – системи автоматизації документообігу в лікувальних та профілактичних закладах, в яких поєднувалися система підтримки рішень в медицині, електронні медичні карти пацієнтів, дані досліджень в цифровому форматі, фінансова та адміністративна інформація тощо.

Можна уявити, на скільки стрімким є щорічний приріст даних в таких системах. Окрім надвеликого об’єму, записи в системі мають і інші особливості, що ускладнюють аналіз: пропуски в даних, неточності або суб’єктивність суджень.

І якщо ще на початку сторіччя вже ставало важко працювати з такими обсягами даних, то згодом навіть з’явилась потреба в появі нового терміну: «великі дані» або Big Data.

Відповідно до дефініції Oxford Dictionary, під Big Data розуміють надзвичайно великі масиви даних, які піддаються аналізу за допомогою обчислень для виявлення шаблонів, тенденцій та кореляцій, особливо пов'язаних з поведінкою людей та їх взаємодією. [2]

На 2012 рік, цифрові дані сфери охорони здоров’я за оцінками становили 500 петабайт і очікується, що досягнуть розміру в 25 000 петабайт в 2020 році, доповідає IBM TJ Watson Research Center . [3]

Очевидно, що без інноваційних високотехнологічних інструментів аналізу даних, ймовірність помилки є дуже високою, адже легко пропустити якийсь важливий фактор, що є критичним, але загубленим серед іншої, менш важливої інформації. Окрім цього, спеціально навчені алгоритми здатні відшукати зв’язки та паттерни, які раніше лишалися непоміченими.

Важливий аспект, що відрізняє медичні дані від більшості інших, це те, що об’єктивність, точність, якість та своєчасність результатів є критично важливими і мають постійно ставитись під сумнів.

Але складнощі спричиняють не тільки самі дані, а також і процес їх збору. Усі дані звідкись беруться, але на жаль багатьох провайдерів послуг сфери охорони здоров’я, вони не завжди приходять з джерела, в якому інформацією керують бездоганно. Збір чистих, повних, точних і коректно відформатованих даних для використання в декількох системах – битва, що триває щодня для організацій, більшість яких програють.

Більшість провайдерів сфери охорони знайомі із важливістю чистоти в приміщеннях лікарень, але можуть не до кінця усвідомлювати, на скільки велике значення має очистка даних. «Брудні» дані можуть швидко звести нанівець великий проект аналізу даних, особливо якщо взяти до уваги зведення інформації з різноманітних ресурсів, що можуть реєструвати записи у форматах, що різняться між собою.

Очищення даних дозволяє впевнитись в тому, що набір даних точний, правильний, послідовний, актуальний та ніяким чином не пошкоджений. В той час як більшість процесів очищення даних все ще виконуються вручну, деякі продавці IT послуг пропонують інструменти, що для очищення використовують правила логіки для порівняння та протиставлення, а також корегують великі набори даних. Подібне програмне забезпечення з кожним днем стає все більш ефективним та досконалим завдяки технікам машинного навчання, що швидко розвиваються, зменшуючи час та ресурси, необхідні для забезпечення високого

рівня точності та цілісності в сховищах медичних даних. Приклад таких інструментів можна знайти, наприклад, в джерелі [4]

Окрім цього, згідно з [5], для багатьох лікарів процес вводу даних в комп’ютер спричиняє сильний дискомфорт та стрес, сповільнює роботу або й взагалі шкодить спілкуванню з пацієнтом.

Для вирішення цієї проблеми, деякі лікарні в США розглядали можливість використання розробки Amazon – Alexa (цифровий персональний асистент, що активується голосом [6], що могла б записувати та транскрибувати розмову з пацієнтом без втручання лікаря. Однак, виявилось, що це програмне забезпечення не відповідає усім необхідним нормам американського законодавства щоб працювати з особистими даними пацієнтів [7].

Скажімо, ми вже маємо розумний алгоритм штучного інтелекту, що здатний виявити закономірності в великих обсягах даних. Тут може з’явитися інша проблема. Для якісної роботи подібного алгоритму необхідна достатня навчальна вибірка, що може становити сотні тисяч об’єктів. Однак, дослідники можуть не мати доступу до такої кількості інформації, адже деякі дані є закритими або ж взагалі – нецифрованими. Через це дуже часто спроби досліджень застрягали на етапі пошуку інформації в старих фармакологічних архівах або записах приватних лікарів. Це непроста дилема. Адже необхідно поважати конфіденційність пацієнтів, але в той самий час використання «закритих» даних може врятувати багато життів в майбутньому.

За словами Чарлі Дейві, керуючого директора інноваційної групи NHS (Національна служба охорони здоров'я Великобританії), «Точне лікування – це чудово, але вчасне запобігання – набагато важливіше. Якщо б ми могли володіти інформацією про фактори ризику людей, нам би було під силу виконувати обслуговування до того, як проблеми з’являться взагалі. Але це має бути зроблене в взаємодії з суспільством, щоб впевнитися, що ми не створюємо життя під наглядом Великого Брата». [8]

## 1.2 Методи інтелектуального аналізу даних в медицині

Індустрія інформаційних технологій в сфері охорони здоров’я змінюється щодня і організації, що не виходили на ринок впродовж деякого часу, можуть бути не в курсі останніх інноваційних пропозицій від амбіційних нових «гравців» на полі.

Ринок технологій інтелектуального аналізу даних може бути поділений на 4 головні категорії, базуючись на рішеннях, які вони намагаються забезпечити. [8]

Внаслідок цього стрімкого розвитку, новий урожай наукових досягнень дозріває в області машинного навчання, даруючи сфері охорони здоров’я можливість скористатися цілою низкою революційних інструментів, що використовують обробку природньої мови, розпізнавання образів та «глибоке» навчання аби забезпечити більш надійне піклування.

Звичайно, індустрії ще є над чим працювати перед тим, як машинне навчання та штучний інтелект будуть в змозі відповідати потребам сучасної медицини, але варто зазначити, що інноваційні технології вже залишають свій слід в середовищі аналізу медичних даних. Ось декілька першочергових ініціатив і досліджень, що справді інтригують і наразі важко працюють над удосконаленням своїх інструментів.

### 1.2.1 Аналіз зображень та патологій

Покращення аналізу зображень та патологій за допомогою машинного навчання викликає особливий інтерес для організацій охорони здоров'я, які в іншому випадку залишили б велику кількість великих даних викинутими на вітер.

Машинознавство може доповнити навички радіологічних працівників, виявляючи більш жорсткі зміни при скануванні зображень швидше, що потенційно може призвести до більш ранньої та більш точної діагностики.

Ряд виробників технологічної індустрії вже почав вкладати значні кошти в проекти аналізу та аналізу зображень. [9]

IBM Watson розгортає клінічну службу огляду зображень, яка допомагає ідентифікувати стеноз аорти [10], тоді як корпорація Майкрософт спрямувала свої зусилля на фенотипування біомаркерів із зображенням, щоб доповнити свої доробки у сфері дослідження раку.

Академічні установи також входять на перший рівень сучасного розпізнавання образів. У Індіанському університеті та Університеті Пердью, штат Індіанаполіс, дослідники дають волю алгоритмам машинного навчання на зображеннях патологій, щоб прогнозувати частоту рецидивів гострого мієлогенного лейкозу. У невеликому дослідженні, опублікованому раніше цього року, один алгоритм міг ідентифікувати пацієнтів, які можуть захворіти повторно зі 100-відсотковою точністю.

І в Стенфордському університеті інструменти для машинного навчання показали кращі результати, ніж спеціалісти, при розрізненні двох видів раку легенів. Комп'ютер також перевершив своїх людей-суперників в прогнозуванні тривалості життя пацієнта.

Тим часом дослідники Google вже перевищили точність патологів, які вивчали зображення метастазированої тканини раку молочної залози, зменшуючи помилки типу false negative до однієї чверті від частки пацієнтів. [11]

### 1.2.2 Аналіз природньої мови та даних типу free-text

Від електронних медичних карт(EHR) до МРТ, неструктуровані дані є скрізь у галузі охорони здоров'я – створені навмисне або випадково.

PDF-зображення факсимільних лабораторних звітів, голосові записи споживчих взаємодій та вхідні дані електронної пошти з вільним текстом становлять значні труднощі для традиційних інструментів аналізу, однак машинне навчання пропонує новий спосіб видобутку корисного значення з цих джерел даних.

Використовуючи алгоритми обробки природних мов (NLP), алгоритми машинного навчання можуть перетворювати зображення тексту в документи, що піддаються редагуванню, витягати семантичне значення з цих документів або обробляти пошукові запити, написані звичайним текстом, для повернення точних результатів.

Медичний центр Анни Арундел (AAMC) використовує інтерфейс природної мови, подібний до будь-якого з широко відомих пошукових систем Інтернету, щоб дозволити користувачам отримувати доступ до даних та отримувати достовірні результати. [12]

З інтерфейсом, надбудованим над EHR організації, "ви можете запитати щось на кшталт: скільки відмов було у нас в минулому місяці? або скільки пацієнтів ми маємо з певним діагнозом раку? ", - каже Девід Лер, виконавчий директор аналітичного підрозділу в AAMC.

"Він спирається на джерела інформації, що включає дані про відмови, а потім надає відповідне джерело користувачеві, щоб відповісти на це питання. Якщо він не може знайти конкретну відповідь, він все одно може визначити, де може бути відповідь, щоб користувач міг глибше дослідити дані та з'ясувати те, що він хоче знати ".

НЛП також можна поставити на роботу зі збором інформації формату free-text, такої як неструктуровані клінічні нотатки в EHR, статті академічних журналів, дослідження задоволеності пацієнтів або інші описи.

В одному проектів Великої Британії дослідники застосували інструменти для обробки природної мови для оцінок лікарів своїх однолітків, і виявили, що інструменти NLP узгоджуються з оцінками вмісту людиною в 98 відсотків випадків. [13]

"Складність інформації з відкритим текстом означає, що, на відміну від оцінок, отриманих від перевіреного пацієнтом досвіду та результатів, ці слова не можуть бути просто додані для створення розуміння та сенсу", пояснили дослідники. "Таким чином, завдання розуміння таких даних історично завершено вручну кваліфікованими якісними аналітиками".

Але НЛП може значно скоротити цей процес, дозволяючи провайдерам органічно взаємодіяти з споживачами та діловими партнерами, не втрачаючи часу на обробку даних після цього.

### 1.2.3 Підтримка прийняття рішень і прогнозування

Здатність витягати значення з великих обсягів вільного тексту також має вирішальне значення для підтримки клінічних рішень та прогнозування - ще однієї області, де розробки машинного навчання освітлюють дорогу.

Швидке виявлення та усунення ризиків може значно покращити результати для пацієнтів з будь-якою кількістю серйозних станів, як клінічних, так і поведінкових.

Наприклад, організація "Beacon Health Options" використовує технологію машинного навчання для точного налаштування його можливостей для розшарування ризику, що дозволяє керівникам випадків більш активно контактувати з пацієнтами з високим рівнем ризику та краще координувати процес піклування.

З клінічної точки зору, дослідники Медичної Школи Ікани в Маунт Синай (ISMMS) використовують алгоритми для розрізнення двох серцевих захворювань з дуже подібними презентаціями.

"Наш підхід демонструє перспективну тенденцію застосування автоматизованих алгоритмів як методів точної медицини для посилення діагностики, яку виконує лікар", - зазначив автор дослідження Джоел Дадлі,

доктор філософії, директор Інституту охорони здоров'я нового покоління та директор Центру медико-біологічної інформатики ISMMS. [14]

Каліфорнійський університет у Центрі цифрових медичних інновацій у Сан-Франциско (CDHI) та GE Healthcare створюють бібліотеку алгоритмів прогнозної аналітики для пацієнтів з травмою, намагаючись прискорити надання швидкої допомоги. [15]

Дослідники Медичної школи Уейл Корнелл та Університету Карнегі-Меллона також використовують машинне навчання, щоб виявити варіації моделей витрат та створити клінічні шляхи для управління хронічними захворюваннями, потенційно знижуючи витрати та покращуючи результати для пацієнтів з довгостроковими потребами.

Клінічні та фінансові програми - це лише початок машинного навчання. Кібер-безпека та конфіденційність пацієнтів є критично важливими для кожного постачальника медичних послуг, а інтелектуальні алгоритми можуть бути найкращим способом заповнити прогалини в їх захисті. Здатність патрулювати периметри безпеки з більшою чутливістю та чуйністю, ніж людина, може стати важливою перемогою для галузі.

"Застосування засобів машинного навчання та рішень штучного інтелекту для IT-інфраструктури охорони здоров'я буде швидко перетворювати сектор шляхом створення механізму, за допомогою якого постачальники та постачальники можуть захищати дані клінічного здоров'я, які зберігаються локально або в хмарі", - написав Джеймс Скотт, старший Співробітник Інститут Технологій Критично Важливих Інфраструктур в недавньому звіті.

### 1.2.4 Використання методів штучного інтелекту в діагностуванні захворювань

Від офісів технологічної компанії до університетських дослідницьких лабораторій і навіть простих лікарень, діагностичні програми машинного

навчання швидко переходять від теоретичної розробки до практичного застосування. Важко переоцінити трансформаційний потенціал подібних технологій: миттєва «друга думка», домашній інструмент для раннього виявлення та найрозумніший і точний діагност за всю історію людства. Машинне навчання як інструмент діагностики створить неймовірну ефективність та економію коштів для пацієнтів, лікарів та лікарень. Це стимулюватиме нових переможців у секторі технологій та охорони здоров'я. І найголовніше - це рятуватиме життя.

Медична література описує всі хвороби традиційно: спочатку діагноз (ім'я захворювання), а потім його опис, зокрема, клінічні прояви (симптоми / ознаки). Ось тут ховається найбільша проблема, адже багато різних захворювань проявляються однаковими або подібними симптомами та ознаками (біль у грудях, біль у животі, головний біль, лихоманка, артеріальна гіпертонія тощо). Отже, справжня робота лікаря полягає в клінічних міркуваннях, що рухаються не від діагнозу до ознак, як у підручниках, монографіях та лекціях, а навпаки - від виявлених ознак хворого через диференційоване діагностування всіх можливих захворювань з однаковими або подібними проявами та встановлення найбільш вірогідного діагнозу. Звичайна діагностична методика припускає, що кожен лікар тримає в своїй голові каталог всіх захворювань, всіх їх проявів і всіх критеріїв для швидкої та точної диференціальної діагностики. А також вважається, що лікар має необмежений час для інтенсивного аналізу проблеми кожного пацієнта.

Отже, основу медичного діагностування становить задача класифікації. Діагноз може бути зведено до проблеми з відображенням даних до одного з N різних результатів. У деяких випадках N = 2: завдання полягає лише в тому, щоб на основі даних (наприклад, рентгенограми або ЕКГ) визначити, має пацієнт конкретну хворобу (1) чи ні (0). Варто зазначити, що перевагу, зазвичай, надають алгоритмам, що на виході мають не тільки відповідь, до якого класу належить той чи інший об’єкт, а й ймовірності належності об’єкта кожному з класів.

Саме тут з'являються суперкомп'ютери, такі як, наприклад, IBM Watson, який допомагає лікарям діагностувати пацієнтів у Меморіальному центрі раку Стелла Кеттерінгу в Нью-Йорку.

Watson - це комп'ютер, який робить те, що називається "когнітивне обчислення", ефективно використовуючи величезну кількість інформації, наприклад, симптоми, які ви помітили, дані з вашого фітнес-браслету, нещодавнього аналізу крові, інформації про ваші гени, а також про вашу попередню історію хвороби. Потім він об'єднує і порівнює це з широким спектром даних з попередньої літератури та минулих медичних випадків. і за допомогою алгоритмів він може обробляти цю інформацію, щоб спробувати виробити найбільш логічний діагноз.

Watson все ще знаходиться на ранній стадії розробки і, як і раніше, потребує людського керівництва, але, коли комп'ютер стає все більш витонченим, він може одного дня порівнятися з або навіть перевершити здатність лікаря ставити діагноз та рекомендувати лікування. [16]

У пошуках побудови більш ефективного інструменту для виявлення раку шкіри команда дослідників у Стенфорді під керівництвом Себастьяна Трюна, одного з найвидатніших піонерів машинного навчання у світі, навчила нейронну мережу, використавши навчальну вибірку з майже 130 000 зображень шкірних уражень, всі категорії яких були класифіковані дерматологами як дві тисячі різних захворювань.

У червні 2015 року Трюн та його колеги поставили нейронну мережу на випробування, попросивши діагностувати 14 000 зображень шкірних уражень, необов'язково підтверджених біопсією. Двох сертифікованих дерматологів попросили зробити те ж саме. Результати були вражаючими: нейронна мережа отримала правильну відповідь у 72% випадків; дерматологи – у 66%. Потім команда розширила дослідження, залучивши 25 дерматологів проти нейронної мережі. Вони переглянули вибірку «золотого стандарту» з 2000 образів, підтверджених біопсією. Команда підвела підсумок у своїй статті, опублікованій в Nature в січні: "У кожному тесті нейронна мережа перевершила експертів-дерматологів".

Нейронна мережа Трюна обіцяє допомогти ранньому виявленню раку шкіри на двох основних фронтах. По-перше, дерматологи діагностують ураження на погляд, це трудомісткий процес, який може поставити під загрозу ґрунтовність. Мережа Трюна може не тільки запропонувати миттєву другу точку зору, але й потенційно дозволити делегування первинної оцінки, обмежуючи кількість справ лікаря лише до негайних або тих, що потребують більш глибокого аналізу.

По-друге, для того, щоб дерматолог дійсно оцінив пацієнта, цей пацієнт повинен прийти в клініку. Через відсутність терміновості або відсутності доступу до пацієнта цей бар'єр може бути однією з найбільших причин смертності від раку шкіри. Трюн вважає, що головне використання мережі - це додаток для смартфонів, який дозволить людям робити фотографії своєї шкіри в будь-якому місці і в будь-який час, коли виникають проблеми, а потім негайно отримувати подальші інструкції. Поки що мережа намагалася точно оцінити зображення низької якості - різноманітні кути, погане освітлення, введення в оману тіней тощо.  Вчений вважає, що це вміння можна відносно швидко впровадити і з його допомогою значно підвищити процент діагностування на ранніх стадіях і, як результат, процент врятованих пацієнтів.

Дослідники зі Стенфорда опублікували статтю, в якій повідомляється, що 34-лінійна згорткова нейронна мережа, яку вони розробляють, "перевищує продуктивність сертифікованих кардіологів у виявленні широкого спектру серцевих аритмій з електрокардіограми, записаних за допомогою одноконтактного монітора, що кріпиться на тілі" [17]

Бостонська біофарматична компанія Берг використовує штучний інтелект для дослідження та розробки діагностики та терапевтичного лікування в багатьох областях, включаючи онкологію. Поточні дослідницькі проекти, що ведуться, включають лікарські дослідження для внутрішньовенного лікування пухлини та виявлення та лікування раку передміхурової залози.

Cognoa, запуск Digital Health на базі Palo-Alto, створив платформу для машинного навчання, здатну діагностувати затримки у розвитку дітей, використовуючи лише інформацію та відео, які дистанційно надаються

батьками. Він вже оцінив 300 000 дітей, і компанія тільки завершила черговий раунд фінансування під час підготовки до схвалення FDA.

Інші приклади включають Google DeepMind Health, який минулого року оголосив про співпрацю з декількома партнерами у Великобританії, в тому числі з лікарнею Moorfields Eye Hospital в Лондоні, в якій вони розробляють технології для лікування макулярної дегенерації у очах, що піддаються віковим змінам.

У сфері захворювань, пов'язаних із мозком, таких як депресія, проект Oxford P1vital® Прогнозування реакції на лікування депресії (PReDicT) використовує прогнозну аналітику для діагностики та лікування, з загальною метою створення комерційно доступної емоційної акумуляторної батареї для використання в клінічних установках. [18]

Дослідницька група з медичного центру Бек-Ізраїлю Діаконес та Гарвардської медичної школи навчила нейронну мережу для інтерпретації патологічних зображень для ідентифікації пухлин. У дослідженнях діагностичний показник успішності мережі становив 92%, в порівнянні з 96% для патологів, які брали участь. І все ж, результати перспективні: в поєднанні, результати машини та людини досягли точності до 99,5%.

Дослідники з медичного центру Langone в Нью-Йоркському університеті створили алгоритм машинного навчання для діагностики ПТСР (Посттравматичний Стресовий Розлад), лише прослуховуючи мову людини. Його діагностичний успіх склав 77%. Подібні підходи, що використовують лише голос, також виглядають перспективними при виявленні хвороби Альцгеймера.

Дослідники з Лондона опублікували статтю, в якій використовуються дані з ADNI (Alzheimer's Disease Neuroimaging Initiative) для навчання 3-х шарової нейронної мережі з єдиним згортковим шаром, який може передбачити, чи є МРТ сканування здоровим мозком, мозком з легким когнітивним порушенням або мозком з хворобою Альцгеймера. [19]

Дослідниця Каліфорнійського Університету Лос-Анджелесу Жасмін Чжоу розробила перший аналіз крові, здатний діагностувати рак. Він використовує алгоритм машинного навчання для виявлення пухлинної ДНК у зразках крові та визначення місця в тілі, з якого він надходить. До теперішнього часу вона

зосередила увагу на трьох видах раку - грудях, легенях та печінці. Успішно діагностовано ранню стадію раку в 80% випадків.

Три роки тому відбувся конкурс Kaggle, учасники якого намагалися класифікувати зображення очей на одне з 5 класів (без діабетичної ретинопатії, легкої, помірної, важкої та проліферативної). Рішення, що отримало перемогу, використовувало комбінацію розріджених згорткових мереж та Random Forest, щоб зробити прогноз з пари зображень (лівого та правого ока) до результату.

Всі ці вражаючі результати є лише попередником того, наскільки точними та продуктивними стануть діагностичні можливості машинного навчання в майбутньому. Чим більше даних дослідники подають алгоритмові для навчання, чим більше інструменти починають прийматися лікарнями та лікарями, тим більше "навчальні виборки" зростатимуть, і в свою чергу, точність діагностики покращиться. Розмовляючи з журналом The New Yorker, вчений галузі комп’ютерних наук Джефрі Хінтон пояснює, чому машинне навчання має на меті перевершити людську діагностичну здатність: "Якщо [лікар-людина пропускає] щось, і в пацієнта розвивається рак через п'ять років, немає систематичної процедури, яка говорить про те, як це виправити . Але ви можете побудувати систему, щоб навчити комп'ютер досягти саме цього ».[20]

Штучний інтелект має перспективи трансформації сфери охорони здоров'я, починаючи з розширення можливостей персональної медицини і закінчуючи підвищення ефективності роботи та менеджменту витрат, що базується на прогнозуванні. Проте, діагностика здається найбільш неминучим застосуванням, а руйнівні наслідки сейсмічні. І технічні гіганти, які домінують в машинному навчанні як сервісний ринок - в даний час це IBM, Microsoft і Amazon - будуть добре закріплені, оскільки лікарні та страхові компанії вимагають інфраструктури, що здатні працювати з великими даними.

При цьому, не варто забувати про серйозні питання, які залишаються, щодо ефективності інтегрування машинного навчання в діагностичний процес. Якими будуть юридичні наслідки, якщо алгоритм відповідальний за неправильне діагностування? Чи вони виправдають своє існування як опора для лікарів, компрометуючи їх уважність та притуплюючи їх навички діагностування? І чи

варто медикам довіряти діагнозу, що був отриманий алгоритмом, якщо вони не розуміють, чому машина дійшла такого висновку? Проте незалежно від такої кількості питань, на які ми ще не маємо відповіді, машинне навчання, як інструмент діагностики, демонструє свій потенціал у справі рятування життів.

## Постановка задачі і висновки до розділу

Медична сфера завжди була і залишається одним із найбільш важливих напрямів, на розвиток яких людство направляє багато зусиль. Це, перш за все, зумовлено високою ціною помилки, адже мова йде про людські життя.

Лікувати хвороби – це чудово, але успіх лікування на 90% залежить від вчасного та коректного діагностування. Довгий час ця відповідальність лежала на плечах лікарів. Але навіть найбільш досвідчені спеціалісти – це, перш за все, люди. Людина не може знати всього, а й на додачу існує ймовірність когнітивної помилки через стомленість, стрес чи неуважність.

Окремий ряд проблем викликають медичні дані: їх об’єми, якість та формат представлення. Більше 80% когнітивних помилок викликані якраз неправильно проінтерпретованими даними.

Тут на допомогу приходять новітні технології, що використовують штучний інтелект. Одними з розробників передових технологій, відомих у сфері, є IBM, Microsoft та Amazon.

Згідно з останніми статистичними даними, у наш час серцево-судинні хвороби – причина понад 17 мільйонів забраних життів кожного року. За словами експертів, ця кількість може зрости до 23 мільйонів осіб вже до 2030го року.

Зокрема, в Україні цифри також не є втішними: 68% населення країни страждає на серцево-судинні захворювання.

Статистика по серцевим хворобам і помилкам в їх діагностуванні наведена в роботі [21]

На меті цієї науково-дослідної роботи стоїть розробка і дослідження методів інтелектуального аналізу даних для діагностування Ішемічної хвороби серця.

Для досягнення цієї мети потрібно розв’язати наступні задачі:

- Провести огляд і аналіз базових методів, що існують на даний момент, шляхом вивчення наукових джерел; визначити переваги та недоліки кожного.
- Дослідити найбільш поширені ансамблі класифікаторів та ознайомитися з новітніми розробками в цій сфері.
- Ознайомитися та виконати опис метрик, що застосовуються для оцінки якості результатів роботи методів бінарної класифікації, визначити переваги та недоліки кожної.
- Виконати порівняльний аналіз результатів і обрати найкращі базові моделі і методи агрегування для вирішення обраної практичної задачі.
- Запропонувати модифікацію методу, яка на реальних даних, а саме: діагностиці ішемічної хвороби серця, дозволить підвищити значення критеріїв оцінки якості, надаючи перевагу повноті як найбільш важливій метриці в задачах медичного аналізу.

# РОЗДІЛ 2. ВИБІР ТА ОПИС МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ЗАДАЧІ МЕДИЧНОЇ КЛАСИФІКАЦІЇ

## 2.1 Попередня обробка і аналіз даних

Попередня обробка даних ґрунтується, здебільшого, на методах інтелектуального аналізу даних. Це одна з технік інтелектуального аналізу даних, що передбачає перетворення необроблених «сирих» даних в більш доступний для розуміння та подальшої роботи формат. Реальні дані часто неповні, суперечливі та/або містять багато помилок та пропущених значень. Попередня обробка даних готує вихідні дані для подальшої обробки.

Більшість методів багатовимірного статистичного аналізу даних, такі як факторний, кластерний, регресійний аналіз і багато інших, вимагають відсутності пропусків в аналізованих даних. Однак в реальних даних пропуски зустрічаються дуже часто. На це може бути багато причин. Відсутність того чи іншого значення в даних може означати, що дані були недоступні або непридатні в той момент, або подія, результатом якої мало стати отримане значення, просто не відбулася. Можливо, людина, що вносила дані, не знала правильного значення або ж просто припустилася помилки. Якщо ж це медичні дані, респондент міг відмовитися надавати ту чи іншу інформацію особистого характеру.

Аналіз даних з пропусками, так само як і просте виключення з розгляду об’єктів з пропусками, дає неповну картинку, а висновки, отримані таким шляхом, не є надійними, а часом навіть абсолютно помилковими. Не має потреби говорити, що рішення, прийняте, спираючись на результати даного аналізу, може мати фатальні наслідки.

Дані, що є непригідними для подальшого аналізу та потребують попередньої обробки, називають «брудними». З поняттям «брудних даних» можна ознайомитися в роботі [22], де представлена таксономія 33 типів брудних даних і також запропонований набір методів, що допомагають запобігти або розпізнати і очистити дані. В роботі наведені різноманітні типи «брудних даних», серед них виділено такі групи:

– такі, що можуть бути автоматично виявлені і очищені;
– такі, появу яких можна попередити;
– такі, які є непридатними для автоматичного виявлення і очищення;
– такі, появі яких неможливо запобігти.

Тому важливо розуміти, що спеціальні засоби очищення можуть впоратися не з усіма видами брудних даних.

