**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**
**ІМЕНІ ТАРАСА ШЕВЧЕНКА**
Факультет комп'ютерних наук та кібернетики
Кафедра прикладної статистики

# Кваліфікаційна робота
**на здобуття ступеня магістра**

за спеціальністю 124 Системний аналіз

на тему:
**ЗАСТОСУВАННЯ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ ДЛЯ ПРОГНОЗУВАННЯ РАКУ МОЛОЧНОЇ ЗАЛОЗИ**

Виконав студент 2-го курсу магістратури
Головін Сергій Миколайович

(підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Шарапов Михайло Михайлович

(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент

(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри прикладної статистики
10 травня 2023 р.,
протокол № 9
Завідувач кафедри
І. В. Розора

(підпис)

Київ – 2023

## РЕФЕРАТ

Обсяг роботи 66 сторінка, 27 ілюстрацій, 20 джерел посилань.

КЛАСИФІКАЦІЙНИЙ АНАЛІЗ, ЛОГІСТИЧНА РЕГРЕСІЯ, МЕТОД ГОЛОВНИХ КОМПОНЕНТ, RECURSIVE FEATURE ELIMINATION, DATA MINING, ДОСЛІДНИЦЬКИЙ АНАЛІЗ ДАНИХ, КРОСС ВАЛІДАЦІЯ, МОВА ПРОГРАМУВАННЯ PYTHON.

Об’єктом дослідження є метод класифікації ракової пухлини на основі класифікаційного групування відповідно до ознак подібності або відмінності.

Предметом дослідження є використання логістичної регресії зі знайденням оптимального гіперпараметру та оптимізацією розмірності методом головних компонент.

Метою роботи є узагальнити метод класифікації ракової пухлини з використанням логістичної регресії, пошуку гіперпараметру та методу головних компонент.

Методи розроблення: методи очистки та підготовки дослідницького аналізу, метод класифікаційного аналізу даних логістична регресія з пошуком оптимального гіперпараметру, метод головних компонент, методи оцінювання якості моделі.

Інструмент та середовище розробки: мова програмування Python та Jupyter Notebook.

Значимість роботи полягає в удосконаленні алгоритму аналізу ракових пухлин шляхом використання гіперпараметру моделі, а також методу головних компонент з метою зменшення навантаження на систему та покращення якості класифікації шляхом відсіювання зайвих ознак.

Модель логістичної регресії з використанням методу головних компонент для класифікації ракових пухлин може знайти широке застосування у медицині, при аналізах захворювання та побудові лікування.

# ЗМІСТ

# ВСТУП

**Оцінка сучасного стану об’єкта розробки.** Сьогодні сучасні інструменти для аналізу раку включають широкий спектр методології і алгоритмів машинного навчання. Вони допомагають у діагностиці та класифікації злоякісних пухлин на основі медичних записів, даних аналізів та професійних записах лікарів. Велика кількість методів класифікації відрізняються способами використання та ситуаціями, коли кожен з них краще використовувати. Одним з найпоширенішим у класифікації ракових захворювань є метод логістичної регресії. Вона моделює ймовірність приналежності до класу типу раку на основі ознак у наборі даних, що використовуються для дослідження. До набору даних найчастіше належать результати специфікованих медичних досліджень та аналізів пацієнта.

Також існує велика кількість інших методів класифікаційного аналізу, які активно використовуються: метод головних опорних векторів (SVM), випадковий ліс, наївний байесовий класифікатор, глибинне навчання, тощо.

**Актуальність роботи.** Рак продовжує залишатися однією з найважливіших проблем здоров’я, з якою стикаються люди в усьому світі. Такі методи класифікаційного аналізу, як логістична регресія, опорні векторні машини (SVM), випадкові ліси та глибинне навчання, активно досліджуються та використовуються в боротьбі з раком у світовій науковій та медичній спільноті. Вони мають значний вплив на процес ідентифікації, класифікації та діагностики злоякісних новоутворень, а також на вибір відповідних методів лікування та організацію медичної допомоги.