Ознайомимося більш детально з найпоширенішими видами брудних даних:

– пропущені значення;
– дублікати даних;
– шуми і викиди.

### 2.1.1 Пропущені значення (Missing Values)

Деякі значення даних можуть бути пропущені у зв'язку з тим, що:

– дані взагалі не були зібрані (наприклад, при зборі медичних даних пацієнти відмовилися надати деякі особисті дані);
– певні ознаки не можуть бути використані для характеристики певних об'єктів;
– людина, що вносила дані, припустилася помилки;
– на конкретний момент часу дані були недоступні або непридатні для збору.

Існують наступні шляхи вирішення проблеми:

– Ігнорування пропущених значень - це найпростіший і ефективний спосіб для обробки відсутніх даних. Але цей метод не повинен виконуватися в той час, коли кількість відсутніх значень є величезним або якщо шаблон даних пов'язаний з невизнаним основним коренем формулювання проблеми.

- Заповнення пропущених значень вручну - це один із найкращих методів. Але є одне обмеження, що коли існує великий набір даних, і відсутні значення є значними, то цей підхід є неефективним, оскільки завдання стає занадто трудомістким.
- Заповнення за допомогою обчислених значень - відсутні значення також можуть бути зайняті шляхом обчислення середнього, режиму або медіани спостережуваних заданих значень. Інший метод може бути прогностичними значеннями, які обчислюються за допомогою будь-якого алгоритму машинного навчання або глибокого навчання. Але один недолік цього підходу полягає в тому, що він може спричинити зміщення в даних, оскільки розрахункові значення не є точні щодо спостережуваних значень.
- Заміна пропущених значень на певні значення, що витікають з інших показників ознак об’єктів у вибірці.

### 2.1.2 Дублювання даних (Duplicate Data)

Набір даних може включати дублікати. Дублікатами називаються об’єкти, значення яких для всіх атрибутів співпадають.

Наявність дублікатів в наборі даних може бути способом підвищення значущості деяких записів. Така необхідність іноді виникає для особливого виділення певних записів з набору даних. Однак в більшості випадків, дублікати є результатом помилок при створені даних.

Існує два способи усунення проблеми з наявністю дублікаті:

1. Перший спосіб пропонує видалити всю групу об’єктів, що містить дублікати. Цей варіант використовується в тому випадку, якщо наявність дублікатів викликає недовіру до інформації, повністю її знецінює.

2. Другий варіант полягає в заміні групи дублікатів на один унікальний запис.

### 2.1.3 Шуми і викиди

Викиди – окремі ознаки об’єктів або цілі записи, що значно вибиваються із загальної картини.

Неточність одиничного вимірювання та точки даних може бути прийнятною та пов'язана з властивою технічною помилкою вимірювального приладу. Отже, очищення даних повинно зосереджуватися на тих помилках, які виходять за межі малих технічних відхилень і що є основним зрушенням у межах або за межами розподілу вибірки. У свою чергу, очищення даних повинно базуватися на знанні технічних помилок та очікуваних діапазонах нормальних значень.

Деякі помилки заслуговують більш пріоритетного ставлення, але які саме з них є найважливішими, дуже сильно залежить від типу досліду. У більшості клінічних досліджень помилки, які потребують видалення за будь-яку ціну, включають відсутність статі, недостовірність щодо статі, дати народження чи помилку експертизи, дублювання або злиття записів, а також біологічно неможливі результати. Наприклад, в дослідженнях з харчування помилки дат призводять до вікових помилок, що, в свою чергу, призводить до помилок у підрахунку ваги за віком і, відповідно, до неправильної класифікації суб'єктів як недоліків чи надмірної ваги.

Помилки статі та дати особливо важливі, оскільки вони забруднюють похідні змінні. Визначення пріоритетів має важливе значення, якщо дослідження є під тиском часу або якщо ресурси для очищення даних є обмеженими.

З проблемою шуму і викидів в сфері аналізу даних стикаються дуже часто. Викиди можуть як являти собою окремі спостереження, так і бути об'єднаними в якісь групи. Завдання спеціаліста – не тільки їх виявити, але і оцінити ступінь

їх впливу на результати подальшого аналізу. Якщо викиди є інформативною частиною набору даних, що аналізується, то використовують робустні методи і процедури.

Досить поширеним є підхід, що передбачає проведення двоетапного аналізу – з викидами та без них і порівняння отриманих результатів.

Різні методи інтелектуального аналізу даних мають різну чутливість до викидів, цей факт необхідно враховувати при виборі методу. Також деякі інструменти Data Mining мають вбудовані методи очищення від шумів і викидів.

Дуже наглядним способом виявлення викидів є візуалізація. Приклад наявності викидів зображений на діаграмі розсіювання на рис. 2.1. Вирізняються кілька спостережень, які значно відрізняються від інших (які перебувають на великій відстані від більшості спостережень).

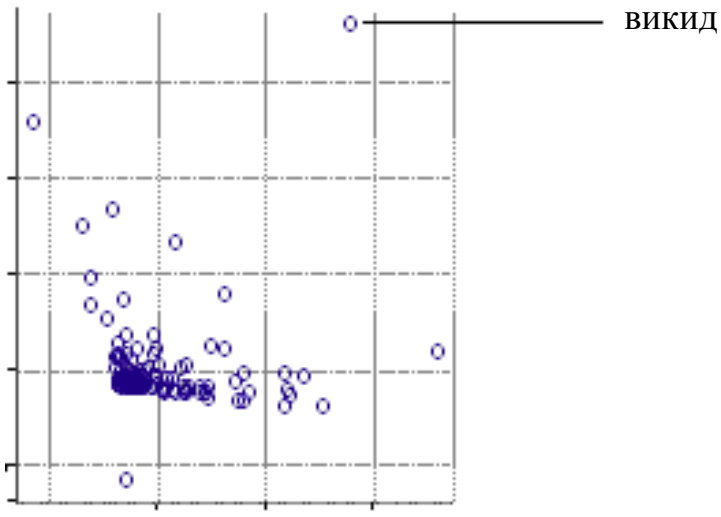



Рисунок 2.1 – Приклад набору даних з викидами

Окрему проблему полягає в помилкових даних, що були внесені невірно, але знаходяться в межах очікуваного діапазону, тобто не можуть бути помічені як викиди. Іноді, ці так звані “inliers” виявляються підозрілими, якщо розглядати їх по відношенню до інших змінних, використовуючи ділянки розсіювання, регресійний аналіз або перевірки послідовності. Деякі з них можна виявити, вивчаючи історію кожної точки даних або шляхом повторного вимірювання, але таке дослідження рідко можливо. Натомість, можна оцінити та/або змінити вибірку об’єктів-помилок для оцінки частоти помилок.

Очевидно, що висновки зроблені на основі результатів аналізу брудних даних не можуть вважатися достатньо надійними для прийняття рішень. Однак наявність таких даних не обов'язково означає необхідність їх очищення або ж запобігання появи. Завжди повинен бути розумний вибір між наявністю брудних даних і вартістю і/або часом, що буде витрачено на їх очищення.

### 2.1.4 Очищення даних

Очищення даних (data cleaning, data cleansing або scrubbing) займається виявленням і видаленням помилок і несумісностей в даних з метою поліпшення їх якості.

Проблеми з якістю зустрічаються в окремих наборах даних – таких як файли і бази даних. Коли інтеграції підлягає безліч джерел даних (наприклад в Сховищах, інтегрованих системах баз даних або глобальних інформаційних Інтернет-системах), необхідність в очищенні даних істотно зростає. Це відбувається тому, що джерела часто містять розрізнені дані в різноманітному представленні.

Спеціальні засоби очищення зазвичай мають справу з конкретними областями – в основному це імена і адреси – або ж з виключенням дублікатів. Перетворення забезпечуються або у формі бібліотеки правил, або користувачем в інтерактивному режимі. Перетворення даних можуть бути автоматично отримані за допомогою засобів узгодження схеми [23].

На сьогоднішній день інтерес до очищення даних зростає. Цілий ряд дослідницьких груп займається загальними проблемами, пов'язаними з очищенням даних, в тому числі, зі специфічними підходами до Data Mining і перетворенню даних на підставі зіставлення схеми.

### 2.1.5 Методи нормування даних

Кожна вибірка вихідних даних, що завантажується в аналітичний додаток, характеризується набором властивостей, які можуть вплинути на ефективність роботи моделі і знизити вірогідність результатів аналізу. Навіть якщо дані очищені від таких факторів, що погіршують їх якість, як дублікати, протиріччя, шуми, аномальні значення, пропуски та ін., вони все ще можуть не відповідати методиці і цілям аналізу.

Дані можуть бути не впорядковані, представлені в форматах, з якими не працює той чи інший алгоритм. Для вирішення цієї проблеми використовують трансформацію даних, тобто перетворення до певного формату/виду тощо. Під трансформацією розуміють набір підходів та алгоритмів, використання яких дозволить оптимізувати формати та представлення даних з точки зору поставленої задачі її цілей та використати дані з максимальною ефективністю.

До методів трансформації відносять:

- квантування або дискретизацію даних;
- сортування;
- злиття;
- налаштування даних;
- додавання нових атрибутів, значення яких обчислені на основі тих, що вже наявні в даних;
- нормалізація даних;

Нормалізація даних дозволяє привести всі дані до одного діапазону. Частіше за все використовують логарифмування або стиснення всіх значень до інтервалу [0; 1]. Окрім цього використовують десяткове масштабування, мінімаксну нормалізацію, використання стандартного відхилення і поелементних перетворень.

### 2.1.6 Методи заповнення пропусків

Проблема пропущених значень досить актуальна, наприклад для соціології. Часто при прийнятті важливих стратегічних рішень, що стосуються наприклад, ринку праці, використовуються результати соціологічних досліджень, заснованих на масових опитуваннях.

Причинами неповноти даних опитування можуть служити безліч факторів: неуважність респондента, помилки в анкеті, відмінність в даних анкет і т.д. В результаті на етапі аналізу даних ми маємо неповний масив. Саме для таких випадків і призначені методи відновлення даних з пропусками.

На даний момент існує безліч методик, які дозволяють ефективно відновлювати дані з пропусками, але у кожної з них є свої плюси і мінуси.

Існує безліч способів заповнення пропусків (ремонту вибірки) вже після етапу збору даних: заповнення середнім значенням, пропорційне розміщення спостережень з пропущеними, розрахунок можливого значення за допомогою регресійної моделі і так далі. Серед них:

- Виключення рядків пропусками. Даний метод легко реалізувати, але необхідною умовою його застосування є проходження даних вимогу MCAR (missing completely at random), тобто пропуски в даних по змінним повинні бути повністю випадковими. Крім того, він зазвичай застосовується лише при незначній кількості пропусків у таблиці, інакше отримана на виході таблиця даних стає непредставницьким. Головний недолік такого підходу зумовлений втратою інформації при виключенні неповних даних.
- Заповнення пропусків середніми по стовпчику значеннями. Даний метод також легко реалізувати, але його застосування має сенс тільки в разі задоволення даних умові MCAR (missing at random), тобто коли пропуски в даних є випадковими і сам механізм пропусків неістотний. До недоліків методу відносять спотворення розподілу даних, зменшення дисперсії.

Регресійне моделювання пропусків У більшості випадків, заповнення пропусків за допомогою регресійних моделей здійснюється в два етапи:

1. На першому етапі за сукупністю повних спостережень відбудовується регресійна модель, і оцінюються коефіцієнти в рівнянні, де в якості залежної змінної виступає цільова змінна – пропущені значення по якій необхідно відновити;
2. Потім за отриманим на попередньому етапі рівнянню, в яке підставляються відомі значення незалежних змінних предикторів, для кожного цільового об'єкта розраховується відсутнє значення по залежній цільовій змінній. У разі інтервальних і абсолютних змінних розраховується конкретне значення, а для порядкових і номінальних змінних з певною ймовірністю передбачається категорія, до якої повинен бути віднесений об'єкт.
3. Метод Бартлетта є ще одним підходом до вирішення проблеми наявності пропусків в вихідних даних. У разі активного експерименту значення факторів задаються дослідником, і пропуски в вихідних змінних спостерігаються частіше, ніж у вхідних факторах. У методі Бартлетта пропущеному значенню присвоюється якесь початкове (наприклад, нуль або середнє арифметичне стовпчика), потім будується багатомірне лінійне регресійне рівняння з використанням початкового значення пропуску. Кінцеве значення присвоюється, виходячи з результатів аналізу регресії.
4. Метод максимізації очікувань (EM – expectation maximization) дозволяє не тільки відновлювати пропущені значення з використанням двохетапного ітеративного алгоритму, а й оцінювати середні значення, коваріаційні і кореляційні матриці для кількісних змінних. EM – алгоритм, в загальному сенсі являє собою ітераційну процедуру, ціллю якої є вирішення задачі оптимізації деякого функціоналу, через пошук екстремуму цільової функції. Цей алгоритм реалізується в 2 етапи.

## 2.2 Вибір базових класифікаторів

### 2.2.1 Загальна постановка задачі класифікації

Загалом, розрізняють навчання з вчителем, навчання без вчителя, напівавтоматичне та навчання з підкріпленням (рис 2.2) .

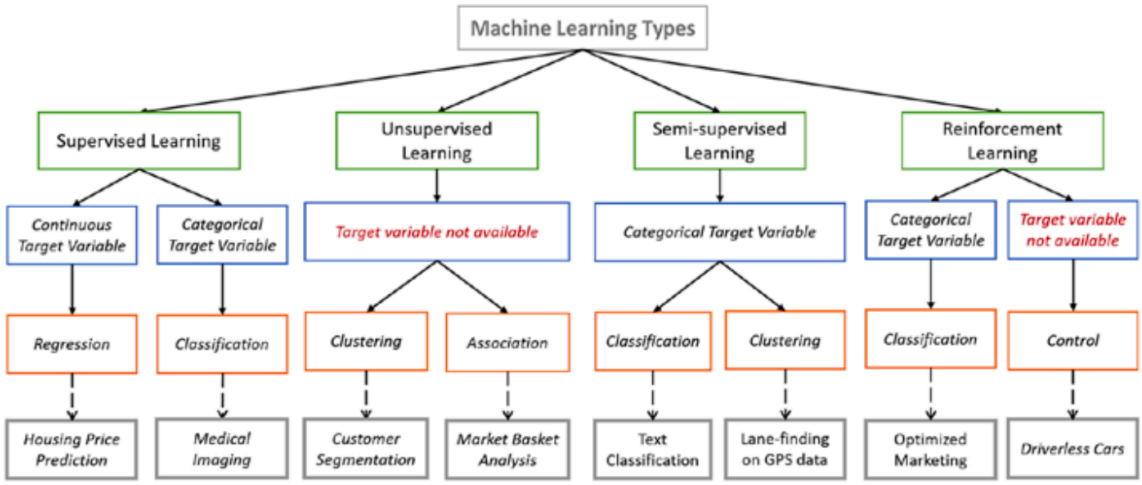



Рисунок 2.2 – Типи задач машинного навчання

Предмет вивчення в даній роботі охоплює тип машинного навчання з вчителем, а саме – класифікацію.

Згідно з матеріалом лекцій з машинного навчання К. В. Воронцова [24], задача навчання по прецедентам формально виглядає наступним чином:

Нехай X – множина об’єктів, Y – множина відповідей, $y: X \to Y$ – невідома залежність (target function).

Дано:

$\{x_1, \dots, x_l\} \subset X$ - навчальна вибірка (training sample);

$y_i = y(x_i), i = 1, \dots, l$ - відомі відповіді.

Знайти:

$a: X \to Y$ – алгоритм, функцію прийняття рішення (decision function), що приближує $y$ на всій множині $X$.

Розглянемо більш детально, що можуть собою представляти об’єкти та відповіді.

$f_j: X \to D_j, j = 1, \ldots, n$ – ознаки об'єктів (features).

Типи ознак:

– $D_j = \{0,1\}$ – бінарна ознака $f_j$;

– $|D_j| < \infty$ - номінальна ознака $f_j$;

– $|D_j| < \infty$, $D_j$ впорядкована – порядкова ознака $f_j$;

– $D_j = R$ – кількісна ознака $f_j$;

Вектор $(f_1(x), \ldots, f_n(x))$ називають вектором ознак об'єкта $x$.

Розглядають матрицю «об'єкти-ознаки» (feature data)

$$F = \begin{pmatrix} f_1(x_1) & \cdots & f_n(x_1) \\ \vdots & \ddots & \vdots \\ f_1(x_l) & \cdots & f_n(x_l) \end{pmatrix}$$

Відповіді можуть бути задані наступним чином.

Для задач класифікації (classification):

– $Y = \{-1; +1\}$ – класифікація на 2 класи.

– $Y = \{1, \ldots, M\}$ – на M класів, що не перетинаються.

– $Y = \{0,1\}^M$ – на M класів, що можуть перетинатися.

Для задач регресії (regression):

– $Y = R$ або $Y = R^m$.

Для задач ранжування (ranking):

– Y – скінчена впорядкована множина.

Метод навчання (learning algorithm) це відображення вигляду

$$\mu: (X \times Y)^l \to A,$$

що будь-якій виборці $X^l$ ставить у відповідність деякий алгоритм $a \in A$.

Задачі навчання по прецедентах завжди складаються з двох етапів: навчання та тестування.

В якості функціоналів якості використовують поняття функції втрат (loss function) $\mathcal{L}(a, x)$ – величина помилки алгоритму $a \in A$ на об'єкті $x \in X$.

Для задач класифікації, як правило, використовують $\mathcal{L}(a,x) = [a(x) \neq y(x)]$ – індикатор помилки. У випадку вирішення задач регресії користуються або абсолютним значенням помилки ($\mathcal{L}(a,x) = |a(x) - y(x)|$) або квадратичною похибкою ($\mathcal{L}(a,x) = (a(x) - y(x))^2$).

Задача навчання зводиться до задачі оптимізації шляхом введення поняття емпіричного ризику як функціоналу якості алгоритму *a* на $X^l$:

$$Q(a, X^l) = \frac{1}{l}\sum_{i=1}^{l} \mathcal{L}(a, x_i),$$

$$\mu(X^l) = \arg\min_{a \in A} Q(a, X^l)$$

Отже, класифікація – це один із розділів машинного навчання, що вирішує наступну задачу: нехай маємо множину об’єктів, що розділені на класи за деякою або рядом ознак. Окрім цього, задана скінчена множина об’єктів, щодо яких відомо, яким саме класам вони належать. Цю множину називають навчальною вибіркою. До якого класу відносяться решта об’єктів – невідомо.

Необхідно побудувати алгоритм, що буде здатний класифікувати будь-який з об’єктів вихідної множини. Під класифікацією об’єкта розуміють відповідь алгоритму, що вказує номер (або назву) класу, до якого він відноситься.

Задача класифікації в машинному навчанні відноситься до задач навчання з вчителем. Їх поділяють:

- За типом вхідних даних: описи ознак, матриця відстаней, часові ряди, зображення тощо
- За типами класів: двокласова класифікація, багатокласова, класи, що не перетинаються, класи, що перетинаються, та нечіткі класи.

Алгоритми класифікації широко застосовуються в усіх сферах життя людини. Найбільш розповсюдженими є застосування в скорингових моделях (оцінки кредитоспроможності), медицині, маркетингу, розпізнавання образів, мови та символів, пошук спаму та навіть в геології.

### 2.2.2 Лінійні класифікатори

Розглянемо задачу кредитного скорингу (рис 2.3). Нехай ми хочемо побудувати класифікатор, що розрізняє позичальника, що є потенціальним банкрутом. Характеристики прибутку та кредитний рейтинг визначають потенційних дефолтерів.

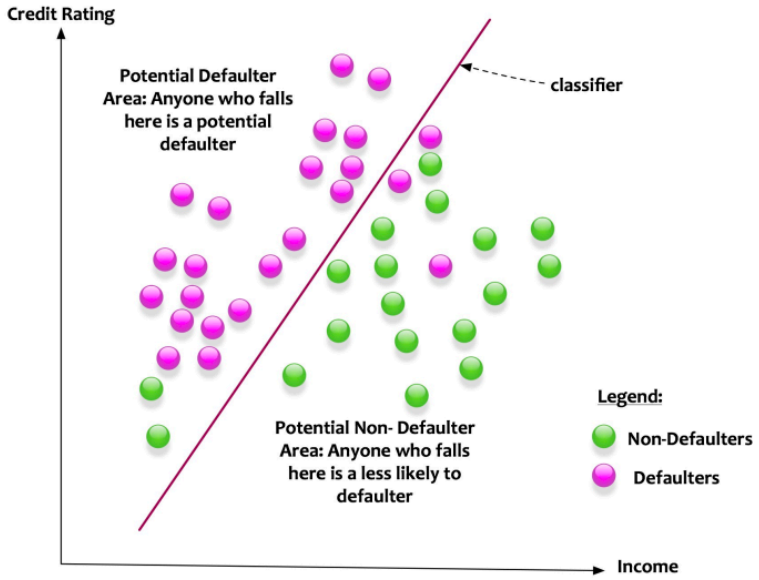



Рисунок 2.3 – Задача кредитного скорингу

Дана діаграма ілюструє сценарій. Для простоти вважаємо, що простір ознак це перетин доходу та кредитного рейтингу. Зелені точки це надійні позичальники, рожеві – банкрути. Класифікатор навчається на вхідних даних та створює лінію, що розділяє всі об’єкти на дві групи. Результатом роботи алгоритму є модель, що класифікує клієнтів за наступним принципом:

– Всі, хто потрапив наліво від лінії – потенційні банкрути.

– Всі, хто потрапив справа від лінії – потенційні небанкрути.

Подібні класифікатори, що розбивають простір ознак лінією (або площиною в просторі, розмірність якого більше двох), називаються лінійними.

Професор К. В. Воронцов формально описує задачу лінійної класифікації наступним чином. [24]

Нехай об’єкти описуються n числовими ознаками $f_j: X \to R$. Тоді простір ознакових описів об’єкту $X = R^n$. Нехай $Y$ – скінчена множина маркерів класів.

У випадку бінарної класифікації ($Y$ = {-1; +1}) лінійним класифікатором називають алгоритм класифікації $a: X \to Y$ вигляду:

$$a(x, w) = sign\left(\sum_{j=1}^{n} w_j f_j(x) - w_0\right) = sign\langle x, w\rangle,$$

де $w_j$ – вага j-тої ознаки, $w_0$ – поріг прийняття рішення, $\langle x, w\rangle$ – скалярний добуток.

Навчання лінійних класифікаторів зазвичай проводиться методом мінімізації емпіричного ризику.

Налаштування класифікатора даним методом полягає в тому, щоб на навчальній вибірці пар «об’єкт, відповідь» побудувати алгоритм, що мінімізуватиме функціонал емпіричного ризику, що має вигляд:

$$Q(w) = \sum_{i=1}^{m} [a(x_i, w) \neq y_i] \to \min_w$$

У випадку бінарної класифікації зручно визначити для довільного об’єкта з навчальної вибірки $x_i \in X^m$ величину відступу (margin):

$$M(x_i) = y_i \langle x_i, w\rangle$$

У випадку довільного числа класів, відступ визначається як

$$M(x_i) = \langle x_i, w_{y_i}\rangle - \max_{y \in Y, y \neq y_i} \langle x_i, w\rangle$$

Взагалі, відступ можна розуміти як «степінь занурення» об’єкта в свій клас. Чим менше значення відступу, тим ближче знаходиться об’єкт до розділяючої поверхні і відтак, ймовірність невірної його класифікації

підвищується. Відступ $M(x_i)$ набуває від'ємних значень тоді і тільки тоді, коли алгоритм $a(x)$ припускається помилки на об'єкті $x_i$ . Це спостереження дозволяє записати функціонал емпіричного ризику в наступному вигляді:

$$Q(w) = \sum_{i=1}^{m} [M(x_i) < 0]$$

Під знаком суми стоїть порогова функція втрат, а отже функціонал $Q(w)$ не є ні випуклим, ні навіть неперервним, що є причиною незручності мінімізації і потребує апроксимації функції втрат.

Лінійні методи класифікації різняться між собою, перш за все, вибором функції L(M), неперервною апроксимацією порогової функції втрат, способом регуляризації та вибором численного методу оптимізації функціоналу.

Регуляризація – це один із способів боротьби з феноменом перенавчання. Він полягає у додаванні штрафного доданку, який не дозволяє додавати занадто великі значення норми вектору вагових коефіцієнтів і тим самим підвищує його стійкість до незначних змін в навчальній виборці.

2.2.2.1 Лінійний дискримінант Фішера

Відповідає квадратичній апроксимації за умови, що від кожного вектору ознак попередньо віднімається вектор центрів класів:

$$[M < 0] \leq (1 - M)^2]$$

Оптимізація проводиться методом найменших квадратів.

2.2.2.2 Одношаровий персептрон

Використовується апроксимація сигмоїдною або логіт-функцією:

$$[M < 0] \leq \frac{2}{1 + e^{\alpha M}}$$

де параметр α - деякий параметр, що задається апріорі. В цьому випадку необхідне використання додаткових евристик для зменшення перенавчання таких, як регуляризація та стохастичні методи для вибивання із локальних мінімумів.

2.2.2.3 Логістична регресія

Логістична регресія отримала свою назву від функції, що лежить в її основі. Логістична функція, або сигмоїдна функція (рис 2.4) була розроблена статистами для опису росту популяції в екології. Вона має форму літери S та може приймати будь-яке значення між 0 та 1, але ніколи не дорівнює їм.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

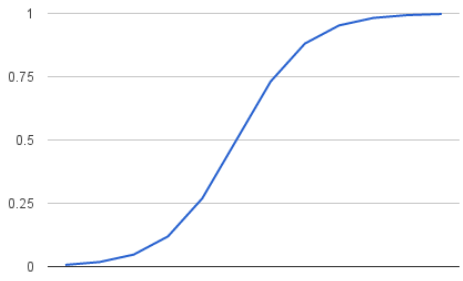



Рисунок 2.4 – Логістична або сігмоїдна функція

Якщо розглядати введене вище поняття відступу, то в цьому випадку порогову функцію втрат апроксимують неперервною оцінкою зверху наступним чином:

$$[M < 0] \leq \log_2(1 + e^{-M})$$

Навчання зазвичай проводять методом градієнтного спуску другого порядку.

2.2.2.4 Метод опорних векторів (Support Vector Machine or SVM)

Один з найпоширеніших методів навчання з учителем. Відповідає кусково-лінійній апроксимації:

$$[M < 0] \leq (1 - M)_+$$

В даному випадку використання регуляризації із квадратичною нормою є обов'язковим. Навчання виконують за допомогою спеціалізованих методів квадратичного програмування.

До переваг методу належить його швидкість, гарантована наявність єдиного розв'язку (наслідок зведення до задачі квадратичного програмування), а також здатність методу знаходити розділяючу полосу максимальної ширини, що дозволяє в подальшому більш впевнено проводити класифікацію.

До недоліків методи опорних векторів відносять його чутливість до шуму та відсутність єдиного підходу до вибору ядра та його параметрів.

Існують також нелінійні модифікації вектору опорних векторів. Лінійність або нелінійність алгоритму визначається типом ядра.

Наприклад, метод, запропонований Майкрософт - Two-Class Locally Deep Support Vector Machine, програмна реалізація якого є частиною Azure Machine Learning Studio. Особливістю цього підходу є особлива функція ядра, що була

розроблена спеціально для зменшення часу навчання при відсутності погіршення точності алгоритму класифікації.

### 2.2.3 Метод k найближчих сусідів

*Узагальнений метричний класифікатор*

Для довільного $x \in X$ відранжуємо об'єкти $x_1, \ldots x_l$:

$\rho(x, x^{(1)}) \leq \rho(x, x^{(2)}) \leq \cdots \leq \rho(x, x^{(l)})$,

$x^{(i)}$ – $i$-тий сусід об'єкта $x$ серед $x_1, \ldots x_l$;

$y^{(i)}$ – відповідь на $i$-тому сусіді об'єкта $x$.

Тоді метричний алгоритм класифікації шукаємо в наступному вигляді:

$$a(x; X^l) = \arg\max_{y \in Y} \sum_{i=1}^{l} [y^{(i)} = y]\, w(i, x)$$

$w(i, x)$ – ваговий коефіцієнт, що вказує на степінь важливості $i$-того сусіда об'єкта $x$, невід'ємний та не зростає по і.

$\sum_{i=1}^{l} [y^{(i)} = y]\, w(i, x)$ – оцінка близькості об'єкта $x$ до класу $y$.

У випадку, коли $w(i, x) = [i \leq 1]$, маємо метод найближчого сусіда.

Якщо ж $w(i, x) = [i \leq k]$ – метод k найближчих сусідів.

Іншими словами, об'єкт відноситься до того чи іншого класу в залежності від того, якому класу належить більшість його найближчих сусідів.

До переваг методу відносяться:

– простота реалізації;

– можна адаптувати під конкретну задачу, обравши метрику чи ядро;

– як правило, чудово працює як першочергове вирішення задачі; при чому, не тільки регресії або класифікації, але, наприклад, рекомендації.

Серед недоліків алгоритму зазначають:

- метод вважається швидким, але в реальних задачах, як правило, число сусідів, що використовуються для класифікації, буде доволі великим (100-150), а в такому випадку алгоритм працюватиме повільніше за, скажімо, дерева рішень;
- якщо простір ознак об’єкта великий, то важко підібрати відповідні вагові коефіцієнти та визначити, які з ознак є неважливими для вирішення даної задачі;
- залежність від обраної метрики виміру відстані між об’єктами (як правило, вибір за замовчуванням евклідової відстані є необґрунтованим). Можна відшукати найкраще рішення шляхом перебору параметрів, але для великого об’єму даних така процедура забирає багато часу;
- немає теоретичного обґрунтування обраній кількості сусідів, тільки шляхом перебору. У випадку недостатньої кількості сусідів метод схильних до перенавчання, тобто стає чутливим до викидів;

### 2.2.4 Наївний байєсовський класифікатор

На відміну від методу k найближчих сусідів, що для класифікації нових об’єктів використовує поняття близькості (як правило евклідова відстань або косинусну), наївний класифікатор Байєса використовує концепцію «ймовірності».

Іншими словами, це класифікатор, що використовує Теорему Байєса та визначає можливість належності об’єкта тому чи іншому класу як ймовірність належності, розрахована для кожного класу. В результаті об’єкт відносять до класу, показний ймовірності для якого вищий.

Класифікація за даним методом проводиться в припущенні, що кожен із об’єктів $x \in X$ описується $n$ незалежними ознаками. Це означає, що функція правдоподібності для класів розпадається на добуток.

Проф Воронцов стверджує [24] , що припущення про незалежність ознак значно спрощує задачу, адже оцінити *n* одномірних щільностей значно простіше, ніж одну *n*-мірну щільність. На жаль, воно дуже зрідка справджується на практиці, звідси й назва – «наївний».

Класифікатор в даному випадку може бути як параметричним, так і непараметричним, в залежності від того, яким методом відновлюються одновимірні щільності.

До переваг методу відносять просту реалізацію та невисокі обчислювальні витрати під час навчання та класифікації. У тих надзвичайно рідких випадках, коли ознаки дійсно взаємо незалежні, наївний класифікатор байєса – (майже) оптимальний.

Основний його недолік – відносно низька точність класифікації в більшості реальних задач.

Найчастіше ним користуються для порівняння різних моделей алгоритмів або як структурну одиницю в більш складних ансамблях алгоритмів.

### 2.2.5 Дерева рішень

Decision Trees або дерева рішень - один з логічних методів класифікації. Як зазначає в своїх лекціях проф Воронцов [25], деревом називають скінченний зв’язний граф з множиною вершин V, що не містить циклів і має виділену вершину $v0 \in V$, в яку не входить не одне ребро. Ця вершина називається коренем дерева. Вершина, що не має ребер, які б з неї виходили, називається термінальною або «листом». Інші вершини називаються внутрішніми. Дерево називають «бінарним», якщо з будь-якої його внутрішньої вершини виходить рівно 2 ребра.

Бінарним деревом рішень називають класифікатор, що задається бінарним деревом, в кожній вершині якого прописаний предикат $\beta v : X \rightarrow \{0, 1\}$, а кожному «листку» відповідає мітка класу.

Використовується поняття ентропії або кількості інформації за Шенноном, визначається вона наступним чином:

$$S = -\sum_{i=1}^{N} p_i \log_2 p_i,$$

де $p_i$ – ймовірність знаходження системи в стані $i$.

В основі поширених алгоритмів побудови дерев рішень, таких як ID3 та C4.5, лежить принцип жадібної максимізації приросту інформації. Іншими словами, на кожному кроці обирається та ознака, при розбитті якої приріст ентропії виявляється найбільшим.

Після цього процедуру повторюють до тих пір, поки кількість інформації не дорівнюватиме нулю або не стане менше заданого числа. Різні алгоритми використовують різноманітні евристики для усічення, щоб не трапилось перенавчання.

Серед інших критеріїв якості розбиття в задачах класифікації виділяють:

- Невизначеність Джині (Gini impurity): $G = 1 - \sum_k (p_k)^2$. Під максимізацією цього критерію розуміють максимізацію числа пар об’єктів одного класу, що опиняються в одному під-дереві. [26]
- Помилку класифікації (misclassification error):

$$E = \max_k p_k$$

На практиці помилку класифікації майже не використовують, а ентропія та невизначеність Джині дають приблизно однакові результати роботи.

Переваги використання дерев рішень:

- інтуїтивно зрозумілий алгоритм, який легко представити в графічному вигляді та інтерпретувати;
- можуть працювати як з числовими, так і з категоріальними ознаками;

- вимагають відносно невеликої попередньої обробки даних з боку користувача;
- нелінійні залежності між параметрами не впливають на якість роботи алгоритму.

Недоліки:

- нестійкі до перенавчання;
- нестійкі навіть до невеликих відхилень в даних (для усунення цього недоліку рекомендується використання дерев як частини ансамблю алгоритмів таких як беггінг та бустинг);

## 2.3 Використання ансамблів моделей класифікації як більш ефективного алгоритму

### 2.3.1 Беггінг

Розглянемо спочатку поняття бутстрепінгу. У машинному навчанні метод бутстреп (bootstrap) відноситься до випадкового вибору піднаборів даних з заміною. Цей зразок називається ресамплером. Це дозволяє моделі або алгоритму краще зрозуміти різноманітні упередження, відхилення та особливості, що існують у ресамплері. Приймаючи зразок даних, resample може містити різні характеристики, такі, які могли міститися в цілому датасеті. Продемонстровано на рисунку 2.5 , де кожна вибірка населення має різні частини, і жоден з них не ідентичний. Це вплине на загальне середнє значення, стандартне відхилення та інші описові показники набору даних. У свою чергу, такий підхід може допомогти розвинути більш надійні моделі.

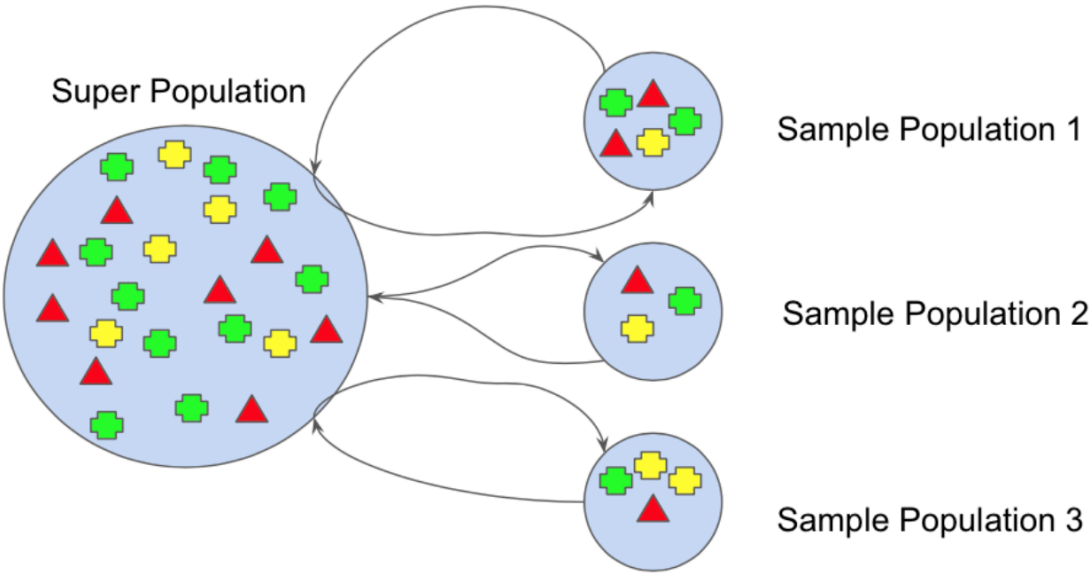



Рисунок 2.5 – Іллюстрація бутстрепу

Бутстрепінг також чудово підходить для наборів даних невеликого об’єму, які можуть мати тенденцію до перенавчання. Причиною використання методу бутстрепу є те, що він може перевірити стабільність рішення. Використання кількох вибіркових наборів даних, а потім тестування декількох моделей може підвищити надійність. Можливо, один вибірковий набір даних має більше середнє значення, ніж інший, або інше стандартне відхилення. Це може зламати модель, яка була перенавчена, а не тестувалась з використанням наборів даних з різними варіантами.

Однією з багатьох причин такої популярності даного підходу стало збільшення обчислювальної потужності. Це дозволяє робити в багато разів більше перестановок з різними піднаборами, ніж раніше. Бутстрепінг використовується як в Беггінгу, так і в Бустингу.

Беггінг (Bagging or Bootstrap Aggregating) – один з основних методів агрегації в машинному навчанні, який використовує паралельне навчання класифікаторів. Загалом, процес беггінгу можна описати наступними кроками:

1. Перш за все, із множини вихідних даних випадковим чином відбирається декілька підмножин, що містять в собі кількість об’єктів, що відповідає кількості об’єктів початкового набору.

2. Варто тримати в голові, що, оскільки відбір реалізується випадковим чином, то набір об’єктів завжди буде різним: деякі приклади потраплять в декілька підмножин, а деякі – в жодну.

3. На основі кожної вибірки будується класифікатор.

4. Результати роботи класифікаторів агрегуються.

Агрегація результатів, зазвичай, відбувається шляхом усереднення або голосування. При чому, перший варіант використовується в задачі регресії, а другий – класифікації.

Використання беггінгу дозволяє зменшити відхилення.

Один з класичних алгоритмів класифікації, в основу якого покладено бегінг – це Випадковий Ліс або Random Forest.

Випадковий ліс - це алгоритм, що базується на деревах рішень, який включає створення декількох дерев, а потім поєднання їх результатів для покращення можливостей узагальнення моделі. Метод об'єднання дерев відомий як метод ансамблю. Ансамбль - це не що інше, як поєднання слабких учнів (окремих дерев) для створення сильного учня.

Випадкові ліси стійкі до перенавчання. Кількість дерев можна збільшувати як завгодно, швидкість роботи все ще буде високою.

Як працює випадковий ліс: навчальна вибірка для поточного дерева складається шляхом відбору зразків (samples) із заміною, близько однієї третини випадків залишаються поза зразком. Ці дані, що мають назву “out-of-bag”, використовуються для отримання неупередженої оцінки помилок класифікації по мірі того, як нові дерева додаються до лісу. Цей підхід також використовується для отримання оцінок важливості змінних.

Після побудови кожного дерева всі дані пропускаються вниз по дереву, а близькість обчислюється для кожної пари випадків. Якщо два випадки потрапили в одну й ту ж кінцеву вершину (лист), значення їх близькості збільшується на один. В кінці пробігу, показники близькості нормалізуються шляхом ділення на кількість дерев.

У випадкових лісах відсутня потреба в крос-валідації або окрема тестова вибірка для отримання об'єктивної оцінки помилки тестового набору.

Внутрішньо, під час запуску, вона оцінюється наступним чином: кожне дерево конструюється за допомогою іншого зразка бутстреп з вихідних даних. Близько однієї третини випадків залишаються поза піднабором та не використовуються при побудові k-го дерева. Кожен об’єкт, що не був використаний при побудові дерева k-го дерева, пропускається через алгоритм, щоб отримати класифікацію. Таким чином, класифікація об’єктів тестової вибірки проводиться в середньому третиною дерев. В кінці пробігу для кожного об’єкта обирають j -тий клас, який отримав більшу кількість голосів кожного разу, коли цього об’єкта не було в навчальній вибірці. Частка випадків, коли j не дорівнює дійсному класу n, усереднена по всіх випадках, дає нам оцінку помилки.

У кожному дереві, вирощеному в лісі, відкладемо випадки out-of-bag і підрахуємо кількість голосів, «відданих» за правильний клас. Тепер випадково перемішаємо значення змінної m і пропустимо через у дерево. Віднімемо кількість голосів за правильний клас у змінних даних out-of-bag від кількості голосів за правильний клас на недоторканих даних. Середнє значення цього числа у всіх деревах лісу є значенням важливості для змінної m.

Якщо значення цього показника від дерева до дерева незалежні, то стандартна помилка може бути обчислена за допомогою стандартного обчислення. Кореляції цих оцінок між деревами були обчислені для ряду наборів даних і виявилися досить низькими, тому ми класично вираховуємо стандартні похибки, ділимо оцінку рядку за стандартною помилкою, щоб отримати z-бал, і призначати рівень значущості до z-оцінки припускаючи нормальність.

Якщо кількість змінних є дуже великою, ліси можуть бути запущені один раз з усіма змінами, а потім запустити знову, використовуючи лише найважливіші змінні з першого запуску.

У деяких наборах даних помилка прогнозування між класами вельми незбалансована. Деякі класи мають низьку помилку прогнозування, інші - високу. Це відбувається зазвичай, коли один клас набагато більшим, ніж інший. Тоді випадкові ліси, намагаючись мінімізувати загальну частоту помилок, будуть підтримувати низький рівень помилок у великому класі, дозволяючи меншим класам мати більший рівень помилок. Наприклад, при відкритті ліків,

коли дана молекула класифікується як активна чи ні, загальним є те, що кількість активів перевищує 10 на 1, до 100 до 1. У таких випадках частота помилок на цікавому класі (активах) буде дуже високим.

Зазвичай, користувач може виявляти дисбаланс за виводами коефіцієнтів помилок для окремих класів.

Щоб штучно збалансувати помилку в таких випадках, окремим класам назначають вагові коефіцієнти. Чим більша вага класу, тим менша помилка на ньому.

Окрім цього, використовується метод усічення (pruning), коли видаляються вузли нижчих рівнів.

Окремим розширенням ідеї методу випадкових лісів є ротаційний ліс або Rotation Forest, що був вперше запропонований в роботі [27], основна особливість якого полягає в підготовці даних для навчання простих класифікаторів. До піднаборів, отриманих випадковим розбиттям початкової вибірки, застосовується метод головних компонент. Основною метою методу, за словами розробників, є заохочення одночасно індивідуальної точності та різноманітності в ансамблі.

### 2.3.2 Бустинг

Як і беггінг, основним завданням цього методу агрегації є перетворення набору слабких класифікаторів ( тобто таких, що припускаються багатьох помилок на тестовій виборці) в один більш сильний. Але, на відміну від попереднього, тут навчання відбувається послідовно і кожний наступний класифікатор має на меті компенсацію недоліків попереднього.

Фінальний результат це, як правило, зважена лінійна комбінація відповідей всіх використаних алгоритмів.

Перелік алгоритмів, в основі яких знаходиться принцип бустингу, дуже великий:

- Stochastic Gradient Boosting – в основі лежить ідея використання методу градієнтного спуску, але кожний новий алгоритм налаштовувати на випадковому піднаборі даних. Детальний опис алгоритму наведений в роботі [28]
- LPBoost: Linear Programming Boosting – бустинговий алгоритм класифікації, що максимізує відступ;
- BrownBoost: дозволяє підвищити надійність алгоритму на наборах даних, що містять багато шуму, відкидаючи об’єкти, що постійно класифікуються помилково;
- LogitBoost: використовується логарифмічна функція втрат;
- AdaBoost – частковий випадок градієнтного бустинга, в якому в якості функції втрат використовується експоненційна. AdaBoost є першим теоретично дослідженим варіантом бустинга, для якого вперше була доведена основна теорема з аналітично виведеними оптимальними $b_m$ на кожному кроці. Початкова версія алгоритму повертала тільки номер класу в бінарній класифікації, а пізніше він був узагальнений і тепер є можливість отримати ймовірність належності тому чи іншому класу;
- CatBoost – технологія, що була запропонована Яндексом минулого року. В основі алгоритму лежить, знову ж таки, градієнтний бустинг. Але на відміну від більшості існуючих алгоритмів, цей здатний ефективно працювати з категоріальними даними. Яндекс використовує його для вирішення задач ранжування, прогнозування та побудови рекомендацій.
- XGBoost: скорочено від “Extreme Gradient Boosting”, бібліотека, що реалізує градієнтний бустинг над деревами [29]
- LightGBM: в основі також лежить градієнтний бустинг над деревами, але на відміну, від XGBoost, побудова дерева відбувається горизонтальним шляхом.

Різницю між двома останніми алгоритмами наглядно демонструє наступна схема (рис 2.6 – 2.7).

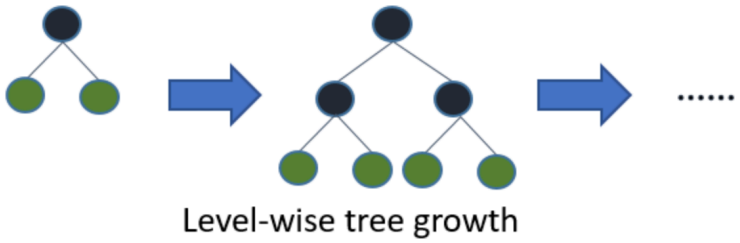



Рисунок 2.6 - Нарощування листків з огляду на рівень в (XGBoost)

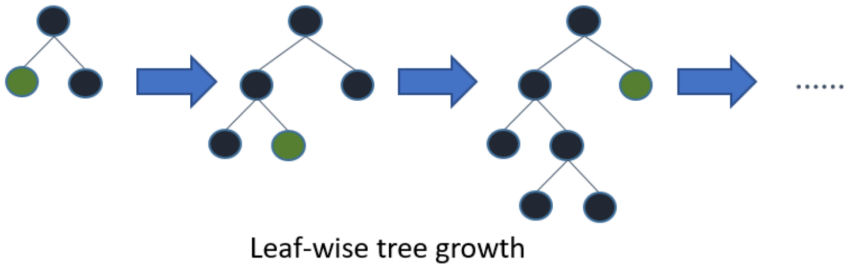



Рисунок 2.7 - Горизонтальне нарощування (LightGBM)

На цьому перелік моделей, що використовують бустинг, не закінчується.

У роботі [30] досліджують проблему кореляції між простими класифікаторами та пропонують 4 шляхи її вирішення, серед яких:

- Навчати кожний класифікатор на окремих піднаборах з даних, що не перетинаються;
- Генерація нової навчальної вибірки шляхом «усічення»: відкидання надлишкової інформації з вхідних даних;
- Використання ресемплінгу
- Зважене комбінування та розбиття на піднабори шляхом використання певного принципу, а не випадковим чином, як в попередніх методах.

## 2.4 Метрики оцінки якості роботи класифікаторів

Після етапу проектування та відбору ознак і, звичайно ж, побудови моделі і отримання певних результатів у формі класів або ймовірностей, наступним

кроком має бути перевірка ефективності моделі за допомогою спеціалізованих метрик та тестового набору даних. В залежності від типу задачі машинного навчання, аналітик даних обирає, якими метриками користуватися в тому чи іншому випадку.

Метрики якості, які ви обираєте для оцінки моделі машинного навчання, - дуже важливі. Вибір метрик має безпосередній вплив на те, як алгоритми оцінюються та порівнюються між собою. Для цього необхідно чітко розуміти природу досліджуваної проблеми.

Оцінка якості роботи класифікаторів базується на понятті Матриці Помилок (Confusion Matrix), що для задачі бінарної класифікації має наступний вигляд (табл. 2.1):

Таблиця 2.1 – Матриця помилок

| Predicted values | True Values | y = 1 | y=0 |
|---|---|---|---|
| | a(x) = 1 | True Positive (TP) | False Positive (FP) |
| | a(x) = 0 | False Negative (FN) | True Negative (TN) |

Нехай розглядається задача медичної діагностики, де $y = 1$ означає наявність хвороби, а $y=0$, відповідно, її відсутність.

*True Positive* – випадок, коли реальне значення дорівнювало 1 (пацієнт хворий) і відповідь алгоритму – також 1.

*True Negative* – у здорового пацієнта алгоритм діагностував відсутність захворювання.

*False Positive* – випадок, коли реальне значення дорівнювало нулю, а прогнозоване – одиниці. «Невірна» відповідь тому, що алгоритм видав помилковий результат, але помилка вважається «позитивною», адже насправді людина здорова. Цей клас помилок несе в собі менше загрози ніж *False Negative,* коли алгоритм не розпізнав хворобу і визнав хвору людину здоровою. Тут йде річ про ціну помилки, що є дуже важливим аспектом в сфері медицини.

Очевидно, що в ідеальному випадку ми прагнемо до того, щоб алгоритм класифікації давав нуль помилок класів *FP* та *FN,* але в реальному житті так майже ніколи не буває, але кожна модель має мінімізувати кількість помилок.

### 2.4.1 Правильність (Accuracy)

Правильність класифікації визначається як співвідношення між вірними відповідями та загальною кількістю відповідей.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy – ефективна метрика якості, коли маємо справу із збалансованими класами. Однак, якщо кількість об’єктів одного класу значно перевищує інший, то точність покаже хороші результати, що насправді будуть неправдивими.

Наприклад, повернемося до задачі діагностування. Нехай у вибірці зі 100 людей всього 5 хворих. Нехай наша модель працює погано і діагностує всіх як «здорових». Отже, 95 здорових людей діагностовано вірно, а 5 насправді хворих – помилково. Не дивлячись на жахливу якість моделі, значення Accuracy буде становити 95%.

### 2.4.2 Точність (Precision)

$$Precision = \frac{TP}{TP + FP}$$

Точність це міра, що вказує на те, яка частка пацієнтів, в яких діагностували хворобу, насправді були хворими.

Повернемося до прикладу зі 100 об’єктами, 5 з яких мають захворювання. Скажімо, наша модель неякісна і в кожному випадку дає відповідь «хворий». У цьому випадку, матимемо співвідношення 5 до 100, а отже значення Precision становить 5%.

### 2.4.3 Повнота (Recall) або Чутливість (Sensitivity)

Повнота це міра, що надає нам інформацію про те, який відсоток хворих пацієнтів був діагностований як «хворий».

$$Recall = \frac{TP}{TP + FN}$$

Наприклад, 100 пацієнтів, 5 дійсно хворих. Нехай модель діагностує кожний випадок як наявність захворювання. В цьому випадку і чисельник, і знаменник дорівнює 5, а отже Повнота моделі становить 100%. В той час як Точність, як ми спостерігали вище, - 5%.

Даний приклад вказує на те, що треба бути надзвичайно уважним при виборі та використанні метрик якості, адже не в усіх випадках, отримане значення буде точно оцінювати ефективність моделі.

В яких випадках краще використовувати Точність, а в яких Повноту? Очевидно, що повнота дає нам інформацію про роботу класифікатора з огляду на *FN* (тих значень, що алгоритм «не вгадав»), в той час як Точність бере до уваги *FP*.

Отже загалом, все залежить від природи задачі та ціни помилки. Якщо ми маємо за мету мінімізувати кількість «невгаданих» значень, то намагатимемось наблизити Точність якомога ближче до 100 відсотків, намагаючись при цьому мати не найгірше значення Повноти. І навпаки, якщо важливіше мінімізувати

кількість *False positives* (здорові люди, діагностовані алгоритмом як «хворі»), то основний фокус має бути на підвищенні значення Точності.

### 2.4.4 Специфічність (Specificity)

$$Specificity = \frac{TN}{TN + FP}$$

Вказує на частку «негативних» прикладів, що були класифіковані вірно.

### 2.4.5 F-міра

Не дуже зручно весь час звертатися до Точності та Повноти тому кращим варіантом є використання міри, що б одночасно враховувала б обидві метрики.

Першим, що спадає на думку, є використання простого арифметичного середнього, але, як показує практика, це зовсім неефективно. Тому використовують гармонічне середнє або F-міру. Отож маємо:

$$F = \frac{2 * precision * recall}{precision + recall}$$

### 2.4.6 Log-loss (logarithmic loss)

Використовується у випадку, коли на виході алгоритму маємо не клас, до якого належить той чи інший об’єкт, а ймовірність, з якою об’єкт можна віднести

до певного класу. Log-loss - це "м'яке" вимірювання точності, що інкорпорує ідею довірчого інтервалу.

$$log - loss = -\frac{1}{N}\sum_{i=1}^{N} y_i * \log(p_i) + (1 - y_i)\log(1 - p_i),$$

де N – об’єм вибірки, $y_i$ – реальна мітка класу, $p_i$ – відповідь алгоритму на $i$-тому об’єкті.

### 2.4.6 ROC крива (Receiver Operating Characteristics Curve)

У статистиці ROC крива це графік, що ілюструє ефективність бінарної системи класифікації в залежності від заданого порогу для ймовірності належності класу. ROC крива надає детальну інформацію про поведінку класифікатора. Крива є результатом зображення на графіку взаємозв’язку *True Positive rate (TPR)* та False Positive rate (FPR) в залежності від порогу.

Таким чином, ROC крива ілюструє чутливість класифікатору, показуючи скільки правильно класифікованих об’єктів можна отримати, дозволяючи більше й більше випадків FP.

ROC крива (рис 2.8) для якісної моделі має велику площу під собою (адже частка правильно класифікованих об’єктів *(TPR)* зростає до 100% дуже швидко. Відповідно, графік кривої для поганого класифікатора лишатиме під собою дуже маленьку площу. Ідеальна модель, яка ніколи не помиляється, досягатиме стовідсоткової точності моментально, але такого майже ніколи не трапляється на практиці.

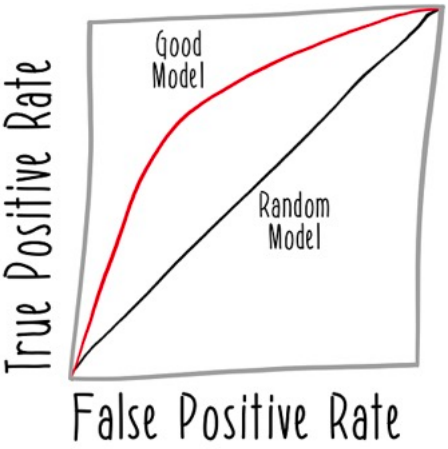



Рисунок 2.8 – ROC-крива

У роботі [32] зазначається, що площа під ROC кривою (AUC) – дуже розповсюджена міра якості роботи алгоритмів класифікації, адже має привабливу властивість об’єктивності, не вимагаючи особливого вкладу від користувача.

З іншого боку, AUC має й недоліки, деякі з яких дуже добре відомі дослідникам. Наприклад, AUC може видати потенційно оманливі результати, якщо криві перетинаються. Однак, AUC має ще один більш серйозний недолік, який, виявляється, не помічали. Він полягає у тому, що ця метрика фундаментально некогерентна в відношенні ціни невірної класифікації: AUC використовую різні розподіли ціни помилки для різних класифікаторів. Це означає, що використання AUC еквівалентне використанню різних метрик для оцінки різних правил класифікації або те ж саме, що сказати, що помилкове віднесення об’єкта до класу 1 в p разів гірше, ніж помилкове віднесення до класу 0, але, використовуючи інший класифікатор, помилкова класифікація до 1 в P разів гірша, де $p \neq P$. Це безглуздо тому, що відносна серйозність помилкової класифікації окремих об’єктів – це характеристика проблеми, що вивчається, а не класифікаторів, що були обрані для її вирішення. Це теж саме, що й порівняння росту двох людей, виміряного за допомогою лінійок, базові одиниці виміру яких самі залежать від зросту.

У роботі [32] математично обґрунтовано недоліки використання AUC та запропоновано альтернативу – H-міру. Автори стверджують, що, якщо

співвідношення вартості не визначено дослідником, то неминуче виникає неконтрольований аспект, пов'язаний з будь-якою метрикою, що інтегрується над таким розподілом. Для вимірювання H це полягає у виборі бета-розподілу та рівних значень параметрів. Проте ця незручність набагато менш тривожна, ніж внутрішня неоднорідність, спричинена мірою AUC, яка використовує різні розподіли для оцінки різних класифікаторів. Це означає, принаймні, що міра H робить справедливі порівняння.