**Мета й завдання роботи.** Метою кваліфікаційної роботи є побудувати модель класифікаційної логістичної регресії зі знайденим гіперпараметром та оптимізованою розмірністю з використанням методу головних компонент. Для досягнення цієї мети поставлено такі завдання:

1) Провести дослідницький аналіз датасету, що використовується в роботі. Підготувати дані до подальшого аналізу.

1) Провести перевірку ознак з використанням кореляційної матриці.

2) Розробити алгоритм створення датасету з 2, 3 та 6 компонентами на базі методу головних компонент та перевірити вплив на точність моделі, навести приклади візуалізацій для кожного з випадків.

3) Розділити вибірка на тестову та тренувальну, знайти гіперпараметр для моделі логістичної регресії та побудувати її з використанням бібліотеки scikit–learn.

4) Створити модифіковану модель логістичної регресії з використанням рекурсивного виключення ознак (RFE) для подальшої її роботи з великими даними при використані моделі на більш великому обсягу даних.

5) Перевірка та порівняння оцінок якості обох моделей за допомогою матриці помилок, ROC графіку, кросс–валідації та графіку навчання. Зробити висновки щодо їх використання.

**Об'єкт, методи й засоби розроблення**. Об’єктом дослідження є метод класифікації ракової пухлини на основі класифікаційного групування відповідно до ознак подібності або відмінності. При виконанні роботи використовувались такі методи:

1) методи очистки та підготовки дослідницького аналізу;

2) метод класифікаційного аналізу даних логістична регресія з пошуком оптимального гіперпараметру;

3) метод головних компонент;

4) методи оцінювання якості моделі.

В якості інструменту створення програмного забезпечення було обрано Jupyter Notebook, мова програмування Python.

**Можливі сфери застосування.** Модель логістичної регресії з використанням методу головних компонент для класифікації ракових пухлин

може знайти широке застосування у медицині, при аналізах захворювання та побудові лікування.

# РОЗДІЛ 1 ОГЛЯД ОСНОВНИХ МЕТОДІВ ДЛЯ ДОСЛІДНИЦЬКОГО АНАЛІЗУ ДАНИХ ТА ПІДГОТОВКИ ДАНИХ

## 1.1 Методи Data Mining для вирішення задач класифікації

Методи Data Mining допомагають вирішити багато завдань, із якими стикається аналітик. Серед них основними є: класифікація, регресія, пошук асоціативних правил та кластеризація. Завдання класифікації зводиться до визначення класу об'єкта за його характеристиками. Слід зазначити, що у цій задачі кількість класів, до яких можна віднести об'єкт, відомо заздалегідь.

Завдання регресії подібно до завдання класифікації дозволяє визначити за відомим характеристикам об'єкта значення деякого параметра. На відміну від завдання класифікації значенням параметра не кінцева безліч класів, а безліч дійсних чисел. При пошуку асоціативних правил метою є знаходження залежностей (чи асоціацій) між об'єктами чи подіями. Знайдені залежності подаються у вигляді правил і можуть бути використані як для кращого розуміння природи даних, що аналізуються, так й для передбачення появи подій.

Завдання кластеризації полягає у пошуку незалежних груп (кластерів) та їх характеристик у всій кількості аналізованих даних. Вирішення цього завдання допомагає краще зрозуміти дані. Крім того, угруповання однорідних об'єктів дозволяє скоротити їх число, а отже, і полегшити аналіз. Перелічені завдання призначення діляться на описові і передбачувані [5].

Метою описової діяльності є покращення розуміння даних, що оцінюються. Головна особливість таких моделей полягає в тому, наскільки прості та зрозумілі висновки для людського сприйняття вона робить. Навіть якщо спостережувані закономірності є унікальними для конкретних даних дослідження і їх неможливо знайти в інших місцях, вони все одно можуть бути корисними, тому важливо їх знати. Цей вид проблеми передбачає пошук і групування правил асоціації.