У праці [33] запропоновано використання так званої Кривої Брієра – однієї з «кривих ціни», що зображує на графіку взаємозалежність втрат та умов роботи класифікатора. Тобто, площа під кривою характеризує очікуваний рівень втрат протягом всього робочого діапазону.

Окрім цього також використовують криву Presicion-Recall, на якій зображують точність та повноту при різних порогових значеннях. Крива presicion-recall ілюструє компроміс точності та моделі за різними порогами, що дозволяє легко порівнювати між собою алгоритми. Крива, яка проходить ближче до правої верхньої частини графіку, відповідає кращій моделі.

У роботі "Beyond Accuracy, F-score and ROC" [34] демонструють, що для задач медичної діагностики та біоінформатики використання класичних метрик може бути не завжди доречним та виконують пошук додаткових метрик. Основною вимогою до методів оцінювання якості є внесення нових характеристик в роботу алгоритму. А також, щоб моделі, оцінені вказаними метриками, було легко порівнювати між собою. Автори зацікавлені в двох характеристиках алгоритму:

- можливість підтвердження щодо класів, тобто оцінка імовірності правильних прогнозів маркерів обох класів;
- здатність уникнути невдач, а саме оцінка оцінки доповнення ймовірності невдачі.

Після детального вивчення існуючих підходів, були обрані 3 метрики, що використовуються в медичній діагностиці: індекс [35], ймовірність [36] та Дискримінантна степінь [37].

### 2.4.7 Перехресна перевірка

Важливим поняттям процесу оцінки роботи класифікатора є перехресна перевірка або крос-валідація. Крос-валідація це клас методів, що полягає у виключенні певної частини даних перед початком навчання класифікатора. Потім роботу моделі перевіряють на вилучених раніше, «нових» для моделі даних.

Метод Holdout - найпростіший вид перехресної перевірки. Набір даних складається з двох наборів, навчального та тестового. Апроксиматор функції налаштовує модель, використовуючи тільки тренувальний набір. Тоді аппроксиматору функції пропонується передбачити вихідні значення для даних в тестовій вибірці (ці значення алгоритмові зовсім незнайомі). Помилки, які він робить, накопичуються як і раніше, щоб дати середню помилку абсолютного тестової вибірки, яка використовується для оцінки моделі. Однак оцінка, отримана цим шляхом, може мати високу дисперсію. Оцінка може значною мірою залежить від того, які об’єкти опиняються в навчальній вибірці і які потрапляють до тестового набору, і, отже, оцінка може суттєво відрізнятися в залежності від того, як здійснюється поділ.

Перехресна перевірка K-fold є одним із способів поліпшення методу Holdout. Набір даних ділиться на k підмножин, а метод holdout повторюється k разів. Кожного разу, як один з підмножин k використовується як тестовий набір, а інші підмножини k-1 складаються разом для формування навчального набору. Тоді обчислюється середня помилка у всіх k випробуваннях. Перевага цього методу полягає в тому, що менше значення надається тому, як дані розділяються. Кожен об’єкт має бути рівно один раз в тестовій вибірці, а в навчальній - k-1 разів. Відхилення отриманої оцінки зменшується, коли k збільшується. Недоліком даного методу є те, що алгоритм тренування повинен бути перезаписаний з нуля k разів, що означає, що для здійснення оцінки потрібно витратити в кілька разів більше обчислень. Варіація цього методу полягає у випадковому розподілі даних на тестовий та тренувальний набір $k$ разів.

Перевага полягає в тому, що ви можете самостійно вибрати, наскільки великим є кожен тестовий набір, і на якій кількості випробовувань усереднювати отримане значення.

Leave-one-out крос-валідація - це перехресна перевірка K-fold, доведена до її логічної крайності, коли K дорівнює N, кількості об’єктів у наборі. Це означає, що N разів, апроксиматор функції тренується на всіх даних, за винятком однієї точки, і будує передбачення для цієї точки. Як і раніше, середня помилка обчислюється і використовується для оцінки моделі. Оцінка, надана помилкою Leave-one-out (LOO-XVE), хороша, але спочатку здається дуже громіздкою для обчислень. На щастя, класифікатори з локальними вагами можуть спрогнозувати LOO так само легко, як вони виконують звичайні прогнози. Це означає, що обчислення LOO-XVE займає не більше часу, ніж обчислення залишкової помилки, і це набагато кращий спосіб оцінити моделі. [34]

## Висновки до розділу

У даному розділі були розглянута постановка задачі машинного навчання та, зокрема, бінарної класифікації. Були вивчені поширені моделі класифікаціїї такі, як логістична регресія, байєсівський класифікатор, одношаровий персептрон, метод опорних векторів, метод *k* найближчих сусідів, дерева рішень та інші, детально були розглянуті переваги та недоліки кожного. Були розглянуті сучасні методи створення ансамблів класифікаторів, описана ідея кожного та виконаний огляд найновітніших розробок в цій сфері.

Було розглянуто методи оцінки якості роботи моделей, такі як точність, повнота, крива roc-auc та інші. Була пояснена концепція кросс-валідації, як підходу, що знижує варіативність результатів.

# РОЗДІЛ 3. ОЦІНЮВАННЯ ЯКОСТІ РОБОТИ БАЗОВИХ КЛАСИФІКАТОРІВ, ЇХ АНСАМБЛІВ ТА ЗАПРОПОНОВАНОЇ МОДЕЛІ ДЛЯ ДІАГНОСТУВАННЯ ПАЦІЄНТА

## 3.1 Попередня обробка даних тестової та навчальної вибірки

### 3.1.1 Опис набору даних

Розширений датасет Z-Alizadeh Sani [38] містить дані о 303 пацієнтах (об’єкти), кожен з яких описується 56 атрибутами. Атрибути розділені на 4 категорії: демографічні, симптоми та результати огляду, ЕКГ, а також атрибути лабораторних досліджень і знімків.

Маємо наступні атрибути для кожного об’єкта:

- 'Age' – вік;
- 'Weight' – вага;
- 'Length' – зріст;
- 'Sex' – стать;
- 'BMI' – індекс маси тіла;
- 'DM' – хворий на сахарний діабет (так/ні);
- 'HTN' - має гіпертонію (так/ні);
- 'Current Smoker' – палить (так/ні);
- 'EX-Smoker' – чи палив в минулому (так/ні);
- 'FH' - сімейна гіперхолестеринемія (так/ні);
- 'Obesity' – страждає від ожиріння (так/ні);
- 'CRF' – хронічна ниркова недостатність (так/ні);
- 'CVA' – цереброваскулярна аварія (так/ні)
- 'Airway disease' – хвороба дихальних шляхів (так/ні);
- 'Thyroid Disease' – захворювання щитовидної залози (так/ні);
- 'CHF' – застійна серцева недостатність (так/ні);
- 'DLP' – дисліпопротеїнемія (так/ні);
- 'BP' – кров’яний тиск;

- 'PR' – протромбіновий коефіцієнт;
- 'Edema' – наявність набряків (так/ні);
- 'Weak Peripheral Pulse' – слабкий периферійний пульс (так/ні);
- 'Lung rales' – хрипи в легенях (так/ні);
- 'Systolic Murmur' – систолічний шум (так/ні);
- 'Diastolic Murmur' – діастолічний шум (так/ні);
- 'Typical Chest Pain' – типовий біль у грудях (так/ні);
- 'Dyspnea' – задишка (так/ні);
- 'Function Class' – функціональний клас (4 типи);
- 'Atypical' – нетиповий об’єкт (так/ні);
- 'Exertional CP' – фізичний церебральний параліч (так/ні);
- 'Q Wave' – хвилі типу Q (характеристика ЕКГ);
- 'St Elevation' – сильний підйом;
- 'St Depression' – сильна депресія;
- 'Tinversion' – характеристика ЕКГ;
- 'LVH' – гіпертрофія лівого шлуночка (так/ні);
- 'BBB' – гематоенцефалічний бар'єр (немає/лівий/правий);
- 'FBS' – рівень цукру у крові;
- 'CR' – умовний рефлекс
- 'TG' – тигроглобулін;
- 'LDL' – ліпопротеїн низької щільності;
- 'HDL' – ліпопротеїн високої щільності;
- 'BUN' – азот сечовини крові;
- 'ESR' – електронний спіновий резонанс;
- 'HB' – гемоглобін;
- 'K' – калій;
- 'Na' – натрій;
- 'WBC' – білі кров'яні клітини;
- 'Lymph' – лімфа;

– 'Neut' – бікарбонат натрію;
– 'PLT' – пробний тест на лімфоцити;
– 'EF-TTE' – ехокардіограма;
– 'Region RWMA' – Регіон аномального руху стінок (ехокардіограма): 0-5;
– 'VHD' – клапанна хвороба серця (відсутня або одна з трьох степеней серйозності захворювання)
– 'Cath' – індикатор наявності ішемічної хвороби серця, цільова змінна.

### 3.1.2 Застосування методів попередньої обробки даних, зокрема, feature selection, до практичної задачі

Попередня обробка даних включала в себе перетворення категоріальних даних в числові, трансформація даних та відбір ознак (feature selection).

Навіщо потрібен процес відбору ознак? Це процес, в якому ви автоматично вибираєте ті атрибути об’єктів у ваших даних, які найбільшою мірою впливають на зміну, що прогнозується чи будь-який інший результат, який вас цікавить.

Наявність нерелевантних ознак в даних може зменшити точність багатьох моделей, особливо лінійних алгоритмів, таких як лінійна та логістична регресія.

Відбір ознак має три основні переваги:

– Зменшує перенавчання: менша кількість надмірних даних лишає менше можливостей для прийняття рішень на основі шумів.
– Покращує точність: менше даних, що можуть ввести в оману, позитивно впливає на точність моделювання.
– Знижує час налаштування моделі: менше даних означає, що алгоритми тренуються швидше.

Методи відбору ознак (feature selection) можна розділити на дві групи на основі критеріїв оцінки: підходи, що застосовують фільтрацію та ті, яким властиве використання «обгортки». Методи фільтрації не залежать від способу

навчання класифікаторів і використовують такі методи вимірювання, як кореляція, відстань та послідовність в даних, щоб знайти найбільш вдалу підмножину з цілого набору атрибутів [39].

Незважаючи на швидкість цього підходу, він нехтує кореляцією між продуктивністю алгоритму індукції та вибраною підмножиною. В підході «обгортки» [40] попередньо визначений алгоритм навчання використовується для вимірювання ефективності узагальнення за допомогою алгоритму стохастичного пошуку для максимізації точності.

Найбільшим недоліком підходів, що застосовують методи «обгортки», є вимоги для обчислення, оскільки всі можливі піднабори ознак мають бути оцінені з використанням класифікатора. Ненадійність підходів фільтрації і висока обчислювальна вартість методів обгортки стали причиною появи гібридного методу, який використовує евристики для більш якісного процесу відбору релевантних ознак, які після одна за одною будуть оцінюватися моделлю.

В літературі запропоновано багато гібрідних технік, які поєднують ефективний алгоритм відбору ознак з більш точним прогнозуванням. Детальну інформацію про окремі розробки та результати їх застосування можна знайти в [41, 42, 43].

До популярних методів відбору ознак відносяться:

- відбір, що базується на аналізі однієї змінної та сили її зв’язку з цільовою. В такому випадку дослідник сам вирішує, яку кількість змінних залишати. До таких методів відноситься, наприклад, обчислення відхилення для кожної змінної та вибір порогу, який допускає залишити в вибірці тільки ті атрибути, відхилення яких перевищує поріг.
- На противагу попередній техніці, існують підходи, що дозволяють автоматизувати прийняття рішення щодо кінцевої кількості змінних: так звані «жадібні» методи. Наприклад, метод RFECV, що включає рекурсивне видалення атрибутів шляхом ранжування та застосування крос-валідації.

Для зменшення розмірності простору ознак використовують Метод головних компонент (PCA) [44] та лінійно дискримінантний аналіз (LDA).

Окрім цього, алгоритми класифікації, на кшталт дерев рішень, виконують ранжування ознак за степенем їх важливості. Ця інформація також може бути використана як вхідні дані для, наприклад, лінійних методів класифікації.

Результати відбору ознак приведені в табл 3.1

Таблиця 3.1 Feature Selection results

| **feature selection approach** | **number of features** |
|---|---|
| decision trees | 13 |
| rfecv (logistic regression) | 33 |
| LinearSVC | 25 |
| PCA | 20 |

Для f-classif та mutual_info користувач самостійно вказує кількість ознак, яку він бажає залишити.

В роботі застосовувалися 2 способи перетворення даних: стандартизація та мінімаксне масштабування.

Мінімаксна трансформація масштабує і трансформує кожну ознаку індивідуально таким чином, щоб її значення знаходились в заданому діапазоні для навчальної вибірки, тобто від нуля до одиниці.

Враховуючи незбалансованість класів (об’єктів класу «1» - 216, «0» - 87), для уникнення втрати важливої інформації, були обраховані вагові коефіцієнти для кожного з класів.

80% даних були використані для навчання алгоритмів, 20% - для тестування.

## 3.2 Результати застосування простих класифікаторів для вирішення поставленої задачі та їх порівняльний аналіз

Кожна з моделей класифікаторів потребувала налаштування та підбору параметрів. Для підбору використовувалася крос-валідація на 4 блоках. Для покращення якості роботи кожна модель була протестована на початковій та трансформованій вибірці, а також із різними наборами ознак, що стали результатом застосування методів feature selection.

### 3.2.1 Результати застосування логістичної регресії

Результати класифікації (діагностування пацієнта) за допомогою моделі логістичної регресії наведені в табл 3.1.

Таблиця 3.1 – Результати роботи логістичної регресії

| | **accuracy** | **precision** | **recall** | **specificity** | **f1 score** | **roc auc score** |
|---|---|---|---|---|---|---|
| | 0.87 | 0.89 | 0.93 | 0.72 | 0.91 | 0.83 |
| **rescaled (minmax)** | 0.87 | 0.91 | 0.91 | 0.78 | 0.91 | 0.84 |
| **rescaled (stand)** | 0.87 | 0.91 | 0.91 | 0.78 | 0.91 | 0.84 |
| **feature selection (decision trees)** | 0.89 | 0.95 | 0.88 | 0.89 | 0.92 | 0.89 |
| **feature selection (rfecv)** | 0.85 | 0.9 | 0.88 | 0.78 | 0.89 | 0.83 |
| **feature selection (f-classif)** | 0.84 | 0.87 | 0.91 | 0.89 | 0.79 | 0.67 |
| **feature selection (mutual information)** | 0.89 | 0.91 | 0.93 | 0.78 | 0.92 | 0.85 |
| **feature selection (svm)** | 0.87 | 0.89 | 0.93 | 0.72 | 0.91 | 0.83 |

Продовження табл. 3.1

| | | | | | | |
|---|---|---|---|---|---|---|
| **feature selection (pca)** | 0.89 | 0.91 | 0.93 | 0.78 | 0.92 | 0.85 |
| **feature selection (decision trees) + rescaled (minmax)** | 0.89 | 0.93 | 0.91 | 0.83 | 0.92 | 0.87 |
| **feature selection (decision trees) + rescaled (stand)** | 0.9 | 0.97 | 0.88 | 0.94 | 0.93 | 0.91 |
| **feature selection (rfecv) + rescaled (minmax)** | 0.87 | 0.91 | 0.91 | 0.78 | 0.91 | 0.84 |
| **feature selection (rfecv) + rescaled (stand)** | 0.89 | 0.97 | 0.86 | 0.94 | 0.91 | 0.9 |
| **feature selection (f-classif) + rescaled (minmax)** | 0.87 | 0.91 | 0.91 | 0.78 | 0.91 | 0.84 |
| **feature selection (f-classif) + rescaled (stand)** | 0.98 | 0.93 | 0.91 | 0.83 | 0.92 | 0.87 |
| **feature selection (mutual info) + rescaled (minmax)** | 0.85 | 0.89 | 0.91 | 0.72 | 0.9 | 0.81 |
| **feature selection (mutual info) + rescaled (stand)** | 0.84 | 0.92 | 0.84 | 0.83 | 0.88 | 0.84 |
| **feature selection (svm) + rescaled (minmax)** | 0.87 | 0.91 | 0.91 | 0.78 | 0.91 | 0.84 |
| **feature selection (svm) + rescaled (stand)** | 0.85 | 0.9 | 0.88 | 0.78 | 0.89 | 0.83 |
| **feature selection (pca) + rescaled (minmax)** | 0.9 | 0.95 | 0.91 | 0.89 | 0.93 | 0.9 |
| **feature selection (pca) + rescaled (stand)** | 0.85 | 0.93 | 0.86 | 0.83 | 0.89 | 0.85 |

### 3.2.2 Результати застосування наївного Байєсовського класифікатора

Результати класифікації (діагностування пацієнта) за допомогою моделі NB наведені в табл 3.2.

Таблиця 3.2 – Результати роботи наївного байєсовського класифікатора

| | accuracy | precision | recall | specificity | f1 score | roc auc score |
|---|---|---|---|---|---|---|
| | 0.84 | 0.87 | 0.91 | 0.67 | 0.89 | 0.79 |
| **rescaled (minmax)** | 0.84 | 0.87 | 0.91 | 0.67 | 0.89 | 0.79 |
| **rescaled (stand)** | 0.87 | 0.89 | 0.93 | 0.72 | 0.91 | 0.83 |
| **feature selection (decision trees)** | 0.85 | 0.9 | 0.88 | 0.78 | 0.89 | 0.83 |
| **feature selection (rfecv/ f-classif/ mutual-info)** | 0.84 | 0.87 | 0.91 | 0.67 | 0.89 | 0.79 |
| **feature selection (svm)** | 0.84 | 0.84 | 0.95 | 0.56 | 0.89 | 0.75 |
| **feature selection (decision trees) + rescaled (minmax)** | 0.85 | 0.9 | 0.88 | 0.78 | 0.89 | 0.83 |
| **feature selection (decision trees) + rescaled (stand)** | 0.87 | 0.91 | 0.91 | 0.78 | 0.91 | 0.84 |
| **feature selection (rfecv) + rescaled (minmax)** | 0.84 | 0.87 | 0.91 | 0.67 | 0.89 | 0.79 |
| **feature selection (f-classif) + rescaled (stand)** | 0.89 | 0.91 | 0.93 | 0.78 | 0.92 | 0.85 |
| **feature selection (mutual info) + rescaled (minmax)** | 0.84 | 0.87 | 0.91 | 0.67 | 0.89 | 0.79 |
| **feature selection (mutual info) + rescaled (stand)** | 0.85 | 0.87 | 0.93 | 0.67 | 0.9 | 0.8 |
| **feature selection (svm) + rescaled (minmax)** | 0.84 | 0.87 | 0.91 | 0.67 | 0.89 | 0.79 |
| **feature selection (svm) + rescaled (stand)** | 0.87 | 0.89 | 0.93 | 0.72 | 0.91 | 0.83 |
| **feature selection (pca) + rescaled (stand)** | 0.84 | 0.85 | 0.93 | 0.61 | 0.89 | 0.77 |

### 3.2.3 Результати застосування методу *k* найближчих сусідів

Шляхом використання крос-валідації та вимірюванням повноти та точності, було виявлено, що найбільш оптимальним значенням параметра *k* моделі є 14.

Але, не дивлячись на підбір значення параметру кількості сусідів, якість роботи класифікатора на сирих даних доволі погана. Однак, результат вдається покращити, здійснивши оброку даних через трансформацію та відбір ознак (табл 3.3).

Таблиця 3.3 – Результати роботи методу к найближчих сусідів

| | **accuracy** | **precision** | **recall** | **specificity** | **f1 score** | **roc auc score** |
|---|---|---|---|---|---|---|
| | 0.62 | 0.68 | 0.88 | 0 | 0.77 | 0.44 |
| **rescaled (minmax)** | 0.82 | 0.83 | 0.93 | 0.56 | 0.88 | 0.74 |
| **rescaled (stand)** | 0.84 | 0.84 | 0.95 | 0.56 | 0.89 | 0.75 |
| **feature selection (rfecv)** | 0.8 | 0.83 | 0.91 | 0.56 | 0.87 | 0.73 |
| **feature selection (rfecv) + rescaled (stand)** | 0.85 | 0.85 | 0.95 | 0.61 | 0.9 | 0.78 |

### 3.2.4 Результати застосування дерев рішень

Результати класифікації (діагностування пацієнта) за допомогою дерев рішень на чистих та попередньо оброблених даних (відбір ознак та трансформація) наведені в табл 3.4.

Таблиця 3.4 – Результати роботи дерев рішень

| | accuracy | precision | recall | specificity | f1 score | roc auc score |
|---|---|---|---|---|---|---|
| | 0.82 | 0.88 | 0.86 | 0.72 | 0.87 | 0.79 |
| **feature selection (decision trees)** | 0.87 | 0.91 | 0.91 | 0.78 | 0.91 | 0.84 |
| **feature selection (rfecv)** | 0.82 | 0.83 | 0.93 | 0.56 | 0.88 | 0.74 |
| **feature selection (f-classif)** | 0.84 | 0.87 | 0.91 | 0.67 | 0.89 | 0.79 |
| **feature selection (mutual information)** | 0.87 | 0.87 | 0.95 | 0.67 | 0.91 | 0.81 |
| **feature selection (svm)** | 0.84 | 0.87 | 0.91 | 0.67 | 0.89 | 0.79 |
| **feature selection (decision trees) + rescaled (minmax)** | 0.87 | 0.91 | 0.91 | 0.78 | 0.91 | 0.84 |
| **feature selection (decision trees) + rescaled (stand)** | 0.85 | 0.9 | 0.88 | 0.78 | 0.89 | 0.83 |
| **feature selection (rfecv) + rescaled (minmax)** | 0.82 | 0.86 | 0.88 | 0.67 | 0.87 | 0.78 |
| **feature selection (rfecv) + rescaled (stand)** | 0.8 | 0.84 | 0.88 | 0.61 | 0.86 | 0.75 |
| **feature selection (mutual info) + rescaled (minmax)** | 0.72 | 0.84 | 0.74 | 0.67 | 0.79 | 0.71 |
| **feature selection (mutual info) + rescaled (stand)** | 0.85 | 0.87 | 0.93 | 0.67 | 0.9 | 0.8 |
| **feature selection (svm) + rescaled (minmax)** | 0.84 | 0.85 | 0.93 | 0.61 | 0.89 | 0.77 |
| **feature selection (svm) + rescaled (stand)** | 0.84 | 0.85 | 0.93 | 0.61 | 0.89 | 0.77 |
| **feature selection (pca) + rescaled (minmax)** | 0.7 | 0.8 | 0.77 | 0.56 | 0.79 | 0.66 |
| **feature selection (pca) + rescaled (stand)** | 0.69 | 0.8 | 0.74 | 0.56 | 0.77 | 0.65 |

### 3.2.5 Результати застосування методу опорних векторів

Для пошуку оптимального значення параметра *C* та типу ядра був використаний інструмент бібілотеки sklearn.grid_search *GridSearchCV*, що налаштовує параметри, використовуючи крос-валідацію та *accuracy* в якості метрики якості. Отримали наступні результати:

- mean: 0.73267, std: 0.02558, params: {'C': 0.001, 'kernel': 'linear'},
- mean: 0.48185, std: 0.22123, params: {'C': 0.001, 'kernel': 'rbf'},
- mean: 0.76898, std: 0.04200, params: {'C': 0.01, 'kernel': 'linear'},
- mean: 0.48185, std: 0.22123, params: {'C': 0.01, 'kernel': 'rbf'},
- mean: 0.80858, std: 0.05245, params: {'C': 0.1, 'kernel': 'linear'},
- mean: 0.48185, std: 0.22123, params: {'C': 0.1, 'kernel': 'rbf'},
- mean: 0.77888, std: 0.03942, params: {'C': 1, 'kernel': 'linear'},
- mean: 0.71287, std: 0.06380, params: {'C': 1, 'kernel': 'rbf'},
- mean: 0.79208, std: 0.04162, params: {'C': 10, 'kernel': 'linear'},
- mean: 0.71287, std: 0.06380, params: {'C': 10, 'kernel': 'rbf'}

Отже, найбільш доцільним є використання значення параметра *C*, що дорівнює 0.1 та лінійного ядра (табл 3.5).

Таблиця 3.5 – Результати роботи методу опорних векторів

| | **accuracy** | **precision** | **recall** | **specificity** | **f1 score** | **roc auc score** |
|---|---|---|---|---|---|---|
| | 0.85 | 0.9 | 0.88 | 0.78 | 0.89 | 0.83 |
| **feature selection (decision trees)** | 0.89 | 0.91 | 0.93 | 0.92 | 0.85 | 0.78 |
| **feature selection (rfecv)** | 0.87 | 0.93 | 0.88 | 0.9 | 0.86 | 0.83 |
| **feature selection (f-classif)** | 0.89 | 0.93 | 0.91 | 0.83 | 0.92 | 0.87 |

Продовження таблиці 3.5

| | | | | | | |
|---|---|---|---|---|---|---|
| **feature selection (mutual information)** | 0.89 | 0.91 | 0.93 | 0.78 | 0.92 | 0.85 |
| **feature selection (svm)** | 0.8 | 0.86 | 0.86 | 0.67 | 0.86 | 0.76 |
| **feature selection (pca)** | 0.82 | 0.88 | 0.86 | 0.72 | 0.87 | 0.79 |
| **feature selection (decision trees) + rescaled (minmax)** | 0.74 | 0.89 | 0.72 | 0.78 | 0.79 | 0.75 |
| **feature selection (decision trees) + rescaled (stand)** | 0.85 | 0.89 | 0.91 | 0.72 | 0.9 | 0.81 |
| **feature selection (rfecv) + rescaled (minmax)** | 0.8 | 0.92 | 0.79 | 0.83 | 0.85 | 0.81 |
| **feature selection (rfecv) + rescaled (stand)** | 0.9 | 0.95 | 0.91 | 0.89 | 0.93 | 0.9 |
| **feature selection (f-classif) + rescaled (minmax)** | 0.84 | 0.92 | 0.84 | 0.83 | 0.88 | 0.84 |
| **feature selection (f-classif) + rescaled (stand)** | 0.87 | 0.93 | 0.88 | 0.83 | 0.9 | 0.86 |
| **feature selection (mutual info) + rescaled (minmax)** | 0.82 | 0.9 | 0.84 | 0.78 | 0.87 | 0.81 |
| **feature selection (mutual info) + rescaled (stand)** | 0.9 | 0.93 | 0.93 | 0.83 | 0.93 | 0.88 |
| **feature selection (svm) + rescaled (minmax)** | 0.82 | 0.9 | 0.84 | 0.78 | 0.87 | 0.81 |
| **feature selection (svm) + rescaled (stand)** | 0.85 | 0.93 | 0.86 | 0.83 | 0.89 | 0.85 |
| **feature selection (pca) + rescaled (minmax)** | 0.8 | 0.9 | 0.81 | 0.78 | 0.85 | 0.8 |
| **feature selection (pca) + rescaled (stand)** | 0.87 | 0.95 | 0.86 | 0.89 | 0.9 | 0.87 |

### 3.2.6 Результати застосування методу стохастичного градієнтного спуску

Результати класифікації (діагностування пацієнта) за допомогою методу стохастичного градієнтного спуску на чистих та попередньо оброблених даних (відбір ознак та трансформація) наведені в табл 3.6.

Таблиця 3.6 – Результати роботи методу градієнтного спуску

| | **accuracy** | **precision** | **recall** | **specificity** | **f1 score** | **roc auc score** |
|---|---|---|---|---|---|---|
| | 0.72 | 0.77 | 0.86 | 0.39 | 0.81 | 0.62 |
| **rescaled (stand)** | 0.84 | 0.84 | 0.95 | 0.89 | 0.75 | 0.56 |
| **feature selection (decision trees) + rescaled (stand)** | 0.87 | 0.91 | 0.91 | 0.78 | 0.91 | 0.84 |
| **feature selection (rfecv) + rescaled (stand)** | 0.84 | 0.88 | 0.88 | 0.72 | 0.88 | 0.8 |
| **feature selection (f-classif) + rescaled (stand)** | 0.9 | 0.91 | 0.95 | 0.78 | 0.93 | 0.87 |
| **feature selection (mutual info) + rescaled (stand)** | 0.84 | 0.88 | 0.88 | 0.72 | 0.88 | 0.8 |
| **feature selection (svm) + rescaled (stand)** | 0.87 | 0.93 | 0.88 | 0.83 | 0.9 | 0.86 |
| **feature selection (pca) + rescaled (stand)** | 0.85 | 0.89 | 0.91 | 0.72 | 0.9 | 0.81 |

### 3.2.7 Результати застосування лінійно-дискримінантного аналізу (LDA)

Використання алгоритму на даних без відбору ознак, видає попередження про їх колінеарність. У цьому випадку, застосування методів відбору ознак є необхідністю. Результати роботи побудованих моделей наведені в табл. 3.7

Таблиця 3.7 – Результати роботи лінійно-дискримінантного аналізу

| | accuracy | precision | recall | specificity | f1 score | roc auc score |
|---|---|---|---|---|---|---|
| **feature selection (decision trees) + rescaled (minmax)** | 0.92 | 0.93 | 0.95 | 0.83 | 0.94 | 0.89 |
| **feature selection (decision trees) + rescaled (stand)** | 0.89 | 0.91 | 0.93 | 0.78 | 0.92 | 0.85 |
| **feature selection (rfecv) + rescaled (minmax)** | 0.85 | 0.85 | 0.95 | 0.61 | 0.9 | 0.78 |
| **feature selection (rfecv) + rescaled (stand)** | 0.84 | 0.87 | 0.91 | 0.67 | 0.89 | 0.79 |
| **feature selection (f-classif)** | 0.87 | 0.87 | 0.95 | 0.67 | 0.91 | 0.81 |
| **feature selection (mutual info) + rescaled (minmax)** | 0.84 | 0.85 | 0.93 | 0.61 | 0.89 | 0.77 |
| **feature selection (mutual info) + rescaled (stand)** | 0.8 | 0.84 | 0.88 | 0.61 | 0.86 | 0.75 |
| **feature selection (svm) + rescaled (minmax)** | 0.85 | 0.85 | 0.95 | 0.61 | 0.9 | 0.78 |
| **feature selection (svm) + rescaled (stand)** | 0.89 | 0.89 | 0.95 | 0.72 | 0.92 | 0.84 |
| **feature selection (pca) + rescaled (minmax)** | 0.89 | 0.91 | 0.93 | 0.78 | 0.92 | 0.85 |
| **feature selection (pca) + rescaled (stand)** | 0.89 | 0.89 | 0.95 | 0.72 | 0.92 | 0.84 |

## 3.3 Результати застосування ансамблів базових моделей

В якості ансамблів класифікаторів використовувалися наступні моделі: випадковий ліс та XGBoost. Результати роботи наведені в табл. 3.8.

Таблиця 3.8 – Результати роботи ансамблів

| | | Acc | precision | recall | specificity | f1 score | roc auc score |
|---|---|---|---|---|---|---|---|
| **Random Forest** | | 0.9 | 0.89 | 0.98 | 0.72 | 0.93 | 0.85 |
| | **feature selection (decision trees) + rescaled (minmax)** | 0.85 | 0.85 | 0.95 | 0.61 | 0.9 | 0.78 |
| | **feature selection (decision trees) + rescaled (stand)** | 0.89 | 0.89 | 0.95 | 0.72 | 0.92 | 0.84 |
| | **feature selection (f-classif) + rescaled (minmax)** | 0.87 | 0.87 | 0.95 | 0.67 | 0.91 | 0.81 |
| | **feature selection (f-classif) + rescaled (stand)** | 0.85 | 0.84 | 0.98 | 0.56 | 0.9 | 0.77 |
| | **feature selection (mutual info) + rescaled (stand)** | 0.85 | 0.85 | 0.95 | 0.61 | 0.9 | 0.78 |
| | **feature selection (svm) + rescaled (minmax)** | 0.87 | 0.87 | 0.95 | 0.67 | 0.91 | 0.81 |
| | **feature selection (svm) + rescaled (stand)** | 0.9 | 0.89 | 0.98 | 0.72 | 0.93 | 0.85 |
| | **feature selection (pca) + rescaled (minmax)** | 0.85 | 0.84 | 0.98 | 0.56 | 0.9 | 0.77 |
| | **feature selection (pca) + rescaled (stand)** | 0.84 | 0.87 | 0.91 | 0.67 | 0.89 | 0.79 |
| **XGBoost** | | 0.89 | 0.88 | 0.98 | 0.67 | 0.92 | 0.67 |
| **Bagging** | | 0.89 | 0.93 | 0.91 | 0.83 | 0.92 | 0.87 |

## 3.4 Порівняльний аналіз роботи розглянутих моделей

Якщо взяти до уваги природу обраної практичної задачі та значення «0» та «1» у відповідях класифікаторів, можна прийти висновку, що коректна класифікація об’єктів першого класу є, в нашому випадку, більш важливою, адже

краще зайвий раз зробити всі аналізи здоровій людині, аніж не розпізнати хвороби у дійсно хворої. Отже, при порівнянні моделей між собою і виборі кращої варто надавати перевагу оцінці повноти (recall score), адже саме вона оцінює частку правильно класифікованих об'єктів першого класу. А вже в другу чергу звертати увагу на інші характеристики моделей.

Найкращі результати кожної із використаних моделей наведені в табл 3.9.

Таблиця 3.9 Найкращі показники кожного з алгоритмів класифікації

| **Метод** | **accuracy** | **precision** | **recall** | **specificity** | **f1 score** | **roc auc score** |
|---|---|---|---|---|---|---|
| Логістична регресія | 0.89 | 0.91 | 0.93 | 0.78 | 0.92 | 0.85 |
| Наївний байєсовський класифікатор | 0.87 | 0.89 | 0.93 | 0.72 | 0.91 | 0.83 |
| *K* найближчих сусідів | 0.84 | 0.84 | 0.95 | 0.56 | 0.89 | 0.75 |
| Дерева рішень | 0.87 | 0.91 | 0.91 | 0.78 | 0.91 | 0.86 |
| Метод опорних векторів | 0.89 | 0.91 | 0.93 | 0.92 | 0.85 | 0.78 |
| Метод стохастичного град. спуску | 0.87 | 0.91 | 0.91 | 0.78 | 0.91 | 0.84 |
| Лінійно-дискримінатний аналіз | 0.89 | 0.91 | 0.93 | 0.78 | 0.92 | 0.85 |

З таблиці можна побачити, що найкращу оцінку повноти дає метод *k* найближчих сусідів, в той час як найвищу точність мають логістична регресія, метод опорних векторів та лінійно-дискримінантний аналіз. F-міра має найвищий показник для результатів лінійно-дискримінантного аналізу, а площа під roc-кривою найбільша для результатів класифікації з використанням дерев рішень.

## Висновки до розділу

В даному розділі була проведена велика кількість випробувань різних типів класифікаторів та перевірка якості їх роботи при вирішенні практичної задачі діагностики Ішемічної хвороби серця. Для кожного із класифікаторів були визначені оптимальні значення параметрів шляхом застосування перехресної перевірки. Кожна із моделей була протестована як на повному наборі ознак, так і на його піднаборах, що були визначені за допомогою найбільш поширених методів feature selection.

За результатами роботи класифікаторів було виконано порівняльний аналіз та визначено сильні та слабкі сторони використання кожного з них.

Окрім цього, до вирішення задачі було застосовано ансамблі моделей, а саме беггінг всіх вище згаданих класифікаторів та бустинг, зокрема бустинг над деревами рішень (Random Forest) та XGBoost та був проведений порівняльний аналіз отриманих результатів.

# РОЗДІЛ 4. РОЗРОБКА ДВОРІВНЕВОЇ МОДЕЛІ АГРЕГАЦІЇ АНСАМБЛІВ КЛАСИФІКАТОРІВ

## 4.1 Основна ідея та передумови потреби розробки покращеної моделі

У розділі 3 до вирішення практичної задачі, а саме: діагностики пацієнта на рахунок ішемічної хвороби серця, були використані різні моделі. Серед класифікаторів, якість роботи яких була перевірена на даних, відносяться:

- логістична регресія;
- дерева рішень;
- наївний байєсовський класифікатор;
- метод опорних векторі;
- метод *k* найближчих сусідів;
- лінійно-дискримінатний аналіз.

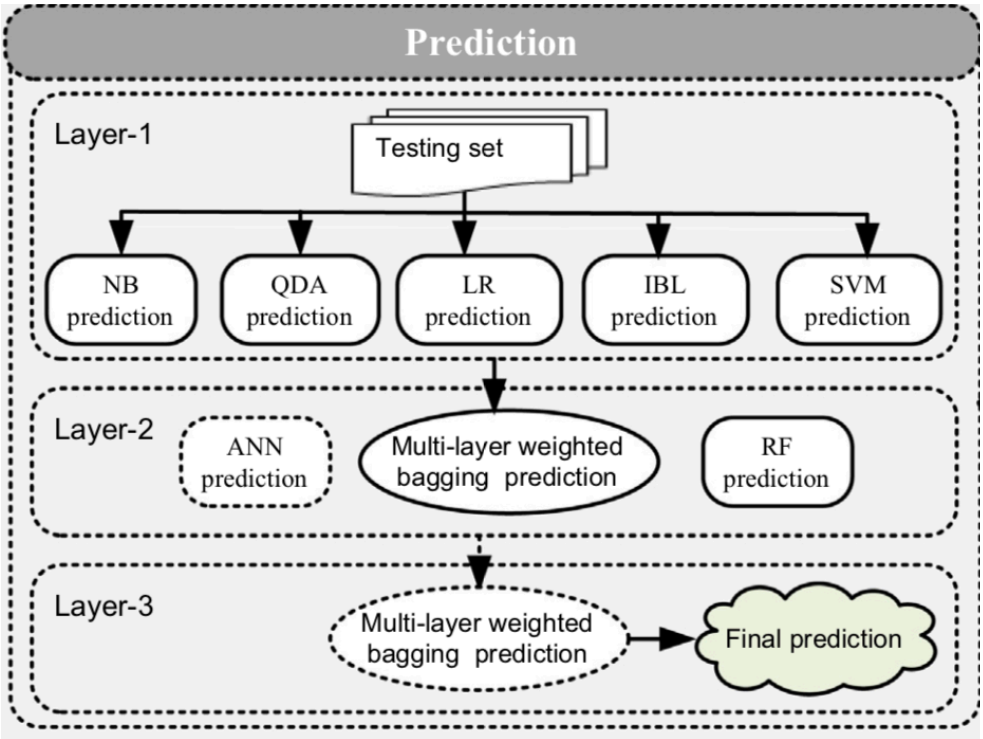
Окрім цього, були застосовані більш складні моделі, в основі яких лежить ідея використання ансамблів простих класифікаторів. Серед використаних ансамблів – Беггінг, XGBoost та RandomForest.

Кожна з моделей була протестована на сирих та трансформованих даних. Окрім цього, класифікатори працювали як на повному наборі ознак, так і на тих, що були обрані різними методами feature selection.

Не зважаючи на доволі непогану якість роботи деяких з моделей, результат все ще потребує покращення, адже мова йде про людські життя. У літературі розглянуто багато різноманітних спроб покращити якість діагностування, та не всі з них успішні.

В роботі [45] автори пропонують систему підтримки прийняття рішень під назвою IntelliHealth, в основі якої знаходиться дворівневий класифікатор HM-BagMoov, що на обох рівнях агрегації використовує зважену суму. Ваги розраховуються на основі значення F-міри кожного з базових алгоритмів. Схематичне зображення запропонованого підходу зображене на рис. 4.1.

Рисунок 4.1 – Система IntelliHealth




Автори [5] стверджують, що застосувавши запропонований ними алгоритм, робота якого була перевірена на 5 різних наборах даних, можна отримати вищі показники accuracy, sensitivity та F-міри в порівнянні з індивідуальними класифікаторами, що використовувалися для діагностування всіх 5 захворювань.

## 4.2 Стекінг як ефективний метод агрегування базових класифікаторів

Стекінг (Stacked Generalization or Stacking) - ще один з найбільш популярних методів створення ансамблів моделей. Використання середнього відповідей декількох простих класифікаторів або навіть зваженого середнього викликає запитання: чому б, власне, не довірити створення ансамблю ще одному алгоритму машинного навчання? Або так званому «метаалгоритму».

Найпростіший варіант стекінгу – Блендинг (Blending). Його ідея полягає в тому, що початкову навчальну вибірку розділяють на дві частини. На першій

виконують навчання простих класифікаторів. Потім отримують їх відповідь для другої половини навчальної вибірки, а також для тестової. Відповідь кожного алгоритма розглядається як нова ознака (або «метаознака»). На метаознаках другої половини навчальної вибірки налаштовують метаалгоритм, який потім запускають на метаознаках тестового піднабору і отримують відповідь. Цей підхід наглядно проілюстровано на рисунку 4.2.

Найбільшим недоліком описаного підходу є розділення навчальної вибірки. Виходить, що ні базові класифікатори, ні метаалгоритм не використовують всієї потужності навчальних даних. Найбільш очевидним шляхом вирішення цієї проблеми є використання декількох блендингів з подальшим усередненням результатів.

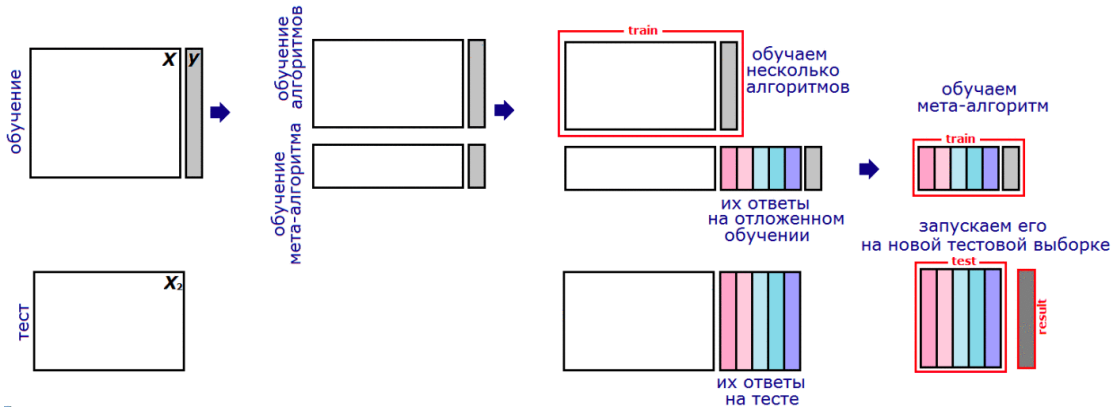



Рисунок 4.2 - Стекінг

Ще одним варіантом, що дозволить використати всю вибірку є використання класичного стекінгу, що проілюстрований нижче на рис 4.3.

Вибірку розбивають на частини, а потім, перебираючи всі піднабори, навчають базові класифікатори, використовуючи в якості навчальної вибірки всі частини, окрім однієї яку залишають для тесту. Метаознаки отримують, навчаючи базові алгоритми на всій навчальній виборці і використовуючи їх відповіді на тестовій. Тут також бажано брати декілька розбиттів, а потім усереднювати отримані результати.

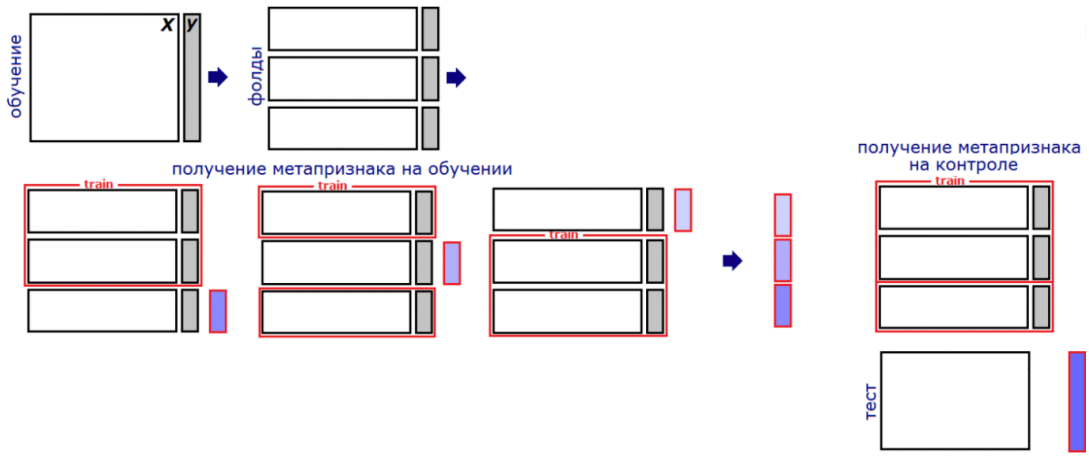



Рисунок 4.3 – Класичний стекінг

Однак, використання класичного стекінгу має один важливий недолік: метаознаки на навчанні, нехай і повній навчальній вибірці, та під час тестування – різні.

Часто зі згаданими недоліками борються за допомогою регуляризації. Якщо в якості метаалгоритму використовується гребенева регресія, то там присутній відповідний параметр. А якщо щось більш складне (наприклад, бустинг над деревами, то до метаознак додають нормальний шум. Саме коефіцієнт, з яким цей шум додають і буде тут певним аналогом коефіцієнта регуляризації.

В роботі [46] приведений приклад системи, що використовує стекінг класифікаторів для прогнозування амінокислотних послідовностей, пов'язаних з раком молочної залози.

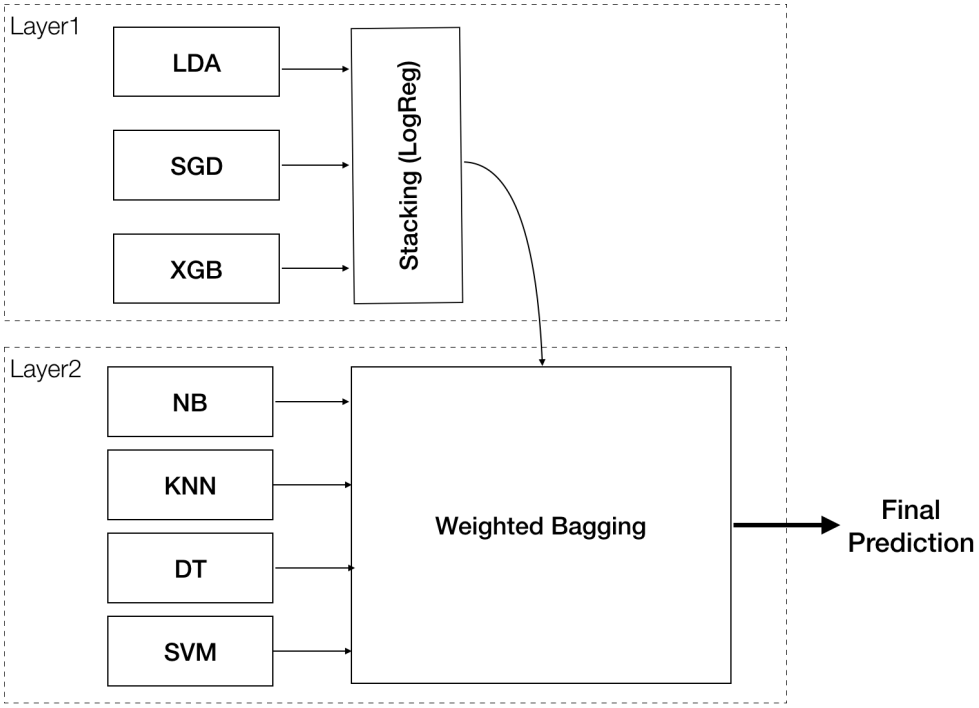
## 4.3 Запропонована модель дворівневої класифікації

Шляхом великої кількості експериментальних випробувань були виявлені можливі шляхи вдосконалення моделі, що була представлена в роботі [45].

Результатом цього наукового дослідження стала дворівнева модель класифікації, агрегація на першому рівні якої відбувається із застосуванням Стекінгу, а на другому рівні – беггінгу.

Схематичне зображення моделі наведене на рис 4.4.

Рисунок 4.4 Запропонований алгоритм класифікації




4.4 Застосування запропонованої моделі до практичної задачі та аналіз результатів

Робота запропонованого алгоритму класифікації була перевірена на двох наборах даних: вибірці для діагнозу Ішемічної хвороби серця, що використовувалася в розділі 3 та діагностиці хронічної хвороби нирок [47].

### 4.4.1 Застосування дворівневої моделі для діагностики хвороби серця

Результати застосування окремих елементів моделі та агрегування на першому рівні наведені в табл. 4.1.

Таблиця 4.1 – Результати роботи першого рівня моделі

| | **accuracy** | **precision** | **recall** | **specificity** | **f1 score** | **roc auc score** |
|---|---|---|---|---|---|---|
| **LDA** | 0.89 | 0.91 | 0.93 | 0.78 | 0.92 | 0.85 |
| **SGD** | 0.87 | 0.91 | 0.91 | 0.78 | 0.91 | 0.84 |
| **XGB** | 0.84 | 0.85 | 0.93 | 0.61 | 0.89 | 0.77 |
| **Stacking (LR)** | 0.9 | 0.93 | 0.93 | 0.83 | 0.93 | 0.88 |

З таблиці можна побачити, що, використовуючи стекінг для комбінування результатів трьох не дуже якісних моделей, можна значно покращити загальний результат. Навчання відповідей алгоритмів відбувалося шляхом використання логістичної регресії.

На наступному рівні моделі результат, отриманий на першому, комбінується з вагами, визначеними через F-міру, з результатами роботи чотирьох різнотипних алгоритмів класифікації, а саме: наївного байєсовського класифікатору, дерев рішень, методу *k* найближчих сусідів та методу опорних векторів (табл 4.2)

Таблиця 4.2 – Результати роботи другого рівня моделі

| | | | | | | |
|---|---|---|---|---|---|---|
| **NB** | 0.85 | 0.9 | 0.88 | 0.78 | 0.89 | 0.83 |
| **KNN** | 0.89 | 0.89 | 0.95 | 0.72 | 0.92 | 0.84 |
| **DT** | 0.87 | 0.91 | 0.91 | 0.78 | 0.91 | 0.84 |
| **SVM** | 0.85 | 0.9 | 0.88 | 0.78 | 0.89 | 0.83 |
| **Stacking (LR)** | 0.9 | 0.93 | 0.93 | 0.83 | 0.93 | 0.88 |
| **Bagging** | 0.92 | 0.93 | 0.95 | 0.83 | 0.94 | 0.89 |

Використовуючи ансамбль всіх згаданих моделей нам вдалося використати сильні сторони кожної з них і отримати результат, що за загальними результатами по всім метрикам (точність, повнота, чутливість, F-міра та площа під roc-кривою) працює краще ніж кожна окремо взята модель.

### 4.4.1 Застосування дворівневої моделі для діагностики хвороби нирок

Для перевірки універсальності запропонованого алгоритмі, було прийняте рішення перевірити його роботу на іншому наборі даних, основною ціллю якого є діагностика хронічної хвороби нирок.

Цей набір даних містить 400 записів (об’єктів), кожен з яких містить 25 атрибутів. Датасет також потребував попередньої обробки: заповнення пропусків, зміни типу ознак з текстових на категоріальні та трансформації ознак. Розділення на навчальну та тестову вибірки відбувалося наступним чином: 70% - навчальна і 30% - тестова. Результати застосування окремих елементів моделі та агрегування на першому рівні наведені в табл. 4.3.

Таблиця 4.3 – Результати роботи першого рівня моделі (хвороба нирок)

| | **accuracy** | **precision** | **recall** | **specificity** | **f1 score** | **roc auc score** |
|---|---|---|---|---|---|---|
| **LDA** | 0.91 | 1 | 0.87 | 1 | 0.93 | 0.93 |
| **SGD** | 0.81 | 0.91 | 0.78 | 0.86 | 0.84 | 0.82 |
| **XGB** | 0.95 | 0.99 | 0.94 | 0.99 | 0.89 | 0.95 |
| **Stacking (LR)** | 0.96 | 1 | 0.95 | 1 | 0.95 | 0.95 |

Як бачимо з результатів роботи окремих класифікаторів, цей набір даних має трохи іншу природу, здорових людей за присутніми ознаками виявити досить легко, тому оцінка точності дуже близька або навіть дорівнює одиниці.

Але нас більше цікавить підвищення значення оцінки повноти, тобто безпомилкового визначення хворих людей поміж здорових.

На наступному рівні моделі результат, отриманий на першому, комбінується з вагами, визначеними через F-міру, з результатами роботи чотирьох різнотипних алгоритмів класифікації, а саме: наївного байєсовського класифікатору, дерев рішень, методу $k$ найближчих сусідів та методу опорних векторів (табл 4.4)

Таблиця 4.4 – Результати роботи другого рівня моделі (хвороба нирок)

| | | | | | | |
|---|---|---|---|---|---|---|
| **NB** | 0.93 | 1 | 0.88 | 1 | 0.94 | 0.94 |
| **KNN** | 0.95 | 1 | 0.92 | 1 | 0.96 | 0.96 |
| **DT** | 0.97 | 1 | 0.94 | 1 | 0.95 | 0.98 |
| **SVM** | 0.93 | 1 | 0.88 | 1 | 0.94 | 0.94 |
| **Stacking (LR)** | 0.96 | 1 | 0.95 | 1 | 0.95 | 0.95 |
| **Bagging** | 0.97 | 1 | 0.96 | 1 | 0.98 | 0.98 |

Використовуючи ансамбль всіх згаданих моделей нам вдалося використати сильні сторони кожної з них і отримати результат, що за загальними результатами по всім метрикам (точність, повнота, чутливість, F-міра та площа під roc-кривою) працює трохи краще ніж кожна окремо взята модель.

Висновки до розділу

У даному розділі був зроблений огляд запропонованої в літературі дворівневої моделі класифікації та запропонований метод її модифікації. Був запропонований власний алгоритм класифікації, що передбачає стекінг результатів роботи трьох різних моделей на першому рівні та беггінг (зважене комбінування) результатів роботи шести моделей на другому рівні.

Якість роботи запропонованої моделі була продемонстрована та виміряна за допомогою основних метрик якості, таких як точність, повнота, чутливість, f-міра, площа під roc-кривою та іншими.

Для демонстрації узагальнюючих властивостей алгоритму, його робота була перевірена на додатковому наборі даних, що призначений для діагностики хронічної хвороби нирок.

# РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

## 5.1 Опис ідеї проекту (товару, послуги, технології)

Описі ідеї проекту представлений таблицями 5.1 та 5.2

Таблиця 5.1. Опис ідеї стартап-проекту

| Зміст ідеї | Напрямки застосування | Вигоди для користувача |
|---|---|---|
| Інтелектуальна система підтримки прийняття рішень для медичного діагностування, робота якої базується на моделях машинного інтелекту | 1. Державні медичні установи | Підвищення точності діагнозів, зменшення ймовірності помилки, вищий рівень впевненості лікаря в прийнятому рішенні. Більше врятованих життів як результат вчасного та точного виявлення хвороби в пацієнта. |
| | 2. Приватні клініки | Надання більш якісних медичних послуг. Підвищення точності діагнозів, зменшення ймовірності помилки, вищий рівень впевненості лікаря в прийнятому рішенні. Більше врятованих життів як результат вчасного та точного виявлення хвороби в пацієнта. |
| | 3. Приватні лікарі | Підвищення точності діагнозів, зменшення ймовірності помилки, вищий рівень впевненості лікаря в прийнятому рішенні. Більше врятованих життів як результат вчасного та точного виявлення хвороби в пацієнта. |

Таблиця 5.2. Визначення сильних, слабких та нейтральних характеристик ідеї проекту

| № п/п | Техніко-економічні характеристики ідеї | (потенційні) товари/концепції конкурентів | | | | W (слабка сторона) | N (нейтральна сторона) | S (сильна сторона) |
|---|---|---|---|---|---|---|---|---|
| | | Мій проект | «Здоров'я сім'ї 1.0» | IntelliHealth | MediFAM | | | |
| 21 | Швидкість роботи | висока | середня | висока | середня | | | + |
| 32 | Спектр можливостей | широкий | обмежений | широкий | дуже обмежений | | | + |
| 33 | Точність результатів | висока | вище середнього | висока | середня | | | + |
| 44 | Вартість впровадження | висока | середня | висока | низька | + | | |
| 5 | Зручність користування | зручна | зручна | зручна | зручна | | + | |
| 6 | Трудомісткість виготовлення | висока | середня | висока | низька | | + | |

## 5.2 Технологічний аудит ідеї проекту

Був проведений технологічний аудит ідей проекту, результати якого представлені в табл. 5.3.