Другий тип завдань відповідає на проблеми прогнозування. На основі набору даних із заздалегідь визначеними результатами на початковому етапі створюється модель. Вона використовується для прогнозування результатів на наступному етапі за допомогою свіжих наборів даних. Природно, побудовані моделі повинні функціонувати настільки коректно, наскільки це можливо в даній ситуації. Проблеми класифікації та регресії є прикладами такого роду проблем.

Навчання без нагляду (unsupervised learning) і навчання під наглядом (supervised learning) – це категорії, які базуються на тому, як вирішуються задачі моделі. Цей псевдонім походить від слова «machine learning», яке часто використовується для позначення всіх технологій інтелектуального аналізу даних в англомовній літературі.

Питання аналізу даних вирішується поетапно при використанні методу навчання під наглядом. Спочатку з досліджуваних даних за допомогою будь–якого методу інтелектуального аналізу даних створюється класифікатор. Після цього проводиться навчання класифікатора. Іншими словами, класифікатор проходить подальше навчання, якщо якість його роботи виявляється нижчою. Це триває до тих пір, поки не буде досягнуто необхідного рівня якості, стане очевидним, що обраний метод не обробляє дані належним чином, або самі дані не мають ідентифікованої структури. Проблеми регресії та класифікації належать до цієї категорії.

Навчання без вчителя поєднує в собі завдання, які розкривають описові закономірності, такі як тенденції в купівельній поведінці клієнтів у великому роздрібному магазині. Такі проблеми мають перевагу в тому, що їх можна вирішити навіть без попереднього знання досліджуваних даних. Кластеризація та пошук правил асоціації – це деякі з цих методів.

При аналізі часто потрібно класифікувати предмети, що досліджуються, або встановити, до якого з відомих класів вони належать, перш ніж проводити аналіз. Наприклад, банківський службовець повинен визначити, чи є

потенційний споживач кредитоспроможним, коли він отримує заявку на кредит. Такий вибір, безсумнівно, ґрунтується на інформації про суб’єкта запиту (у даному прикладі – про особу), наприклад про місце роботи, оплату праці, вік, склад сім’ї. Співробітники банку після оцінки цих даних повинні віднести особу до однієї з двох добре відомих категорій «кредитоспроможність» або «некредитоспроможність».

Фільтрування електронної пошти є ще одним прикладом завдання категоризації. У цьому випадку програма фільтрації повинна класифікувати вхідне повідомлення як лист або спам (спам = небажана електронна пошта). Частота певних термінів у повідомленні, таких як ім’я одержувача, неособиста адреса, а також слова та фрази «купити», «заробити» та «найкраща пропозиція», наприклад, береться до уваги під час вибору.

У більшості ситуацій завдання категоризації можуть мати більше двох класів. За кількістю цифр у десятковій системі числення таких класів може бути 10, наприклад, для ідентифікації зображення цифр. Об'єктом класифікаційного завдання є зображення розпізнаваної цифри, представлене матрицею пікселів. У цьому випадку властивістю елемента, що аналізується, є колір кожного пікселя.

Під класифікацією в інтелектуальному аналізі даних розуміється процес визначення значення одного з параметрів об’єкта, що вивчається, на основі значень інших параметрів. Параметри, які беруть участь у його визначенні, називаються незалежними змінними, тоді як параметр, який визначається, іноді називають залежною змінною. Незалежними змінними в розглянутих випадках були:

- зарплата, вік, кількість дітей тощо;
- частота появи певних слів;
- значення кольору пікселів матриці.

Залежними змінними у цих прикладах були відповідно:

- кредитоспроможність клієнта (можливі значення цієї змінної – "так" і "ні");
- тип повідомлення (можливі значення цієї змінної – "spam" та "mail");
- цифра образу (можливі значення цієї змінної – 0, 1, ..., 9).

Слід підкреслити, що незалежна змінна в кожному з випадків отримувала значення з обмеженого діапазону, включаючи «так», «ні», «спам», «пошта» та значення від 0 до 9. Проблема відома як задачу регресії, якщо значення незалежної та залежної змінних є дійсними числами. Завдання з'ясувати, яку суму позики банк може надати клієнту, є ілюстрацією проблеми регресії.

Щоб вирішити проблему класифікації та регресії, потрібні два набори даних. Перший набір називається навчальним. Він охоплює речі, для яких можуть бути визначені значення як незалежних, так і залежних змінних. У прикладах, які були показані раніше, деякі можливі зразки навчання такі:

- інформація про клієнтів, яким раніше видавалися кредити на різні суми, та інформація про їх погашення;
- повідомлення, класифіковані вручну як спам чи лист;
- розпізнані раніше матриці образів цифр.

Створено модель для ідентифікації значення залежної змінної, а її побудова інформується навчальною вибіркою. Для позначення цієї функції часто використовують термін «функція класифікації» або «функція регресії». Наступні фундаментальні обмеження накладаються на навчальну вибірку, щоб отримати найвищий можливий рівень точності функції:

Вибірка повинна мати достатню кількість предметів, яких має бути достатньо велика кількість. Коли є більше елементів для класифікації або аналізу, класифікація або регресійна функція, створена на їх основі, буде більш точною.

У разі проблеми з класифікацією вибірка повинна включати об’єкти, які є репрезентативними для всіх потенційних класів; у випадку проблеми, що включає регресію, вибірка повинна містити об'єкти, які охоплюють весь діапазон можливих значень.

Вибірка повинна мати відповідну кількість характеристик для кожного класу в задачі, що включає класифікацію, а також для кожного інтервалу діапазону в задачі, що включає регресію.

На другому етапі побудовану модель застосовують до об'єктів, що аналізуються (до об'єктів з невизначеним значенням залежної змінної).

Завдання класифікації та регресії має геометричну інтерпретацію. Розглянемо її на прикладі з двома незалежними змінними, що дозволить уявити її у двовимірному просторі. Кожному об'єкту ставиться у відповідність точка на площині. Символи "+" та "–" позначають належність об'єкта до одного з двох класів. Очевидно, що дані мають чітко виражену структуру: усі точки класу "+" зосереджені у центральній області.

Побудова класифікаційної функції зводиться до побудови поверхні, що обводить центральну область. Вона визначається як функція, що має значення "+" всередині обведеної області та "–" – поза нею.

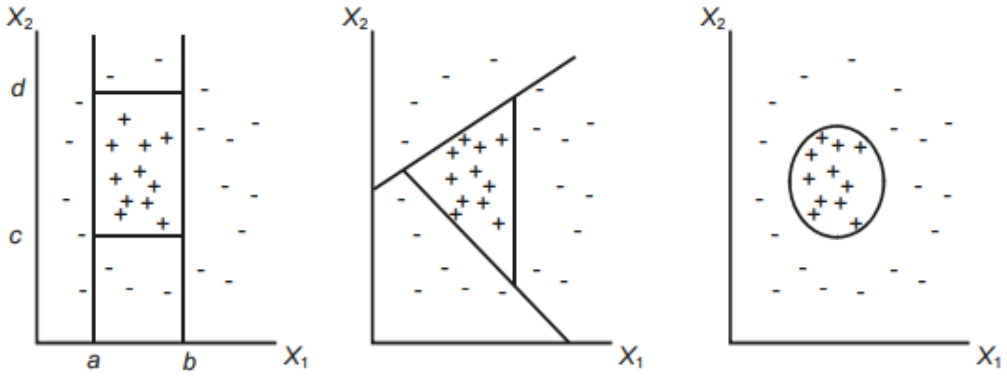



Рисунок 1.1 – Побудова класифікаційної функції

Як видно з малюнка, є кілька можливостей для побудови області, що обводить. Вигляд функції залежить від алгоритму.