Таблиця 5.3. Технологічна здійсненність ідеї проекту

| № п/п | Ідея проекту | Технології її реалізації | Наявність технологій | Доступність технологій |
|---|---|---|---|---|
| 1 | Розробка повноцінної та автономної системи підтримки прийняття рішень | Інтегроване середовище розробки для мови програмування Python | + | + |
| 2 | Реалізація інтелектуальних методів аналізу даних | Спеціалізовані бібліотеки Python | + | + |
| 3 | Збереження та обробка великих масивів даних | Будь-яка сучасна СКБД | + | + |
| 4 | Імплементація методів штучного інтелекту для підтримки прийняття рішення щодо діагностування пацієнта, а саме: побудова багаторівневої моделі-класифікатора | Спеціалізовані бібліотеки Python | +/- необхідно доробити задля підвищення точності результатів | + |
| 5 | Висока точність результатів | Достатня кількість «історичних» випадків для навчання алгоритму | + | +/- |

Обрана технологія реалізації ідеї проекту:

Майже всі необхідні технології є доступними авторам проекту. Єдину складність може становити збір реальних даних пацієнтів для навчання алгоритму, однак, дана перешкода може бути подолана завдяки відкритим медичним даним в репозиторіях, яких стає дедалі більше, а також запити про анонімізовані дані пацієнтів до лікарень світу.

## 5.3 Аналіз ринкових можливостей запуску стартап-проекту

Результати проведення аналізу ринкових можливостей запуску проекту представлені в табл 5.4 – 5.13

Таблиця 5.4. Попередня характеристика потенційного ринку стартап-проекту

| № п/п | Показники стану ринку (найменування) | Характеристика |
|---|---|---|
| 1 | Кількість головних гравців, од | 2 |
| 2 | Загальний обсяг продаж, грн/ум.од | - |
| 3 | Динаміка ринку (якісна оцінка) | Зростає |
| 4 | Наявність обмежень для входу (вказати характер обмежень) | Технологічні (обмеженість інформаційних технологій), надвеликі об’єми даних, наявність пропусків у даних, необхідність налаштування ПП під конкретну галузь, проблема розрізнених (неструктурованих) даних. Неповноцінна система знань про існуючі |

Продовження таблиці 5.4

| | | |
|---|---|---|
| | | захворювання та їх симптоми. |
| 5 | Специфічні вимоги до стандартизації та сертифікації | |
| 6 | Середня норма рентабельності в галузі (або по ринку), % | 30 % |

Таблиця 5.5. Характеристика потенційних клієнтів стартап-проекту

| Потреба, що формує ринок | Цільова аудиторія (цільові сегменти ринку) | Відмінності у поведінці різних потенційних цільових груп клієнтів | Вимоги споживачів до товару |
|---|---|---|---|
| Потреба точної та швидкої постановки діагнозу пацієнтові, що базується на ряді персональних характеристик, медичних показників, симптомів та результатів аналізів. | Державні та приватні медичні установи. | Користувачі очікують, що програмний продукт матиме інтуїтивно-зрозумілий інтерфейс, буде зручним у використанні, проводитиме обробку та аналіз даних швидко та якісно, забезпечуватиме результат високої точності. | Компанія-постачальник має забезпечити користувачів неперервною підтримкою під час впровадження системи на підприємстві та під час безпосереднього використання продукту. |

Таблиця 5.6. Фактори загроз

| № п/п | Фактор | Зміст загрози | Можлива реакція компанії |
|---|---|---|---|
| 1 | Припущення помилок програмою | Не дивлячись на загалом підвищені показники точності та швидкості постановки діагнозу, алгоритм не може давати 100% гарантії, що відповідь класифікатора є вірною. | Серед багатьох на це причин варто перш за все мати на увазі, що помилки алгоритму можуть спричинені великою часткою помилкових діагнозів в історичних даних, на яких відбувалося навчання, недостатніми знаннями про захворювання та їх симптоми, похибками вимірювальних приладів, тощо. Отже компанія має заздалегідь попереджати користувачів про степінь впевненості в відповіді програми. |
| 2 | Постачальники | Постачальники планують підвищити ціну на ліцензійне програмне забезпечення, необхідне для реалізації та підтримки проекту | Пошук більш дешевих замінників. Вести перемови з постачальниками з приводу фіксування ціни за умов довготривалої співпраці. |
| 3 | Конкуренція | Швидкий темп розвитку сфери аналізу даних та можливостей взаємодії з медичною сферою сприяє представленню нових покращених | Неперервно вести дослідження та випускати нові покращені версії продукту. |

Таблиця 5.7. Фактори можливостей

| № п/п | Фактор | Зміст можливості | Можлива реакція компанії |
|---|---|---|---|
| 1 | Припущення помилок програмою | У сфері медицину та, зокрема, діагностування неперервно ведуться дослідження окремих захворювань та їх симптомів, а також вивчення можливостей автоматизації прийняття рішень. | Налагодження зв'язків з медичними інститутами України та країн ЄС з ціллю співробітництва, слідкування за публікаціями та останніми розробками в сфері. |
| 2 | Конкуренція | Існують наукові лабораторії, що займаються найновішими розробками в області Машинного навчання та аналізу даних. | Налагодження зв'язків з інститутами України та країн ЄС з ціллю співробітництва. |
| 3 | Постачальники | Поява нових постачальників, з більш вигідними пропозиціями ресурсів | Налагодити комунікацію з потенційними партнерами |

Таблиця 5.8. Ступеневий аналіз конкуренції на ринку

| Особливості конкурентного середовища | В чому проявляється дана характеристика | Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною) |
|---|---|---|
| 1. Вказати тип конкуренції<br>- олігополія | В галузі домінує невелика кількість конкуруючих фірм, при цьому хоча б одна або дві з них, виробляють значну долю продукції даної галузі | Пропозиція продукту, що відрізнятиметься від вже існуючих на ринку. |

Продовження таблиці 5.8

| | | |
|---|---|---|
| 2. За рівнем конкурентної боротьби<br>- національний | Конкуренти діють на національному рівні – намагаючись забезпечити впровадження свого продукту на якомога більшій кількості вітчизняних підприємств. | Пропозиція продукту, що відрізнятиметься від вже існуючих на ринку. |
| 3. За галузевою ознакою<br>- внутрішньогалузева | Має місце суперництво між окремими підприємцями і фірмами однієї галузі щодо одержання прибутку. | Підвищення продуктивності праці, зменшення витрат виробництва, зниження індивідуальної цінності товару. |
| 4. Конкуренція за видами товарів:<br>- товарно-родова | Різноманітні шляхи задоволення конкретного бажання. Це конкуренція між товарами одного виду – різні базові інструменти обробки та аналізу даних виконують приблизно одні й ті самі функції. | Використовувати цінові та нецінові методи конкурентної боротьби на ринку. |
| 5. За характером конкурентних переваг<br>- нецінова | Проводиться головним чином за допомогою вдосконалення якості продукції, якості отриманих результатів, інновацій та найновітніших технологій. | Пропозиція продукту, що відрізнятиметься від вже існуючих на ринку. |
| 6. За інтенсивністю<br>- не марочна | Торгова марка не відіграє основної ролі. | Немає необхідності вкладати кошти у створення та розкрутку бренду |

Таблиця 5.9. Аналіз конкуренції в галузі за М. Портером

| | | | | | |
|---|---|---|---|---|---|
| Складові аналізу | Прямі конкуренти в галузі | Потенційні конкуренти | Постачальники | Клієнти | Товари-замінники |
| | «Здоров'я сім'ї 1.0», IntelliHealth, MediFAM | - | -Ціна на ресурси<br>- Готовність до співпраці<br>-Якість ресурсів | - Очікування щодо якості обслуговування<br>- Вплив покупців на ринкову ціну | Поява відносно дешевих або більш зручних у використанні замінників |
| Висновки | З боку прямих конкурентів немає сильної загрози, адже продукти «Здоров'я сім'ї 1.0» та MediFAM мають набагато вужчий функціонал. А не дивлячись на те, що IntelliHealth є дійсно сильним конкурентом, на ринку України цей продукт зовсім відсутній. | Є можливості виходу на ринок в найближчі строки | Постачальники контролюють ціни на ресурси та їхню якість, а саме програмного забезпечення. | Ні | Обмежень немає |

Таблиця 5.10. Обґрунтування факторів конкурентоспроможності

| № п/п | Фактор конкурентоспроможності | Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим) |
|---|---|---|
| 1 | Інновації | Інноваційні рішення мають забезпечити перевагу нашим клієнтам над конкурентами |
| 2 | Відсутність продуктів з аналогічним функціоналом на вітчизняному ринку | Актуальність проблеми, рішення до якої пропонує продукт та відсутність замінників на вітчизняному ринку. |
| 3 | Націленість на медичні установи державного та приватного сектору | Орієнтація на компанії певних галузей |
| 4 | Цінова політика | Справедливе співвідношення ціна-якість |

Таблиця 5.11. Порівняльний аналіз сильних та слабких сторін «назва проекту»

| № п/п | Фактор конкурентоспроможності | Бали 1-20 | Рейтинг товарів-конкурентів у порівнянні з IDS | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | –3 | –2 | –1 | 0 | +1 | +2 | +3 |
| 1 | Інновації | 18 | | | | | | | + |
| 2 | Відсутність продуктів з аналогічним функціоналом на вітчизняному ринку | 20 | | | | | | | + |
| 3 | Націленість на медичні установи державного та приватного сектору | 15 | | | | + | | | |
| 4 | Цінова політика | 15 | | + | | | | | |

Таблиця 5.12. SWOT- аналіз стартап-проекту

| | |
|---|---|
| Сильні сторони: інноваційність, унікальність, актуальність, націленість як на державний, так і на приватний сектор | Слабкі сторони: цінова політика, швидкий темп розвитку галузі |
| Можливості: впровадження інноваційних рішень, розширення бізнесу, вихід на міжнародний ринок | Загрози: поява сильних конкурентів на вітчизняному ринку |

Таблиця 5.13. Альтернативи ринкового впровадження стартап-проекту

| № п/п | Альтернатива (орієнтовний комплекс заходів) ринкової поведінки | Ймовірність отримання ресурсів | Строки реалізації |
|---|---|---|---|
| 1 | Розвиток товару на ринку, що вже існує | Середня | 6-9 міс. |
| 2 | Розвиток нового ринку з наявним товаром | Середня | 10 міс. |
| 3 | Впровадження нового товару і створення нового ринку | Низька | 16 міс. |

Обираємо першу альтернативу, для якої ймовірність отримання ресурсів є не нижчою за показники для інших альтернатив, а строк реалізації – відносно невеликий.

## 5.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії проекту запропоноване в табл 5.14 – 5.17

Таблиця 5.14. Вибір цільових груп потенційних споживачів

| № п/п | Опис профілю цільової групи потенційних клієнтів | Готовність споживачів сприйняти продукт | Орієнтовний попит в межах цільової групи (сегменту) | Інтенсивність конкуренції в сегменті | Простота входу у сегмент |
|---|---|---|---|---|---|
| 1 | Державний сектор медичного обслуговування | Середня | Помірний | Низька | Середня |
| 2 | Приватний сектор медичного обслуговування | Висока | Помірний | Середня | Середня |

Які цільові групи обрано: державний та приватний сектор медичного обслуговування

Таблиця 5.15. Визначення базової стратегії розвитку

| Обрана альтернатива розвитку проекту | Стратегія охоплення ринку | Ключові конкурентоспроможні позиції відповідно до обраної альтернативи | Базова стратегія розвитку |
|---|---|---|---|
| Розвиток товару на ринку, що вже існує | Ексклюзивний розподіл | Позиціювання за співвідношенням "ціна – якість", позиціювання на основі порівняння товару фірми з товарами конкурентів та наголошення на його можливостях, унікальних для ринку України | Стратегія диференціації |

Таблиця 5.16. Визначення базової стратегії конкурентної поведінки

| Чи є проект «першопрохідцем» на ринку? | Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів? | Чи буде компанія копіювати основні характеристики товару конкурента, і які? | Стратегія конкурентної поведінки |
|---|---|---|---|
| Ні | Забирати у існуючих конкурентів | Частково, базові характеристики | Стратегія лідера |

Таблиця 5.17. Визначення стратегії позиціонування

| Вимоги до товару цільової аудиторії | Базова стратегія розвитку | Ключові конкурентоспроможні позиції власного стартап-проекту | Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових) |
|---|---|---|---|
| - Зручність у використанні<br>-Багатофункціональність<br>- Висока точність результатів роботи<br>- Відсутність помилок. | Стратегія диференціації | На основі специфічних відчутних характеристик | Позиціювання за співвідношенням "ціна – якість", позиціювання на основі порівняння товару фірми з товарами конкурентів та наголошення на його можливостях, унікальних для ринку України |

5.5 Розроблення маркетингової програми стартап-проекту

Таблиця 5.18. Визначення ключових переваг концепції потенційного товару

| № п/п | Потреба | Вигода, яку пропонує товар | Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити |
|---|---|---|---|
| | Надійність результатів аналізу та побудованих прогнозів | Використання багаторівневого ансамблю класифікаторів задля забезпечення вищої якості результатів роботи продукту. | Використання не одного, а декількох базових класифікаторів, а також ансамблів моделей та декількох рівнів ієрархії. |
| | Робота з великими об’ємами медичних даних | Висока швидкість роботи продукту, зручний інтерфейс | Більш інтуїтивно-зрозумілий інтерфейс, короткий час навчання/адаптації до продукту; Широкий функціонал |

Таблиця 5.19. Опис трьох рівнів моделі товару

| Рівні товару | Сутність та складові |
|---|---|
| I. Товар за задумом | Потреба медичних установ державного та приватного секторів підвищити точність діагностування. Основу продукту становить інноваційний алгоритм, що являє собою багаторівневу структуру, кожний рівень якої – ансамбль базових класифікаторів. Система, що працює разом зі спеціалістом, націлена на діагностування пацієнта по ряду його персональних характеристик, медичних вимірів, результатів аналізів та інших симптомів. |
| II. Товар у реальному виконанні | Якість: оцінка точності прогнозу з використанням метрик оцінки точності роботи бінарних класифікаторів, порівняння з роботою інших алгоритмів на тих самих даних. |
| | Марка: IDS (Intelligent Diagnostic System) |
| III. Товар із підкріпленням | До продажу: тестовий період |
| | Після продажу: допомога в інтеграції продукту на підприємстві; семінар для співробітників з приводу користування системою. Гарантійне обслуговування. |
| За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності | |

Таблиця 5.20. Формування системи збуту

| Специфіка закупівельної поведінки цільових клієнтів | Функції збуту, які має виконувати постачальник товару | Глибина каналу збуту | Оптимальна система збуту |
|---|---|---|---|
| Цільові клієнти, як правило, купують 1 одиницю товару (програмний продукт) для довгострокового користування. | Зберігання, інформування, встановлення контакту, транспортування. | 0-1 | Власні засоби збуту |

Таблиця 5.21. Концепція маркетингових комунікацій

| Специфіка поведінки цільових клієнтів | Канали комунікацій, якими користуються цільові клієнти | Завдання рекламного повідомлення | Концепція рекламного звернення |
|---|---|---|---|
| Цільові клієнти з великою ймовірністю вже користуються певним ПП для цих цілей. SMM – не є ефективним засобом впливу на цільових клієнтів. | Контент-маркетинг, рекомендації знайомих, прямий зв'язок з цільовими клієнтами. | Привернути увагу клієнта, ознайомити з усіма перевагами використання продукту. | Презентація переваг з акцентом на інноваційності та унікальності продукту, а також співвідношення ціна-якість. |

Висновки до розділу

Отже, в цьому розділі був проведений маркетинговий аналіз стартап проекту на базі запропонованого в дисертації науково-технічного рішення з ціллю визначення принципової можливості його ринкового впровадження та можливих напрямків реалізації цього впровадження.

За результатами аналізу можна стверджувати, що запропоноване рішення має високий потенціал ринкової комерціалізації: сфера медичного обслуговування завжди була та буде актуальною, на ринку наявний постійний попит на продукти, що можуть автоматизувати процеси прийняття рішень в медицині, тим самим зменшивши вплив людського фактору і, як результат, частоту помилки.

До потенційних груп клієнтів відносяться як державні, так і приватні медичні установи. Однак, для цих двох цільових груп, фактори ризику відрізняються. Наприклад, цінова політика може стати бар'єром входження на ринок державних лікарень внаслідок недостатнього фінансування урядом.

На даний момент на ринку України представлені декілька систем підтримки роботи лікарів, та не одна з них не фокусується саме на діагностуванні. Окрім цього, не один продукт, представлений на вітчизняному ринку програмного забезпечення в медичній сфері, не використовує інноваційні методи аналізу даних, а саме – багаторівневі ансамблі алгоритмів машинного навчання. Сильним конкурентом могла б бути запропонована Saba Bashir, Usman Qamar та Farhan Hassan Khan система IntelliHealth [ ], але вона відсутня на ринку України та Європейського Союзу загалом. Отже, продукт є конкурентно спроможним для виходу на вітчизняний ринок.

Для ринкової реалізації проекту доцільно обрати розвиток товару на ринку, що вже існує як альтернативу впровадження через значну ймовірність отримання ресурсів та відносно недовгий строк реалізації.

Подальша імплементація та розвиток проекту є доцільною, адже сфера штучного інтелекту розвивається надзвичайно швидко, а її застосування в сфері медицини є дуже перспективними.

# ВИСНОВКИ ПО РОБОТІ І РЕКОМЕНДАЦІЇ ДЛЯ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Сучасні технології дозволяють покращити рівень життя людства в різних сферах і медицина не є винятком.

В роботі було розглянуто питання актуальності теми, що досліджується, а саме: застосування методів інтелектуального аналізу даних для діагностування хвороби в пацієнта за набором показників, таких як симптоми, результати аналізів, демографічних показників і тд.

Для досягнення цих цілей було вирішено такі задачі:

- По-перше, досліджена теоретична база задач, пов'язаних з задачею бінарної класифікації, що включало в себе огляд і аналіз літератури.
- По-друге, в роботі був проведений огляд та опис найбільш поширених методів підготовки даних для подальшої роботи з ними: попередньої обробки даних, відбору ознак та трансформації.
- Був проведений огляд основних моделей класифікації та їх ансамблів, наведені недоліки та переваги роботи кожної. Ефективність роботи кожної була продемонстрована як оцінка якості результатів застосування для вирішення практичної задачі.
- По-третє, в роботі була запропонована покращена модель дворівневої класифікації та наведені результати застосування до вирішення практичної задачі та проведений їх порівняльний аналіз.

В майбутньому рекомендується продовжити дослідження в даному напрямі з метою удосконалення моделі класифікації та покращення її узагальнюючих властивостей, а саме: розширити множину методів класифікації, додати використання нейромереж.

# ПЕРЕЛІК ПОСИЛАНЬ


1. Фармацевтична Енциклопедія [Електронний ресурс]: / уклад. Стречень С.Б – Режим доступу: http://www.pharmencyclopedia.com.ua/article/2526/diagnostika
2. Daniel M. Study Suggests Medical Errors Now Third Leading Cause of Death in the U.S. [Електронний ресурс] / Daniel M. – Режим доступу: https://www.hopkinsmedicine.org/news/media/releases/study_suggests_medical_errors_now_third_leading_cause_of_death_in_the_us
3. English Oxford Dictionary [Електронний ресурс] – Режим доступу: https://en.oxforddictionaries.com/definition/big_data
4. Koktysh L. The state of the art in health data analytics [Електронний ресурс] / Koktysh L. – Режим доступу: https://www.scnsoft.com/blog/health-data-analytics-overview
5. Gemp I. Automated Data Cleansing through Meta-learning [Електронний ресурс] / Gemp I., Theocharous G., Ghavamzadeh M., – Association for the Advancement of Artificial Intelligence, 2017– Режим доступу: https://people.cs.umass.edu/~imgemp/pubs/iaai_2017.pdf
6. Armstrong S. Data, data everywhere: the challenges of personalised medicine [Електронний ресурс] / Armstrong S. – Режим доступу: https://www.bmj.com/content/359/bmj.j4546
7. The Wirecutter. What is Alexa? What is the Amazon Echo, and should you get one? [Електронний ресурс] – Режим доступу: https://thewirecutter.com/reviews/what-is-alexa-what-is-the-amazon-echo-and-should-you-get-one/
8. Bresnick J. How to Choose the Right Healthcare Big Data Analytics Tools [Електронний ресурс] / Bresnick J. – Режим доступу: https://healthitanalytics.com/features/how-to-choose-the-right-healthcare-big-data-analytics-tools

9. Bresnick J. Imaging Analytics Get Big Data Boost from New Partnerships [Електронний ресурс] / Bresnick J. – Режим доступу: https://healthitanalytics.com/news/imaging-analytics-get-big-data-boost-from-new-partnerships
10. Bresnick J. IBM Watson Expands Role in Imaging Analytics, Population Health [Електронний ресурс] / Bresnick J. – Режим доступу: https://healthitanalytics.com/news/ibm-watson-expands-role-in-imaging-analytics-population-health
11. Bresnick J. Google's Machine Learning, Imaging Analytics Flag Breast Cancer [Електронний ресурс] / Bresnick J. – Режим доступу: https://healthitanalytics.com/news/googles-machine-learning-imaging-analytics-flag-breast-cancer
12. Bresnick J. Data Governance Key to Hospital's Natural Language Query Project [Електронний ресурс] / Bresnick J. – Режим доступу: https://healthitanalytics.com/news/data-governance-key-to-hospitals-natural-language-query-project
13. Bresnick J. Machine Learning, NLP Help with Physician Skill Benchmarking [Електронний ресурс] / Bresnick J. – Режим доступу: https://healthitanalytics.com/news/machine-learning-nlp-help-with-physician-skill-benchmarking
14. Bresnick J. Mount Sinai Uses Machine Learning for Heart Imaging Analytics [Електронний ресурс] / Bresnick J. – Режим доступу: https://healthitanalytics.com/news/mount-sinai-uses-machine-learning-for-heart-imaging-analytics
15. Bresnick J. UCSF to Develop Machine Learning for CDS, Imaging Analytics [Електронний ресурс] / Bresnick J. – Режим доступу: https://healthitanalytics.com/news/ucsf-to-develop-machine-learning-for-cds-imaging-analytics
16. IBM Watson Health [Електронний ресурс] – Режим доступу: https://www.ibm.com/watson/health/oncology-and-genomics/oncology/

17. Rajpurkar P. Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks [Електронний ресурс] / Rajpurkar P., Hannun A., Haghpanahi M., Bourn C., NG A.Y. – Stanford – Режим доступу: https://arxiv.org/pdf/1707.01836.pdf
18. Predicting Response to Depression Treatment (PReDicT) project [Електронний ресурс] – Режим доступу: http://p1vital.com/ehealth/?p=376
19. Payan A. Predicting Alzheimer's disease: a neuroimaging study with 3D convolutional neural networks [Електронний ресурс] / Payan A., Montana G. – Режим доступу: https://arxiv.org/abs/1502.02506
20. Mukherjee S. A.I. VERSUS M.D. What happens when diagnosis is automated? [Електронний ресурс] / Mukherjee S. – Режим доступу: https://www.newyorker.com/magazine/2017/04/03/ai-versus-md
21. Naumov L. Main Problems of Modern Medicine in Diagnostics and learning: Ways of Optimal Solution [Електронний ресурс] / Naumov L.: Ana Kar Der. – 2001.– pp. 166-178 – Режим доступу: https://www.journalagent.com/anatoljcardiol/pdfs/AnatolJCardiol_1_3_166_178.pdf
22. Арустамов А. Статья: Предобработка и очистка данных перед загрузкой в хранилище [Електронний ресурс] / Арустамов А. — Режим доступа:http://sysdba.org.ua/proektirovanie-bd/etl/predobrabotka-i-ochistka-dannyih-pered-zagruzkoy-v-hranilische.html
23. Фоурино Р. Электронное качество данных: скрытая перспектива очистки данных [Електронний ресурс] / Фоурино Р.— Режим доступа: http://www.iso.ru/print/rus/document5820.phtml
24. Воронцов К.В. Машинное обучение: курс лекций [Електронний ресурс] / Воронцов К.В. — Режим доступа: http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_%28%D0%BA%D1%83%D1%80%D1%81_%D0%BB%D0%B5%D0%BA%D1

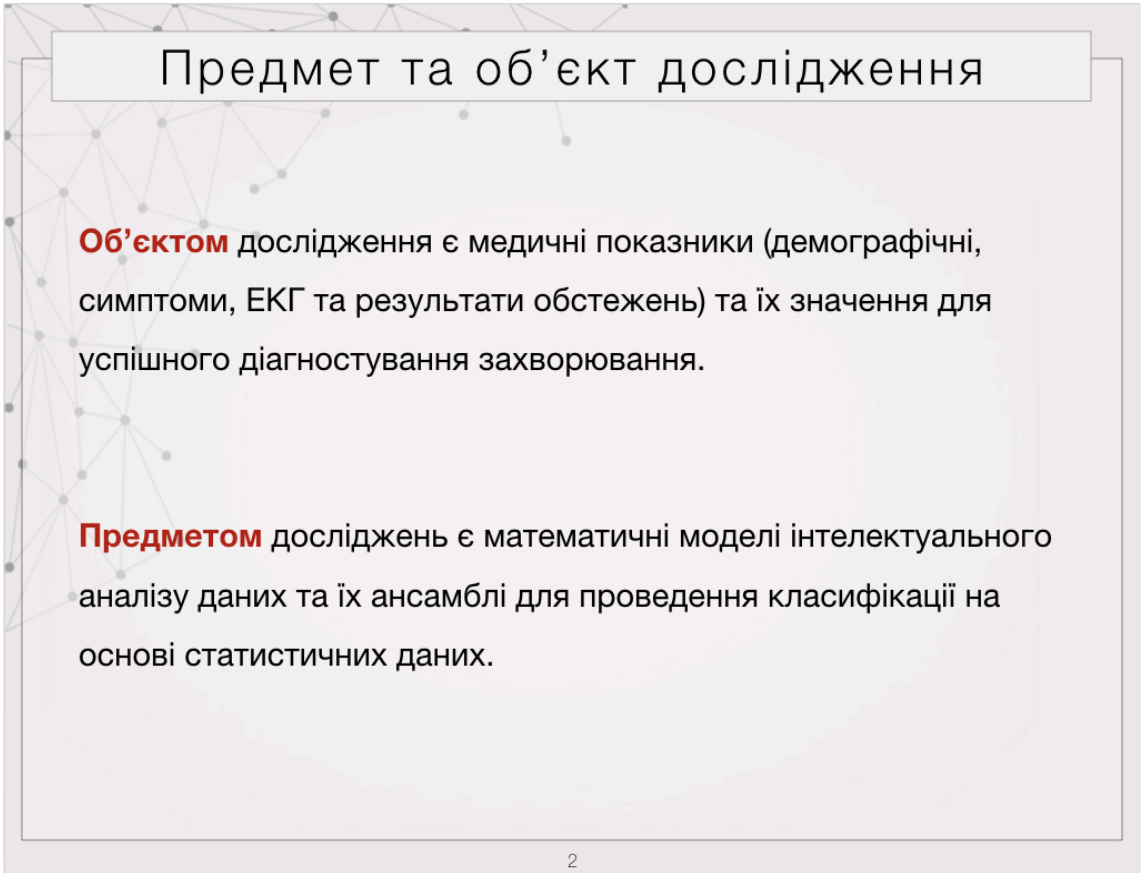
%86%D0%B8%D0%B9%2C_%D0%9A.%D0%92.%D0%92%D0%BE%D1%80%D0%BE%D0%BD%D1%86%D0%BE%D0%B2%29