Незадовільна якість вихідних даних, які можуть містити як неправильні дані, так і відсутні значення, наявність різних типів атрибутів (числових і категоріальних), відмінності в значущості атрибутів і так звані проблеми переобладнання є основними проблемами, які виникають при спробі вирішити проблеми класифікації та регресії.

Перша з цих помилок полягає в тому, що функція класифікації, коли вона створена «надто добре», адаптується до даних і намагається інтерпретувати помилки та аномальні значення, які містяться в даних, як частину внутрішньої структури даних. У цьому полягає суть першої з цих вад. У майбутньому цілком очевидно, що така модель не працюватиме добре з іншими типами даних, у яких характеристики помилок будуть суттєво змінені.

Недостатність – це обставина, яка виникає, коли виникає надмірна кількість помилок під час перевірки класифікатора з використанням даних із навчального набору. Це вказує на те, що в даних немає унікальних закономірностей, або їх немає взагалі, або потрібно буде прийняти альтернативну стратегію для їх виявлення.

Оскільки в медичних і біологічних дослідженнях, а також у клінічній медицині існує така велика різноманітність проблем, які необхідно вирішувати, можливо використовувати будь–який із багатьох підходів інтелектуального аналізу даних. Створення діагностичної системи або дослідження ефективності хірургічного втручання є двома прикладами досліджень, які підпадають під цю категорію.

В даний час існує велика різноманітність спеціалізованого діагностичного програмного забезпечення.

Вони здебільшого побудовані на основі правил, які описують різні комбінації симптомів, пов’язані з певними захворюваннями. За допомогою таких приписів можна не тільки визначити, якою хворобою хворий пацієнт, але і як його лікувати.

Методичні рекомендації допомагають у виборі методу лікування, визначенні показань і протипоказань, орієнтації в лікувальних процесах, встановленні обставин для максимально успішного лікування, прогнозуванні результатів зазначеного курсу лікування та багато інших завдань. Технології інтелектуального аналізу даних дозволяють розпізнавати шаблони в медичних даних, які є фундаментальними будівельними блоками цих правил.

Однією з найсучасніших галузей медицини є біоінформатика, яка є галуззю науки, яка розробляє та застосовує обчислювальні алгоритми для аналізу та систематизації генетичної інформації з метою з'ясування структури та функції макромолекул, подальшого використання цього знання для пояснення різних біологічних явищ і створення нових ліків (Drug Design). Біоінформатика – одна з найсучасніших галузей медицини.

Величезна кількість інформації про послідовності ДНК і фундаментальну структуру білків, яка була відкрита як прямий результат дослідження структури геномів мікробів, тварин і людини, є темою, яку біоінформатика прагне дослідити та зрозуміти.

Якщо ми відійдемо від деталей змісту цієї інформації, ми можемо думати про неї як про набір генетичних текстів, які складаються з розширених послідовностей символів. Одним із завдань, з якими можуть ефективно справлятися інструменти інтелектуального аналізу даних, є ідентифікація структурних шаблонів у таких послідовностях. Одним із методів, який можна використовувати для цієї мети, є послідовний і асоціативний аналіз.

Створення ліків наступного покоління, які кардинально змінять практику сучасної медицини, є основною сферою, в якій біоінформатика використовується в реальному світі. Дослідження та розробка одного препарату в Сполучених Штатах зараз займає в середньому від десяти до дванадцяти років і коштує від трьохсот до п'ятисот мільйонів доларів. Сфера біоінформатики скорочує ці цифри вдвічі. Використовуючи обладнання інтелектуального аналізу даних як основу, біоінформатика має потенціал для

значного прискорення дофармакологічної фази дослідження нових фармацевтичних препаратів, а також скорочення витрат на це.

## 1.2 Методи дослідницького аналізу та підготовки даних перед побудовою моделі

Дослідницький аналіз даних (EDA) використовується дослідниками даних для аналізу та дослідження наборів даних і узагальнення їхніх основних характеристик, часто використовуючи методи візуалізації даних. Це допомагає визначити, як найкраще маніпулювати джерелами даних, щоб отримати потрібні відповіді, полегшуючи дослідникам даних виявлення закономірностей, виявлення аномалій, перевірку гіпотез або перевірку припущень.

EDA використовується здебільшого для вивчення того, що дані можуть розкрити за межами формального моделювання чи перевірки гіпотез, а також дає глибші знання про змінні, включені в колекцію даних, а також про зв’язки між цими змінними. Крім того, це може допомогти визначити, чи підходять статистичні методи, які розглядаються для аналізу даних. Підходи EDA, які вперше були створені як інструмент для процесу виявлення даних у 1970–х роках американським математиком Джоном Тьюкі, продовжують залишатися одним із найпопулярніших методів, що використовуються в галузі сьогодні.

Основна мета EDA – надати допомогу в перегляді даних перед тим, як робити будь–які припущення. Його можна використовувати для виявлення очевидних помилок, а також для кращого розуміння закономірностей у даних, виявлення викидів або незвичайних випадків і встановлення цікавих зв’язків між змінними.

Дослідницький аналіз – це метод, який можуть використовувати дослідники даних, щоб гарантувати, що створені ними висновки є законними та можуть бути застосовані до будь–яких і всіх передбачуваних наслідків і цілей компанії. Крім того, EDA допомагає зацікавленим сторонам,

гарантуючи, що вони задають відповідні запитання. На питання про стандартні відхилення, категоричні змінні та довірчі інтервали можна отримати відповідь за допомогою EDA. Після того, як EDA буде завершено та з нього буде взято розуміння, його характеристики можна буде використовувати для більш розширеного аналізу даних або моделювання, яке може включати машинне навчання [12].

Нижче наведено список конкретних статистичних функцій і підходів, які можна виконувати за допомогою інструментів EDA:

- Методи кластеризації та зменшення розмірності, які допомагають побудувати графічні представлення даних великої розмірності, що містять численні змінні, називаються «методами кластеризації та зменшення розмірності».
- Однофакторне зображення кожного поля в необробленому наборі даних разом зі зведенням статистики.
- Двомірні візуалізації та зведена статистика дають вам можливість оцінити зв’язок, який існує між кожною змінною в наборі даних і змінною, на якій ви зараз зосереджуєтеся.
- Багатоваріантна візуалізація з метою відображення та розуміння зв’язків між багатьма змінними, включеними до даних.
- K–means Clustering – це форма кластеризації, яка використовується в неконтрольованому навчанні. У цьому підході точки даних розміщуються в K груп або кількість кластерів залежно від їх відстані від центроїда кожної групи. Точки даних, які знаходяться на певній відстані від даного центроїда, будуть згруповані під одним заголовком. K–means Clustering – це техніка, яка часто використовується в процесах ідентифікації шаблонів, стиснення зображення та сегментації ринку.

З метою прогнозування майбутніх подій прогнозні моделі, такі як лінійна регресія, використовують статистику та дані.

Є чотири основних типи EDA:

1. Дані одновимірні, а не графічні. У такому дослідженні дані, що оцінюються, складаються лише з однієї змінної за раз. Це найпростіша форма аналізу даних. Оскільки задіяна лише одна змінна, немає обговорення зв'язків чи причинно–наслідкового зв'язку. Основними цілями однофакторного аналізу є надання опису даних і виявлення будь–яких закономірностей, які можуть бути присутніми в них.