25. Воронцов К.В. Лекции по логическим алгоритмам классификации [Електронний ресурс] / Воронцов К.В. — Режим доступа: http://www.ccas.ru/voron/download/LogicAlgs.pdf
26. Курс "Машинное обучение" на ФКН ВШЭ [Електронний ресурс] — Режим доступа: https://github.com/esokolov/ml-course-hse
27. Rodrıguez J.J. Rotation Forest: A New Classifier Ensemble Method [Електронний ресурс] / Rodrıguez J.J., Kuncheva L.I., Alonso C.J. // IEEE Transactions on Pattern Analysis and Machine Intelligence. ВШЭ — Режим доступа: https://ieeexplore.ieee.org/abstract/document/1677518/
28. Дьяконов А. Введение в анализ данных и машинное обучение [Електронний ресурс] / Дьяконов А. — Режим доступа: https://alexanderdyakonov.files.wordpress.com/2017/06/book_boosting_pdf.pdf
29. Chen T. XGBoost: A Scalable Tree Boosting System/ Chen T., Guestrin C. [Електронний ресурс] — Режим доступа: https://arxiv.org/abs/1603.02754
30. Tumer K.A Error Correlation and Error Reduction in Ensemble Classiers [Електронний ресурс] / Tumer K.A., — Режим доступа: https://www.tandfonline.com/doi/abs/10.1080/095400996116839
31. Ali S. Can–Evo–Ens: Classifier stacking based evolutionary ensemble system for prediction of human breast cancer using amino acid sequences [Електронний ресурс] / / Ali S., Majid A. — Режим доступа: https://www.sciencedirect.com/science/article/pii/S1532046415000064?via%3Dihub
32. Hand D.J. Measuring classifier performance: a coherent alternative to the area under the ROC curve [Електронний ресурс] / Hand D.J. – Springer Science+Business Media, LLC 2009 – Режим доступа: https://link.springer.com/content/pdf/10.1007%2Fs10994-009-5119-5.pdf

33. Hern'andez-Orallo J. Brier Curves: A New Cost-Based Visualisation of Classifier Performance [Електронний ресурс] / Hern'andez-Orallo J., Flash P., Ferri C.: праці конф., 2011: Proceedings of the 28th International Conference on Machine Learning, – Bellevue, WA, USA — Режим доступа: https://pdfs.semanticscholar.org/2bc1/743405cf276125e798fbd96290b518c51d56.pdf
34. Sokolova M. Beyond Accuracy, F-score and ROC: a Family of Discriminant Measures for Performance Evaluation [Електронний ресурс] / Sokolova M., Japkowicz N., Szpakowicz S. – American Association for Artificial Intelligence — Режим доступа: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.549.2795&rep=rep1&type=pdf
35. Youden, W. Index for rating diagnostic tests. [Електронний ресурс] / Youden, W. — Режим доступа: https://onlinelibrary.wiley.com/doi/abs/10.1002/1097-0142%281950%293%3A1%3C32%3A%3AAID-CNCR2820030106%3E3.0.CO%3B2-3
36. Biggerstaff, B. Comparing diagnostic tests: a simple graphic using likelihood ratios./ Biggerstaff, B. // Statistics in Medicine. – 2000. – No19(5). – pp. 649–663.
37. Blakeley, D. Noninvasive carotid artery testing / Blakeley, D., Oddone, E. – // Ann Intern Med. – 122. –pp.360–367.
38. Machine Learning Repository [Електронний ресурс] — Режим доступа: http://archive.ics.uci.edu/ml/datasets/extention+of+Z-Alizadeh+sani+dataset
39. Effective Wrapper-Filter Hybridization Through GRASP Schemata / [Esseghir M., Liu H., Motoda H., et al] – The Fourth Workshop and Conference Proceedings on Feature Selection in Data Mining. – 2010. – pp. 45-54

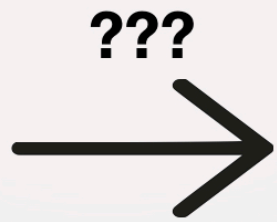
40. Kohavi R. Wrappers for feature subset selection, Artificial Intelligence/ Kohavi R. John G., – vol.97(1–2): pp.273-324
41. On the use of classification reliability for improving performance of the one-per-class decomposition method / [Iannello G., Percannella G., Sansone C., Soda P.] – Data and Knowledge Engineering, – vol. 68(12). – pp. 1398-1410
42. Polat K.A novel hybrid intelligent method based on C4.5 decision tree classifier and one-against-all approach for multi-class classification problems/ Polat K., Gunes S., – An International Journal of Expert Systems with Applications. – 2009. – vol. 36(2). – pp.587-1592
43. Pawlak Z. Rough Sets / Pawlak Z. // International Journal of Computer and Information Science. – 1982. – No11. – pp. 341-356
44. Jolliffe I.T. Principal component analysis: a review and recent developments / Jolliffe I.T., Cadima J. [Електронний ресурс] — Режим доступа: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4792409/
45. Bashir S. IntelliHealth: A medical decision support application using a novel weighted multi-layer classifier ensemble framework / Bashir S., Qamar U., Khan F.H. // Journal of Biomedical Informatics. – 2016. – vol.59. – pp.185-200
46. Ali S. Can–Evo–Ens: Classifier stacking based evolutionary ensemble system for prediction of human breast cancer using amino acid sequences [Електронний ресурс] / / Ali S., Majid A. — Режим доступа: https://www.sciencedirect.com/science/article/pii/S1532046415000064?via%3Dihub
47. U.S. Chronic Disease Indicators (CDI) [Електронний ресурс] — Режим доступа: https://catalog.data.gov/dataset/u-s-chronic-disease-indicators-cdi

# ДОДАТОК А ІЛЮСТРАТИВНИЙ МАТЕРІАЛ

# Актуальність роботи

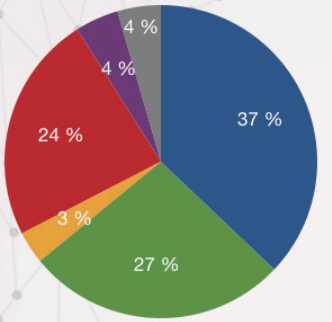
У вузькому розумінні діагностикою називають безпосередньо процес розпізнавання хвороби та оцінки індивідуальних біологічних і соціальних особливостей суб'єкта, який включає цілеспрямоване медичне обстеження, тлумачення отриманих результатів та їх узагальнення у вигляді встановленого діагнозу.




3

# Актуальність роботи

**Типи медичних помилок**




- Діагностичні
- Хірургічні
- Анестезія
- Лікування
- Медикаменти
- Інше


Головним фактором, який спричиняє понад 80% когнітивних помилок, є помилковий синтез інформації.

За даними Світової Організації Охорони Здоров'я існує понад 12 000 хвороб.
Не варто забувати, що лікарі – також люди. Вони так само зазнають впливу втоми, стресу, психологічних факторів тощо.

World Health Organization

4

# Постановка задачі

**На меті цієї науково-дослідної роботи стоїть розробка і дослідження методів інтелектуального аналізу даних для діагностування Ішемічної хвороби серця.**

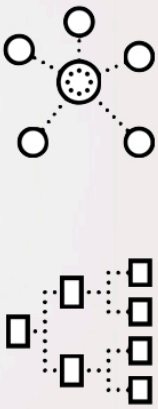
Для досягнення цієї мети потрібно розв'язати наступні задачі:

- ✓ Огляд літератури, опис найбільш вживаних моделей класифікації та використання їх ансамблів;
- ✓ Вивчення метрик, що застосовуються для оцінки якості результатів роботи методів бінарної класифікації, визначити переваги та недоліки кожної.
- ✓ Порівняльний аналіз результатів застосування вищезгаданих підходів та вибір найбільш оптимального для вирішення практичної задачі.
- ✓ Запропонувати модифікацію методу, яка дозволить підвищити значення критеріїв оцінки якості на реальних даних.

5

# Використані базові моделі класифікації

- Логістична регресія
- Naive Bayes Classifier
- K найближчих сусідів
- Дерева рішень
- Метод опорних векторів (SVM)
- Метод стохастичного градієнтного спуску
- Linear Discriminant Analysis



6

# Методи агрегування класифікаторів

**Беггінг (Bagging or Bootstrap Aggregating):**

- паралельне навчання класифікаторів;
- отримання кінцевого результату шляхом голосування;

Один з класичних алгоритмів класифікації, в основу якого покладено бегінг – це *Random Forest*.

**Бустинг (Boosting):**

- послідовне навчання класифікаторів;
- кожний наступний класифікатор має на меті компенсацію недоліків попереднього;

Перелік алгоритмів, в основі яких знаходиться принцип бустингу, дуже великий: *LPBoost, BrownBoost, LogitBoost, AdaBoost, CatBoost, XGBoost, LightGBM* і тд

7

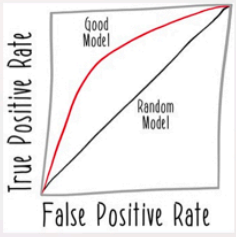
# Опис використаних метрик

- $Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$
- $Precision = \frac{TP}{TP + FP}$
- $Recall = \frac{TP}{TP + FN}$
- $Specificity = \frac{TN}{TN + FP}$
- $F = \frac{2 * precision * recall}{precision + recall}$

Confusion matrix

| | | True values | |
|---|---|---|---|
| | | y= 1 | y=0 |
| Predicted values | a(x) = 1 | True Positive (TP) | False Positive (FP) |
| | a(x) = 0 | False Negative (FN) | True Negative (TN) |

- *Площа під ROC кривою*, яка є результатом зображення на графіку взаємозв'язку True Positive rate (TPR) та False Positive rate (FPR)




8

# Опис набору даних

**Діагностування ішемічної хвороби серця.**

Розширений датасет Z-Alizadeh Sani містить дані о 303 пацієнтах (об'єкти), кожен з яких описується 56 атрибутами.

Атрибути розділені на 4 категорії: демографічні, симптоми та результати огляду, ЕКГ, а також атрибути лабораторних досліджень і знімків.

| | Age | Weight | Length | Sex | BMI | DM | HTN | Current Smoker | EX-Smoker | FH | ... | Lymph | Neut | PLT | EF-TTE | Region RWMA | VHD | LAD | LCX | RCA | Cath |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 53 | 90 | 175 | Male | 29.387755 | 0 | 1 | 1 | 0 | 0 | ... | 39 | 52 | 261 | 50 | 0 | N | Stenotic | Normal | Stenotic | CAD |
| 1 | 67 | 70 | 157 | Fmale | 28.398718 | 0 | 1 | 0 | 0 | 0 | ... | 38 | 55 | 165 | 40 | 4 | N | Stenotic | Stenotic | Normal | CAD |
| 2 | 54 | 54 | 164 | Male | 20.077335 | 0 | 0 | 1 | 0 | 0 | ... | 38 | 60 | 230 | 40 | 2 | mild | Stenotic | Normal | Normal | CAD |
| 3 | 66 | 67 | 158 | Fmale | 26.838648 | 0 | 1 | 0 | 0 | 0 | ... | 18 | 72 | 742 | 55 | 0 | Severe | Normal | Normal | Normal | Normal |
| 4 | 50 | 87 | 153 | Fmale | 37.165193 | 0 | 1 | 0 | 0 | 0 | ... | 55 | 39 | 274 | 50 | 0 | Severe | Normal | Normal | Normal | Normal |

80% даних були використані для навчання алгоритмів, 20% - для тестування.

9

# Попередня обробка даних

1. **Обробка атрибутів, що мають текстові значення**

2. **Трансформація ознак:**
   — стандартизація;
   — мінімаксне перетворення;

3. **Відбір ознак (Feature Selection):**

| feature selection approach | number of features |
|---|---|
| decision trees | 13 |
| rfecv | 33 |
| LinearSVC | 25 |
| PCA | 20 |

Для f-classif та mutual_info користувач самостійно вказує кількість ознак, яку він бажає залишити.

```
{'Age': 38.26151195021897,
 'Airway disease': 1.03658594683569,
 'Atypical': 55.44708100887877,
 'BBB': 1.4511417823550843,
 'BMI': 1.6323746047588417,
 'BP': 14.422641451432781,
 'BUN': 1.7335184268635737,
 'CHF': 0.3978473957332307,
 'CR': 1.3815790483087549,
 'CRF': 1.6196391021565841,
 'CVA': 0.0343554106967321,
 'Current Smoker': 0.25237098079596465,
 'DLP': 0.0007180576540313314,
 'DM': 17.628890945814415,
 'Diastolic Murmur': 6.808775593509114,
 'Dyspnea': 7.575600993375226,
 'EF-TTE': 9.194371843509135,
 'ESR': 11.737463515794577,
 'EX-Smoker': 0.7119847684301657,
 'Edema': 0.36832270795363464,
```

10

# Результати роботи базових класифікаторів

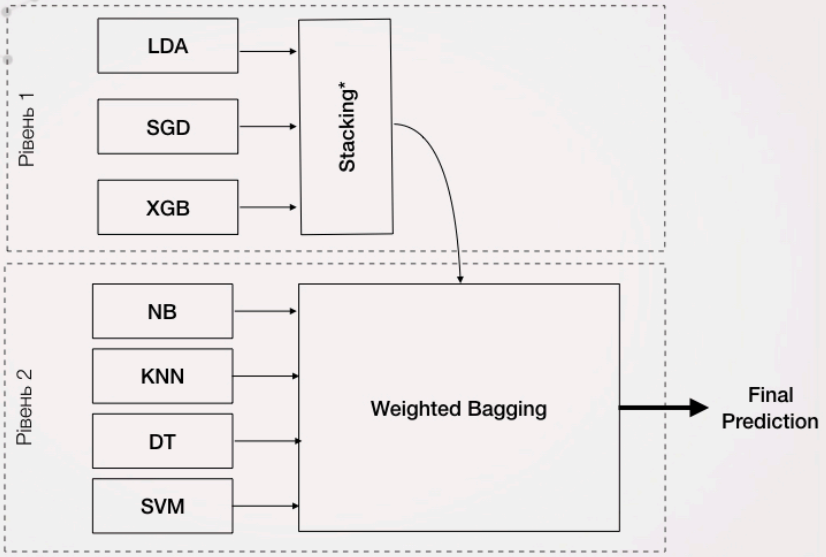
Для кожного із класифікаторів були визначені оптимальні значення параметрів шляхом застосування перехресної перевірки. Кожна із моделей була протестована як на повному наборі ознак, так і на його піднаборах, що були визначені за допомогою згаданих раніше методів feature selection.

| Метод | accuracy | precision | recall | specificity | f1 score | roc auc score |
|---|---|---|---|---|---|---|
| Логістична регресія | 0.87 | 0.91 | 0.91 | 0.78 | 0.91 | 0.84 |
| Наївний байєсовський класифікатор | 0.85 | 0.9 | 0.88 | 0.78 | 0.89 | 0.83 |
| K найближчих сусідів | 0.89 | 0.89 | 0.95 | 0.72 | 0.92 | 0.84 |
| Дерева рішень | 0.87 | 0.91 | 0.91 | 0.78 | 0.91 | 0.86 |
| Метод опорних векторів | 0.85 | 0.9 | 0.88 | 0.78 | 0.89 | 0.83 |
| Метод стохастичного град. спуску | 0.87 | 0.91 | 0.91 | 0.78 | 0.91 | 0.84 |
| Лінійно-дискримінатний аналіз | 0.89 | 0.91 | 0.93 | 0.78 | 0.92 | 0.85 |

З таблиці можна побачити, що найкращу оцінку повноти дає метод k найближчих сусідів, в той час як найвищу точність мають логістична регресія, метод опорних векторів та лінійно-дискримінантний аналіз. F-міра має найвищий показник для результатів лінійно-дискримінантного аналізу, а площа під roc-кривою найбільша для результатів класифікації з використанням дерев рішень.

11

# Запропонована дворівнева модель класифікаціїї




*** Stacking** - такий спосіб агрегування класифікаторів, при якому відповіді кожної моделі є вхідними даними для метаалгоритму, що на виході дає остаточну відповідь

12

# Результати роботи запропонованої моделі

Результати роботи першого рівня моделі

| Метод | accuracy | precision | recall | specificity | f1 score | roc auc score |
|---|---|---|---|---|---|---|
| LDA | 0.89 | 0.91 | 0.93 | 0.78 | 0.92 | 0.85 |
| SGD | 0.87 | 0.91 | 0.91 | 0.78 | 0.91 | 0.84 |
| XGB | 0.84 | 0.85 | 0.93 | 0.61 | 0.89 | 0.77 |
| **Stacking (LR)** | **0.9** | **0.93** | **0.93** | **0.83** | **0.93** | **0.88** |

Результати роботи другого рівня моделі

| Метод | accuracy | precision | recall | specificity | f1 score | roc auc score |
|---|---|---|---|---|---|---|
| NB | 0.85 | 0.9 | 0.88 | 0.78 | 0.89 | 0.83 |
| KNN | 0.89 | 0.89 | 0.95 | 0.72 | 0.92 | 0.84 |
| DT | 0.87 | 0.91 | 0.91 | 0.78 | 0.91 | 0.84 |
| SVM | 0.85 | 0.9 | 0.88 | 0.78 | 0.89 | 0.83 |
| Stacking (LR) | 0.9 | 0.93 | 0.93 | 0.83 | 0.93 | 0.88 |
| **Bagging** | **0.92** | **0.93** | **0.95** | **0.83** | **0.94** | **0.89** |

13

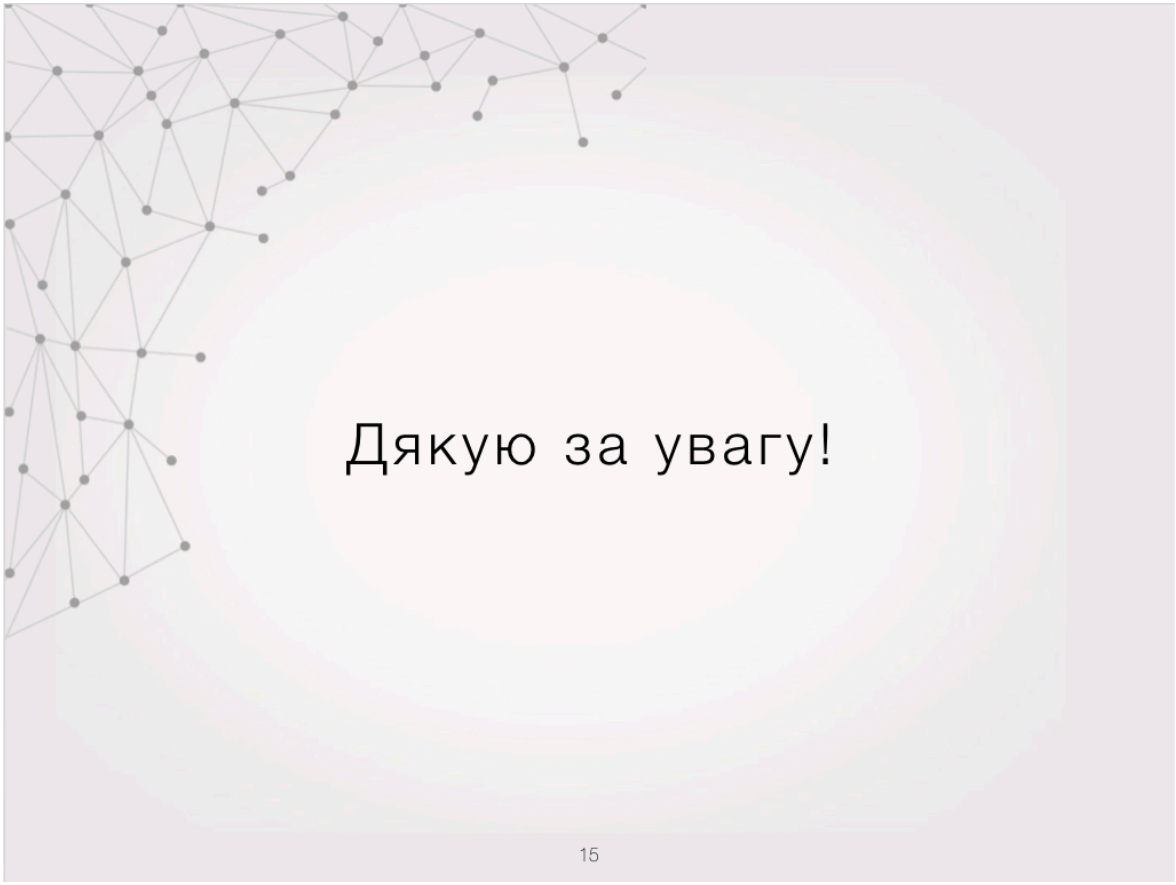
# Висновки по роботі та подальші рекомендації

В роботі було розглянуто питання актуальності теми, що досліджується. Для досягнення поставленої мети було вирішено такі задачі:

- ✓ Було досліджено теоретичну база задач, пов'язаних з проблемою бінарної класифікації, що включало в себе огляд і аналіз літератури.
- ✓ Був проведений огляд та опис найбільш поширених методів підготовки даних для подальшої роботи з ними: попередньої обробки даних, відбору ознак та трансформації.
- ✓ Був проведений огляд основних моделей класифікації та їх ансамблів, наведені недоліки та переваги роботи кожної; продемонстрована ефективність роботи кожної на реальних даних.
- ✓ Було запропоновано покращену модель дворівневої класифікації та наведені результати застосування до вирішення практичної задачі та проведений їх порівняльний аналіз.

В майбутньому рекомендується продовжити дослідження в даному напрямі з метою удосконалення моделі класифікації та покращення її узагальнюючих властивостей, а саме: розширити множину методів класифікації, додати використання нейромереж.