2. Графіка лише з однією змінною. Методи, які не використовують графіки, не можуть забезпечити точне уявлення про факти. Тому необхідно використовувати графічні підходи. Нижче наведено приклади поширених форм одномірної графіки:

- Діаграми, відомі як діаграми стебла та листя, відображають усі дані, а також форму розподілу.
- Гістограми – це різновид стовпчастої діаграми, на якій кожна смужка показує або частоту (кількість), або відсоток (кількість/загальна кількість) випадків для заданого діапазону значень.
- Ящичні діаграми – це графіки, які містять зведення п'яти чисел: найнижчого, першого квартиля, медіани та третього квартиля.

3. Неграфічне представлення багатовимірних даних Багатовимірні дані є результатом взаємодії більш ніж однієї змінної. Перехресна табуляція та статистика – це два методи, які часто використовуються в багатовимірних неграфічних підходах EDA, щоб проілюструвати зв'язок, який існує між двома або більше змінними даних.

4. Графічне представлення багатовимірних даних

Багатовимірні дані використовують візуальні елементи для демонстрації зв’язків між двома чи більше типами даних. Найпоширенішим видом візуалізації є згрупована стовпчаста діаграма або стовпчаста діаграма, у якій кожна група представляє один рівень однієї зі змінних, а кожна смуга всередині групи представляє рівні іншої змінної. Цей тип графіка або діаграми є найпоширенішим.

Нижче наведено кілька прикладів деяких типових форм багатоваріантної графіки:

Точкова діаграма – це свого роду графік, який відображає ступінь впливу на одну змінну іншої шляхом нанесення точок даних уздовж горизонтальної та вертикальної осі.

Багатовимірна діаграма – це графічне зображення зв’язків між змінними, які досліджувались, і знайденими відповідями.

Діаграма пробігу – це лінійний графік, на якому точки даних відображаються з плином часу.

Бульбашкова діаграма – це різновид візуалізації даних, яка малює багато кіл або бульбашок у двовимірному представленні даних.

Теплова карта – це графічне представлення даних, у якому значення відображаються кольором. Теплові карти стають все більш популярними.

Деякі з найпоширеніших інструментів науки про дані, які використовуються для створення EDA, включають:

1. Python – це мова комп’ютерного програмування, яка є інтерпретованою, об’єктно–орієнтованою та має динамічну семантику. Вбудовані високо рів неві структури даних, а також динамічна типізація та динамічне зв’язування роблять його особливо привабливим вибором для швидкого створення додатків, а також для використання як мови сценаріїв або з’єднувальної мови для об’єднання існуючих компонентів. Python і EDA – це два інструменти, які можна використовувати разом для пошуку відсутніх

значень у колекції даних. Це необхідно, щоб ви могли вибрати, як обробляти відсутні значення для машинного навчання.

2. R – це безкоштовна мова програмування з відкритим кодом, яка використовується для статистичних обчислень і графіки. Він підтримується R Foundation for Statistical Computing (RFSC). Мова програмування R широко використовується в галузі науки про дані статистиками для проведення статистичних спостережень і аналізу даних.

# РОЗДІЛ 2 ЛОГІСТИЧНА РЕГРЕСІЯ ТА ЇЇ ПОБУДОВА. МОДИФІКАЦІЯ МОДЕЛІ ЗА ДОПОМОГОЮ МЕТОДУ ГОЛОВНИХ КОМПОНЕНТ

## 2.1 Кореляція, її види та методи виявлення кореляційного зв’язку

У даних з медичною інформацією кореляція має широке застосування у визначенні статистичного зв’язку між двома або більше змінними, які було замірено підчас проведення медичних досліджень. Кореляційна матриця допомагає оцінити ступінь залежності однієї змінної від іншої у вибірці даних.

Можна розрізнити прямі та зворотні кореляції в парній кореляції на основі характеристик змін, які відбуваються в x і y. Значення обох характеристик зсуваються в одному напрямку, коли між ними існує прямий зв'язок; точніше, коли відбувається збільшення (або зменшення) значень x, значення y так само зростають (або падають). Значення факторних і ефективних характеристик можуть змінюватися різними способами, коли забезпечується зворотний зв’язок.