14


Дякую за увагу!
15

# ДОДАТОК Б ЛІСТИНГ ПРОГРАМИ

```
import pandas as pd
import numpy as np

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
roc_auc_score, precision_recall_curve,log_loss
from sklearn.model_selection import train_test_split, KFold
from sklearn.preprocessing import MinMaxScaler,Normalizer, scale
from sklearn.svm import LinearSVC
from sklearn.feature_selection import SelectFromModel, RFECV, SelectKBest, mutual_info_classif
from sklearn.decomposition import PCA
from sklearn.ensemble import ExtraTreesClassifier,RandomForestClassifier
from xgboost import XGBClassifier
from vecstack import stacking
from sklearn.naive_bayes import GaussianNB,BernoulliNB
from sklearn.tree import DecisionTreeClassifier

df = pd.read_excel('CAD.xlsx')
df.head()

df.columns


Y_LAD = df['LAD'].replace({'Stenotic':1, 'Normal':0})
Y_LCX = df['LCX'].replace({'Stenotic':1, 'Normal':0})
Y_RCA = df['RCA'].replace({'Stenotic':1, 'Normal':0})

df = df.drop(labels = ['LAD','LCX','RCA'],axis = 1)

df['Sex'] = df['Sex'].replace({"Male":1,"Fmale":0})
df['Cath'] = df['Cath'].replace({"CAD":1,"Normal":0})
df['VHD'] = df['VHD'].replace({'mild':1,'Severe':2, 'Moderate':3, 'N':0})
df['BBB'] = df['BBB'].replace({'LBBB':1, 'RBBB':2})

pd.value_counts(df['Cath'])

#replacing all Y/N values in columns

def replace_Y_N(row):
    for col in df.columns:
        if row[col] == "Y":
            row[col] = 1
        if row[col] == "N":
            row[col] = 0
    return row

df = df.apply(replace_Y_N,axis = 1)
for col in df.columns:
```

```
    print (col)
    print (df[col].unique())

df['K'].hist()

x = df.iloc[:,:-1]
y = df.iloc[:,-1]

import scipy
scipy.sparse.issparse(x.as_matrix())

def split_data(x,y):
    X_train,X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.2, random_state=42)
    Y_test = Y_test.reset_index()['Cath']
    X_test = X_test.reset_index().iloc[:,1:]
    return X_train,X_test,Y_train, Y_test


X_train,X_test,Y_train, Y_test = split_data(x,y)

pd.value_counts(Y_train)
pd.value_counts(Y_test)

from sklearn.utils import class_weight
class_weight = class_weight.compute_class_weight('balanced', np.unique(Y_train), Y_train)
class_weight

weights = {1:0.69942197,0:1.75362319}

# # Feature Selection

#2. features selected by Decision trees
clf = DecisionTreeClassifier(random_state=241,splitter = 'random',class_weight=weights,
max_depth=20)
clf.fit(X_train,Y_train)
model = SelectFromModel(clf, prefit=True)
X_DT = model.transform(x)
X_DT_train = model.transform(X_train)
X_DT_test = model.transform(X_test)

#3. RFECV
clf = LogisticRegression(random_state=241,class_weight=weights)
selector = RFECV(estimator=clf, cv=5,scoring='neg_mean_squared_error')
selector.fit(X_train, Y_train)
print('Optimal number of features: %d' % selector.n_features_)

dict(zip(list(df.columns),selector.support_))

x_rfecv = selector.transform(x)
X_train_rfecv = selector.transform(X_train)
X_test_rfecv = selector.transform(X_test)

# select K-best
```

```
# f_classif: ANOVA F-value between label/feature for classification tasks.
test = SelectKBest( k=13)
fitted = test.fit(X_train, Y_train)

X_train_kbest = fitted.transform(X_train)
X_test_kbest = fitted.transform(X_test)
# summarize selected features
dict(zip(list(df.columns),fitted.scores_))

#by information gain
from sklearn.feature_selection import mutual_info_classif
#dict(zip(list(df.columns),mutual_info_classif(X_train,Y_train)))

test = SelectKBest(mutual_info_classif, k=13)
fitted = test.fit(X_train, Y_train)

# summarize scores
X_train_kbest_MI = fitted.transform(X_train)
X_test_kbest_MI = fitted.transform(X_test)

dict(zip(list(df.columns),fitted.scores_))

# 4. Selected by random forest

#5. selecting feautures by svm
lsvc = LinearSVC(C=0.1, penalty="l1", dual=False).fit(X_train, Y_train)
model = SelectFromModel(lsvc, prefit=True)
X_svm = model.transform(x)
X_train_svm = model.transform(X_train)
X_test_svm = model.transform (X_test)

#PCA
pca = PCA(n_components=20)
pca.fit(X_train)
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)
x_pca = pca.transform(x)

pca.explained_variance_


def feature_selection(how,train = X_train, test = X_test,returnx = False):
    if how == 'decision_trees':
        clf = DecisionTreeClassifier(random_state=241,splitter = 'random', class_weight=weights,
                                     max_depth=20)
        clf.fit(train,Y_train)
        model = SelectFromModel(clf, prefit=True)
        train1 = model.transform(train)
        test1 = model.transform(test)
        if returnx is True:
            x1 = model.transform(x)
    if how == 'rfecv':
        clf = LogisticRegression(random_state=241,class_weight=weights)
```

```
        selector = RFECV(estimator=clf, cv=5,scoring='neg_mean_squared_error')
        selector.fit(train, Y_train)
        train1 = selector.transform(train)
        test1 = selector.transform(test)
        if returnx is True:
            x1 = selector.transform(x)
    if how == 'f_classif':
        selector = SelectKBest(k=30)
        fitted = selector.fit(train, Y_train)
        train1 = fitted.transform(train)
        test1 = fitted.transform(test)
        if returnx is True:
            x1 = fitted.transform(x)
    if how == 'mutual_info':
        selector = SelectKBest(mutual_info_classif, k=30)
        fitted = selector.fit(train, Y_train)
        train1 = fitted.transform(train)
        test1 = fitted.transform(test)
        if returnx is True:
            x1 = fitted.transform(x)
    if how == 'svm':
        lsvc = LinearSVC(C=0.1, penalty="l1", dual=False).fit(train, Y_train)
        model = SelectFromModel(lsvc, prefit=True)
        train1 = model.transform(train)
        test1 = model.transform (test)
        if returnx is True:
            x1 = model.transform(x)
    if how == 'pca':
        pca = PCA(n_components=30)
        pca.fit(train)
        train1 = pca.transform(train)
        test1 = pca.transform(test)
        if returnx is True:
            x1 = pca.transform(x)

    if returnx is True:
            return train1,test1, x1
    else:
        return train1,test1

fs_list = ['decision_trees','rfecv','f_classif','mutual_info','svm','pca']


# # ----------------------------------------

# rescaling data
def rescaling (train ,test, how):

    if how == 'minmax':
        scaler = MinMaxScaler(feature_range=(0, 1))
        train1 = scaler.fit_transform(train)
        test1 = scaler.transform(test)
    if how == 'stand':
```

```
        train1 = scale(train)
        test1 = scale(test)

    return train1,test1

sc = ['minmax','stand']


def estimate_model(res):
    TP = FP = TN = FN = 0

    for i in range(len(Y_test)):
        if Y_test[i] == 1:
            if res[i] == 1:
                TP+=1
            else:
                FN+=1
        if Y_test[i] == 0:
            if res[i] == 1:
                FP+=1
            else:
                TN+=1

    print ('TP=',TP)
    print ('FP=',FP)
    print ('FN=',FN)
    print ('TN=',TN)
    print ('############################')
    print('accuracy', '%.2f'% (accuracy_score(Y_test,res)))
    print('precision', '%.2f'% (precision_score(Y_test,res)))
    print ('recall/sensitivity','%.2f'% (recall_score(Y_test,res)))
    print ('specificity', '%.2f'%(TN/ (TN+FP)))
    print ('f1 score', '%.2f'% (f1_score(Y_test,res)))
    print ('roc auc score', '%.2f'% (roc_auc_score(Y_test,res)))


### Logistic regression

#clf = LogisticRegression(random_state=241,class_weight={1:0.4,0:0.6})
#clf = LogisticRegression(random_state=241)
clf = LogisticRegression(random_state=241,class_weight=weights)
clf.fit(X_train,Y_train)
res = clf.predict(X_test)
estimate_model(res)

res1 = clf.predict_proba(X_test)
log_loss(Y_test,res1)

#### rescaled

clf =  LogisticRegression(random_state=241,class_weight=weights)

for s in sc:
```

```
    print ('--------',s,'--------')
    X_train1,X_test1 = rescaling (X_train, X_test, s)
    clf.fit(X_train1,Y_train)
    res = clf.predict(X_test1)
    estimate_model(res)
    print ('\n\n')


#### after feauture selection

clf = LogisticRegression(random_state=241,class_weight=weights)

for i in fs_list:
    print ('--------',i,'--------')
    train,test = feature_selection(i)
    clf.fit(train,Y_train)
    res = clf.predict(test)
    estimate_model(res)
    print ('\n\n')

#rescaling + feature selection
clf =  LogisticRegression(random_state=241,class_weight=weights)

for i in fs_list:
    for s in sc:
        print ('--------',i,'--------')
        print ('--------',s,'--------')
        X_train1,X_test1 = rescaling (X_train, X_test, s)
        train,test = feature_selection(i,X_train1, X_test1)

        clf.fit(train,Y_train)
        res = clf.predict(test)
        estimate_model(res)
        print ('\n\n')


# ### Naive Bayes Classifier

# #### Gaussian

clf = GaussianNB()
clf.fit(X_train,Y_train)

res = clf.predict(X_test)
estimate_model(res)


# #### Bernoulli

clf = BernoulliNB()
clf.fit(X_train,Y_train)

res= clf.predict(X_test)
```

```
estimate_model(res)

clf = BernoulliNB()
for s in sc:
    print ('--------',s,'--------')
    X_train1,X_test1 = rescaling (X_train, X_test, s)
    clf.fit(X_train1,Y_train)
    res = clf.predict(X_test1)
    estimate_model(res)
    print ('\n\n')


# ### after feauture selection

clf = BernoulliNB()

for i in fs_list:
    print (i)
    train,test = feature_selection(i)
    clf.fit(train,Y_train)
    res = clf.predict(test)
    estimate_model(res)
    print ('\n\n')


#rescaling + feature selection
clf = BernoulliNB()

for i in fs_list:
    for s in sc:
        print (i)
        print (s)
        X_train1,X_test1 = rescaling (X_train, X_test, s)
        train,test = feature_selection(i,X_train1, X_test1)

        clf.fit(train,Y_train)
        res = clf.predict(test)
        estimate_model(res)
        print ('\n\n')


# ### K neighbors classifier

from sklearn.cross_validation import KFold, cross_val_score
from sklearn.neighbors import KNeighborsClassifier


kf = KFold(303, n_folds = 4, shuffle = True, random_state=42)
errors_acc,errors_roc=[],[]
for k in range(1,15):
    model = KNeighborsClassifier(n_neighbors=k,weights='distance')
    #calculating errors
    a = cross_val_score(model,x,y,scoring = 'recall', cv = kf)
```

```
    errors_acc.append(a.mean())

neib = pd.Series(errors_acc, index =range(1,15))

for i in range(1,15):
    if neib[i] == max(neib):
        print (i, neib[i])


clf = KNeighborsClassifier(n_neighbors=14)
clf.fit(x,y)
res = clf.predict(x)
print('training accuracy')
recall_score(res,y)

clf = KNeighborsClassifier(n_neighbors=14)
clf.fit(X_train,Y_train)
res = clf.predict(X_test)

estimate_model(res)


# #### ----------RESCALED----------
clf = KNeighborsClassifier(n_neighbors=9)

for s in sc:
    print ('--------',s,'--------')
    X_train1,X_test1 = rescaling (X_train, X_test, s)
    clf.fit(X_train1,Y_train)
    res = clf.predict(X_test1)
    estimate_model(res)
    print ('\n\n')


# ### after feauture selection

kf= KFold(303,n_folds = 4, shuffle = True, random_state=42)

for i in fs_list:
    errors=[]
    print ('--------',i,'--------')
    train,test,x1 = feature_selection(i,returnx=True)
    for k in range(1,15):
        model = KNeighborsClassifier(n_neighbors=k)
        a = cross_val_score(model,x1,y,scoring = 'recall', cv = kf)
        errors.append(a.mean())

    neib = pd.Series(errors, index=range(1,15))

    for i in range(1,15):
        if neib[i] == max(neib):
            print (i, neib[i])
```

```
# #### Decision Trees

clf = DecisionTreeClassifier(random_state=241,splitter = 'random',class_weight=weights,
max_depth=20)
clf.fit(X_train,Y_train)
res = clf.predict(X_test)

estimate_model(res)


clf.feature_importances_


for i in range(len(clf.feature_importances_)):
    if clf.feature_importances_[i]>0:
        print (df.columns[i],clf.feature_importances_[i] )


# ### after feauture selection


clf = DecisionTreeClassifier(random_state=241,splitter = 'random')

for i in fs_list:
    print ('--------',i,'--------')
    train,test = feature_selection(i)
    clf.fit(train,Y_train)
    res = clf.predict(test)
    estimate_model(res)
    print ('\n\n')

clf = DecisionTreeClassifier(random_state=241,splitter = 'random')

for i in fs_list:
    for s in sc:
        print ('--------',i,'--------')
        print ('--------',s,'--------')
        X_train1,X_test1 = rescaling (X_train, X_test, s)
        train,test = feature_selection(i,X_train1, X_test1)

        clf.fit(train,Y_train)
        res = clf.predict(test)
        estimate_model(res)
        print ('\n\n')


# ### SVM

from sklearn.svm import SVC
from sklearn.grid_search import GridSearchCV
```

```
parameters = {'kernel':('linear', 'rbf'), 'C':[0.001, 0.01, 0.1, 1, 10]}
cv=KFold(y.size, n_folds = 5, shuffle = True, random_state=241)
svc = SVC(class_weight= weights)
clf = GridSearchCV(svc, parameters,scoring='accuracy',cv=cv)

clf.fit(x,y)

clf.grid_scores_

clf = SVC(C = 0.1, kernel='linear',class_weight=weights)
clf.fit(X_train,Y_train)
res = clf.predict(X_test)
estimate_model(res)




# #### ----------RESCALED----------
clf = SVC(kernel='linear',class_weight=weights)

for s in sc:
    print ('--------',s,'--------')
    X_train1,X_test1 = rescaling (X_train, X_test, s)
    clf.fit(X_train1,Y_train)
    res = clf.predict(X_test1)
    estimate_model(res)
    print ('\n\n')


# ### after feauture selection

clf = SVC(C=0.1, kernel='linear',class_weight=weights)
for i in fs_list:
    print ('--------',i,'--------')
    train,test = feature_selection(i)
    clf.fit(train,Y_train)
    res = clf.predict(test)
    estimate_model(res)
    print ('\n\n')


clf = SVC(C=0.1, kernel='linear',class_weight=weights)

for i in fs_list:
    for s in sc:
        print ('--------',i,'--------')
        print ('--------',s,'--------')
        X_train1,X_test1 = rescaling (X_train, X_test, s)
        train,test = feature_selection(i,X_train1, X_test1)

        clf.fit(train,Y_train)
        res = clf.predict(test)
        estimate_model(res)
```

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


# ### Stochastic Gradient Descent

from sklearn.linear_model import SGDClassifier

clf = SGDClassifier(max_iter=10000,loss='squared_hinge',class_weight=weights)
clf.fit(X_train,Y_train)

res = clf.predict(X_test)
estimate_model(res)

for i in range(len(df.columns)-1):
    print (df.columns[i],clf.coef_[0][i])


# #### ----------RESCALED----------

# For best results using the default learning rate schedule, the data should have zero mean and unit variance.

clf = SGDClassifier(max_iter=10000,loss='squared_hinge',class_weight=weights)
X_train1,X_test1 = rescaling(X_train,X_test,'stand')
clf.fit(X_train1,Y_train)

res = clf.predict(X_test1)
estimate_model(res)


# ### after feauture selection

clf = SGDClassifier(max_iter=10000,loss='modified_huber',class_weight={1:0.4,0:0.6})

for i in fs_list:
    print ('--------',i,'--------')
    print ('-------- standartarized --------')
    X_train1,X_test1 = rescaling(X_train,X_test,'stand')
    train,test = feature_selection(i,X_train1,X_test1)
    clf.fit(train,Y_train)
    res = clf.predict(test)
    estimate_model(res)
    print ('\n\n')


# ## LDA

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
clf = LinearDiscriminantAnalysis()
clf.fit(X_train, Y_train)


clf = LinearDiscriminantAnalysis()
```

```
for i in fs_list:
    for s in sc:
        print ('--------',i,'--------')
        print ('--------',s,'--------')
        X_train1,X_test1 = rescaling (X_train, X_test, s)
        train,test = feature_selection(i,X_train1, X_test1)

        clf.fit(train,Y_train)
        res = clf.predict(test)
        estimate_model(res)
        print ('\n\n')


# ## Multi-layer Perceptron classifier.

from sklearn.neural_network import MLPClassifier

X_train1,X_test1 = rescaling (X_train, X_test, 'stand')
train,test = feature_selection('decision_trees',X_train1, X_test1)

clf = MLPClassifier(activation='tanh',learning_rate='adaptive',alpha =0.01, max_iter=200)

clf.fit(train,Y_train)
res = clf.predict(test)

estimate_model(res)

clf = MLPClassifier(activation='tanh',learning_rate='adaptive',alpha =0.01, max_iter=200)

for i in fs_list:
    for s in sc:
        print ('--------',i,'--------')
        print ('--------',s,'--------')
        X_train1,X_test1 = rescaling (X_train, X_test, s)
        train,test = feature_selection(i,X_train1, X_test1)

        clf.fit(train,Y_train)
        res = clf.predict(test)
        estimate_model(res)
        print ('\n\n')


# # ----------Ensembles-------------

# ### Random Forest
from sklearn.model_selection import cross_val_score
from sklearn.datasets import make_blobs
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
```

```
clf = RandomForestClassifier(n_estimators=20, max_depth=10,random_state=41)
clf.fit(X_train,Y_train)
res = clf.predict(X_test)
estimate_model(res)

clf = RandomForestClassifier(n_estimators=20, max_depth=10,random_state=41)

for i in fs_list:
    for s in sc:
        print ('--------',i,'--------')
        print ('--------',s,'--------')
        X_train1,X_test1 = rescaling (X_train, X_test, s)
        train,test = feature_selection(i,X_train1, X_test1)

        clf.fit(train,Y_train)
        res = clf.predict(test)
        estimate_model(res)
        print ('\n\n')



# ### XGBoost
import xgboost as xgb

dtrain = xgb.DMatrix(X_train, label=Y_train)
dtest = xgb.DMatrix(X_test,label = Y_test)

param = {'max_depth': 2, 'eta': 1, 'silent': 1, 'objective': 'binary:logistic'}
param['nthread'] = 4
param['eval_metric'] = 'auc'

evallist = [(dtest, 'eval'), (dtrain, 'train')]

num_round = 10
bst = xgb.train(param, dtrain, num_round, evallist)

ypred = bst.predict(dtest)
log_loss(Y_test,ypred)

res = []
for i in range(len(ypred)):
    if ypred[i]<0.5:
        res.append(0)
    else:
        res.append(1)

estimate_model(res)

# ## ------------------------------------

# # Bagging

LR = LogisticRegression(random_state=241,class_weight=weights)
```

```
NB = BernoulliNB()
KNN = KNeighborsClassifier(n_neighbors=9)
SVM = SVC(C=0.1, kernel='linear',class_weight=weights, probability=True)
DT = DecisionTreeClassifier(random_state=241, splitter='random')
RF= RandomForestClassifier(n_estimators=20, max_depth=10,random_state=41)
SGD = SGDClassifier(max_iter=10000,loss='modified_huber',class_weight={1:0.4,0:0.6})
LDA = LinearDiscriminantAnalysis()

models = [('nb', NB),('knn', KNN),('LogReg', LR), ('DT', DT),('RF',RF),('svm', SVM)]

agg_weights = [0.89,0.92,0.91,0.85] #f1_scores minmax
agg_weights1 = [0.91,0.93,0.89,0.93] #f1_scores stand




from sklearn.ensemble import VotingClassifier

X_train1,X_test1 = rescaling (X_train, X_test, 'minmax')
############
train,test = feature_selection('decision_trees',X_train1, X_test1)
############

eclf1 = VotingClassifier(estimators=models, voting='hard')
eclf1 = eclf1.fit(train, Y_train)
result1 = eclf1.predict(test)

eclf2 = VotingClassifier(estimators=models,voting='soft')
eclf2 = eclf2.fit(train, Y_train)
result2 = eclf2.predict(test)

eclf3 = VotingClassifier(estimators=models, voting='soft',
                         weights=[3,2,3,1,2,1]
                         #weights = agg_weights
                         , flatten_transform=True)
eclf3 = eclf3.fit(train, Y_train)
result3 = eclf3.predict(test)


print('hard')
print (estimate_model(result1),'\n\n')
print('soft')
print (estimate_model(result2),'\n\n')
print('with weights')
print (estimate_model(result3),'\n\n')

bag_best_minmax = VotingClassifier(estimators=models, voting='soft',
                         weights = agg_weights, flatten_transform=True)




from sklearn.ensemble import VotingClassifier
```

```
X_train1,X_test1 = rescaling (X_train, X_test, 'stand')
############
train,test = feature_selection('decision_trees',X_train1, X_test1)
###########

eclf1 = VotingClassifier(estimators=models, voting='hard')
eclf1 = eclf1.fit(train, Y_train)
result1 = eclf1.predict(test)

eclf2 = VotingClassifier(estimators=models,voting='soft')
eclf2 = eclf2.fit(train, Y_train)
result2 = eclf2.predict(test)

eclf3 = VotingClassifier(estimators=models, voting='soft',
                    #weights=[3,2,1,1]
                    weights = agg_weights1
                    , flatten_transform=True)
eclf3 = eclf3.fit(train, Y_train)
result3 = eclf3.predict(test)


print('hard')
print (estimate_model(result1),'\n\n')
print('soft')
print (estimate_model(result2),'\n\n')
print('with weights')
print (estimate_model(result3),'\n\n')

bag_best_stand = VotingClassifier(estimators=models, voting='soft',
                    weights = agg_weights1, flatten_transform=True)




X_train1,X_test1 = rescaling (X_train, X_test, 'stand')
############
train,test = feature_selection('decision_trees',X_train1, X_test1)
###########

print ("NB")
NB.fit(train,Y_train)
print (estimate_model(NB.predict(test)),'\n')

print ("KNN")
KNN.fit(train,Y_train)
print (estimate_model(KNN.predict(test)),'\n')

print ("DT")
DT.fit(train,Y_train)
print (estimate_model(DT.predict(test)),'\n')

print ("SVM")
```

```
SVM.fit(train,Y_train)
print (estimate_model(SVM.predict(test)),'\n')


# # Stacking


models1 = [ExtraTreesClassifier(random_state=41, n_estimators=20, max_depth=10),
        RandomForestClassifier(random_state=41, n_estimators=20, max_depth=10),
        XGBClassifier(random_state=41,n_estimators=20,
max_depth=2,nthread=4,objective='binary:logistic')]

models3 = [bagging_best,
         ExtraTreesClassifier(random_state=41, n_estimators=20, max_depth=10),
        #MLPClassifier(activation='tanh',learning_rate='adaptive',alpha =0.01, max_iter=200),
        #LDA,
         MLPClassifier(activation='tanh',learning_rate='adaptive',alpha =0.01, max_iter=200),
        #RandomForestClassifier(random_state=41, n_estimators=20, max_depth=10),
       #XGBClassifier(random_state=41,n_estimators=20,
max_depth=2,nthread=4,objective='binary:logistic')
        ]

m_minmax = [bag_best_minmax,
         ExtraTreesClassifier(random_state=41, n_estimators=20, max_depth=10),
         MLPClassifier(activation='tanh',learning_rate='adaptive',alpha =0.01, max_iter=200),
         ]
m_stand = [bag_best_stand,
         ExtraTreesClassifier(random_state=41, n_estimators=20, max_depth=10),
         MLPClassifier(activation='tanh',learning_rate='adaptive',alpha =0.01, max_iter=200),
         ]

X_train1,X_test1 = rescaling (X_train, X_test, 'minmax')
train,test = feature_selection('decision_trees',X_train1, X_test1)

X_train2,X_test2 = rescaling (X_train, X_test, 'stand')
train2,test2 = feature_selection('decision_trees',X_train2, X_test2)

# ### ---------
print ("Bagging result minmax")
clf = bag_best_minmax
clf.fit(train,Y_train)
print (estimate_model(clf.predict(test)),'\n')

print ("Bagging result stand")
clf = bag_best_stand
clf.fit(train2,Y_train)
print (estimate_model(clf.predict(test2)),'\n')


print ("ExtraTrees")
clf = ExtraTreesClassifier(random_state=41, n_estimators=20, max_depth=10)
clf.fit(train,Y_train)
print (estimate_model(clf.predict(test)),'\n')
```

```
#print ('LDA')
#clf = LDA
#clf.fit(train,Y_train)
#print (estimate_model(clf.predict(test)),'\n')

#print ('SGD')
#clf = SGD
#clf.fit(train,Y_train)
#print (estimate_model(clf.predict(test)),'\n')


print ('MLP')
clf = MLPClassifier(activation='tanh',learning_rate='adaptive',alpha =0.01, max_iter=200)
clf.fit(train,Y_train)
print (estimate_model(clf.predict(test)),'\n')

'''print ("RF")
clf = RandomForestClassifier(random_state=41, n_estimators=20, max_depth=10)
clf.fit(train,Y_train)
print (estimate_model(clf.predict(test)),'\n')'''

'''print ("XGB")
clf = XGBClassifier(random_state=41,n_estimators=20,
max_depth=2,nthread=4,objective='binary:logistic')
clf.fit(train,Y_train)
print (estimate_model(clf.predict(test)),'\n')
'''


S_train, S_test = stacking(models3,                   # list of models
                           train, Y_train, test,   # data
                           regression=False,           # classification task (if you need
                                                       #     regression - set to True)
                           mode='oof_pred_bag',        # mode: oof for train set, predict test
                                                       #     set in each fold and vote
                           needs_proba=False,          # predict class labels (if you need
                                                       #     probabilities - set to True)
                           save_dir=None,              # do not save result and log (to save
                                                       #     in current dir - set to '.')
                           metric=accuracy_score,      # metric: callable
                           n_folds=4,                  # number of folds
                           stratified=True,            # stratified split for folds
                           shuffle=True,               # shuffle the data
                           random_state=241,           # ensure reproducibility
                           verbose=2)                  # print all info

# Initialize 2nd level model
#model = XGBClassifier(random_state=41,  learning_rate=0.1, n_estimators=20, max_depth=10)
model = LogisticRegression(random_state=41,class_weight=weights)
#model = XGBClassifier(random_state=41,n_estimators=20,
max_depth=2,nthread=4,objective='binary:logistic')
```

```
# Fit 2nd level model
model = model.fit(S_train, Y_train)

# Predict
y_pred = model.predict(S_test)

# Final prediction score
print('Final prediction accuracy score: [%.8f]' % accuracy_score(Y_test, y_pred))
estimate_model(y_pred)


# ### --------------------


LR = LogisticRegression(random_state=241,class_weight=weights)
NB = BernoulliNB()
KNN = KNeighborsClassifier(n_neighbors=9)
SVM = SVC(C=0.1, kernel='linear',class_weight=weights, probability=True)
DT = DecisionTreeClassifier(random_state=241, splitter='random')
RF= RandomForestClassifier(n_estimators=20, max_depth=10,random_state=41)
SGD = SGDClassifier(max_iter=10000,loss='modified_huber',class_weight={1:0.4,0:0.6})
LDA = LinearDiscriminantAnalysis()


models1 = [#ExtraTreesClassifier(random_state=41, n_estimators=20, max_depth=10),
           #MLPClassifier(activation='tanh',learning_rate='adaptive',alpha =0.01, max_iter=200),
          LDA,
          SGD,
         #RandomForestClassifier(random_state=41, n_estimators=20, max_depth=10),
         XGBClassifier(random_state=41,n_estimators=10,
max_depth=2,nthread=4,objective='binary:logistic')
         ]

X_train1,X_test1 = rescaling (X_train, X_test, 'stand')
train,test = feature_selection('decision_trees',X_train1, X_test1)

print ("ExtraTrees")
clf = ExtraTreesClassifier(random_state=41, n_estimators=20, max_depth=10)
clf.fit(train,Y_train)
print (estimate_model(clf.predict(test)),'\n')

print ('LDA')
clf = LDA
clf.fit(train,Y_train)
print (estimate_model(clf.predict(test)),'\n')

print ('SGD')
clf = SGD
clf.fit(train,Y_train)
print (estimate_model(clf.predict(test)),'\n')


print ('MLP')
```

```
clf = MLPClassifier(activation='tanh',learning_rate='adaptive',alpha =0.01, max_iter=200)
clf.fit(train,Y_train)
print (estimate_model(clf.predict(test)),'\n')

print ("RF")
clf = RandomForestClassifier(random_state=41, n_estimators=20, max_depth=10)
clf.fit(train,Y_train)
print (estimate_model(clf.predict(test)),'\n')

print ("XGB")
clf = XGBClassifier(random_state=41,n_estimators=20,
max_depth=2,nthread=4,objective='binary:logistic')
clf.fit(train,Y_train)
print (estimate_model(clf.predict(test)),'\n')


S_train, S_test = stacking(models1,                  # list of models
                    train, Y_train, test,   # data
                    regression=False,           # classification task (if you need
                                #     regression - set to True)
                    mode='oof_pred_bag',        # mode: oof for train set, predict test
                                #     set in each fold and vote
                    needs_proba=False,          # predict class labels (if you need
                                #     probabilities - set to True)
                    save_dir=None,              # do not save result and log (to save
                                #     in current dir - set to '.')
                    metric=recall_score,      # metric: callable
                    n_folds=4,                 # number of folds
                    stratified=True,           # stratified split for folds
                    shuffle=True,              # shuffle the data
                    random_state=41,            # ensure reproducibility
                    verbose=2)                 # print all info


# Initialize 2nd level model
#model_stack = XGBClassifier(random_state=41,  learning_rate=0.1, n_estimators=20,
max_depth=10)
model_stack = LogisticRegression(random_state=41,class_weight=weights)
#model_stack = XGBClassifier(random_state=41,n_estimators=20,
max_depth=2,nthread=4,objective='binary:logistic')

# Fit 2nd level model
model_stack = model_stack.fit(S_train, Y_train)

# Predict
y_pred = model_stack.predict(S_test)

# Final prediction score
print('Final prediction accuracy score: [%.8f]' % accuracy_score(Y_test, y_pred))
res_stacking = y_pred
estimate_model(y_pred)
```

```
X_train1,X_test1 = rescaling (X_train, X_test, 'minmax')
train,test = feature_selection('decision_trees',X_train1, X_test1)

X_train2,X_test2 = rescaling (X_train, X_test, 'stand')
train2,test2 = feature_selection('decision_trees',X_train2, X_test2)

#minmax
NB.fit(train,Y_train)
res_nb = NB.predict(test)
print('NB', estimate_model(res_nb),'\n')

KNN.fit(train, Y_train)
res_knn = KNN.predict(test)
print('KNN', estimate_model(res_knn),'\n')

DT.fit(train, Y_train)
res_dt = DT.predict(test)
print('DT', estimate_model(res_dt),'\n')

SVM.fit(train,Y_train)
res_svm = SVM.predict(test)
print('SVM', estimate_model(res_nb),'\n')


RES = (res_nb + res_knn + res_dt + res_svm + res_stacking)/5

res_res = []
for i in range(len(RES)):
    if RES[i]>0.5:
        res_res.append(1)
    else:
        res_res.append(0)

estimate_model(res_res)


RES1 = (0.89*res_nb + 0.92*res_knn + 0.91*res_dt + 0.89*res_svm + 0.93*res_stacking)/5
res_res1 = []
for i in range(len(RES1)):
    if RES1[i]>0.5:
        res_res1.append(1)
    else:
        res_res1.append(0)

estimate_model(res_res1)

# ----------------------

df = pd.read_csv('kidney_disease.csv',index_col=0)


for i in df.columns:
    print(i)
```

```
    print(df[i].unique())
    print ('\n')


def replace_Y_N(row):
    for col in df.columns:
        if (row[col] == "abnormal")|(row[col] == 'present')|(row[col] == 'yes')|(row[col] ==
'\tyes')|(row[col] == ' yes')|(row[col] == 'good'):
            row[col] = 1
        if (row[col] == "normal") | (row[col] == 'notpresent') |(row[col] == '\tno') |(row[col] ==
'no')|(row[col] == 'poor'):
            row[col] = 0
    return row

df = df.apply(replace_Y_N,axis = 1)

df = df.drop(labels = [66,162], axis = 0)


df['rbc'].fillna(0,inplace = True)
df['bp'].fillna(math.ceil(df['bp'].mean()), inplace = True)
df['age'].fillna(math.ceil(df['age'].mean()), inplace = True)
df['sg'].fillna(df['sg'].median(), inplace = True)
df['al'].fillna(df['al'].median(), inplace = True)
df['su'].fillna(df['su'].median(), inplace = True)
df['pc'].fillna(df['pc'].median(), inplace = True)
df['pcc'].fillna(df['pcc'].median(), inplace = True)
df['ba'].fillna(df['ba'].median(), inplace = True)
df['bgr'].fillna(df['bgr'].mean(), inplace = True)
df['bu'].fillna(math.ceil(df['bu'].mean()), inplace = True)
df['sc'].fillna(df['sc'].median(), inplace = True)
df['sod'].fillna(math.ceil(df['sod'].mean()), inplace = True)
df['pot'].fillna(df['pot'].median(), inplace = True)
df['hemo'].fillna(df['hemo'].median(), inplace = True)
df['pcv'].replace({'\t43':0},inplace = True)
df['pcv'].fillna(df['pcv'].median(),inplace = True)
df['rc'].replace({'\t?': df['rc'].median()},inplace = True)
df['rc'].fillna(df['rc'].median(),inplace = True)

df.loc[76,'wc'] = 6200
df.loc[133,'wc'] = 8400
df.loc[185,'wc'] = np.nan
df['wc'].fillna(df['wc'].median(),inplace = True)


l = ['htn', 'dm', 'cad', 'appet', 'pe', 'ane']
for col in l:
    df[col].fillna(df[col].median(),inplace = True)

Y = df['classification'].replace({'ckd':1, 'ckd\t':1, 'notckd' :0})
df = df.drop(labels = ['classification'],axis = 1)

data2 = df
```

```
get_ipython().run_line_magic('store', 'data2')
Y.value_counts()

x = df
x.shape


X_train,X_test, Y_train, Y_test = train_test_split(x, Y, test_size=0.3, random_state=142)

class_weight = class_weight.compute_class_weight('balanced', np.unique(Y_train), Y_train)
class_weight
weights = {1:0.81538462,0:1.29268293}


# # ----------
get_ipython().run_line_magic('store', 'x')
get_ipython().run_line_magic('store', 'Y')
```