Статистичне дослідження кореляції можна звести до вирішення трьох проблем: встановлення наявності або відсутності таких зв'язків; визначення математичної моделі такого співвідношення; і вимірювання ступеня близькості, яка існує між досліджуваними змінними.

Методи виявлення кореляцій, такі як метод розгляду паралельних даних, коефіцієнт кореляції знаків Фехнера, графічний метод, метод аналітичних груп і метод кореляційних таблиць, використовуються в галузі статистики для визначення існування кореляції. між знаками та природою цього співвідношення.

Метод оцінки паралельних даних (значення x і y в кожному з n одиниць) вимагає, щоб одиниці спостереження були відсортовані в порядку зростання значень факторної ознаки x, а потім поведінка результуючої ознаки y візуально порівняно з цією поведінкою.

Коефіцієнт кореляції за знаком Фехнера є найпростішою мірою ступеня зв’язку двох змінних. Його виводять із порівняння поведінки відхилень окремих значень атрибута x і наступного атрибута y від їхніх відповідних середніх значень. При цьому до уваги беруться не величини відхилень $(\mathrm{x_i} - \bar{\mathrm{x}})$ та $(y_i - \bar{y})$, а їх знаки («+» чи «–»).

Після визначення показників відхилень від середнього значення в кожному рядку оцініть усі можливі пари символів і підрахуйте, скільки разів вони збігаються і скільки разів не збігаються. Після цього визначається коефіцієнт Фехнера, визначаючи відношення різниці чисел пар збігів і розбіжностей ознак до суми цих чисел або до загальної кількості одиниць, які спостерігалися:

$$K_{\text{ф}} = \frac{\sum n_a - \sum n_b}{\sum n_a - \sum n_b}$$

Очевидно, якщо знаки всіх відхилень за кожною ознакою збігатимуться, то $K_{\text{ф}} = 1$, що характеризує наявність прямої зв'язку. Якщо всі знаки не збігатимуться, то $K_{\text{ф}} = -1$ (зворотній зв’язок). Якщо ж $\sum n_a = \sum n_b$, то $K_\phi = 0$.

Отже, як і будь–який показник тісноти зв'язку, коефіцієнт Фехнер може приймати значення від 0 до ±1. Однак якщо $K_\phi = 1$, то це жодною мірою не можна сприймати як свідчення функціональної залежності між x і y.

Графічний метод – це графічне зображення кореляційної залежності, коли кожну пару взаємопов'язаних значень x та y зображують у вигляді точки на площині з координатами x та y у прямокутній системі координат. Сукупність отриманих точок є кореляційне поле, а з'єднуючи послідовно нанесені точки відрізками, отримують ламану лінію, іменовану емпіричною лінією регресії. Візуально аналізуючи графік, можна припустити характер залежності між ознаками x та y.

Метод аналітичних угруповань використовується при великій кількості спостережень для виявлення кореляційного зв'язку між двома кількісними

ознаками Щоб виявити наявність кореляційного зв'язку між двома ознаками, проводиться угруповання одиниць сукупності за факторною ознакою х та для кожної виділеної групи розраховується середнє значення результативної ознаки $\overline{y_j}$. Якщо результативна ознака у залежить від факторної ознаки х, то зміна середнього значення $\overline{y_j}$ буде простежуватися певна закономірність.

Метод кореляційних таблиць передбачає комбінаційне розподіл у таблиці одиниць сукупності за двома кількісними ознаками. Така таблиця будується за типом шахової, тобто в таблиці, що підлягає (рядках), зазначені групи за факторною ознакою х, у присудку (стовпцях) – за результативним у (або навпаки), а в клітинах таблиці на перетині х і у зазначено число випадків збіги кожного значення х із відповідним значенням у.

Вигляд таблиці, а точніше положення частот у ній, можна використовувати, щоб зробити висновки про існування зв’язку, а також шлях, яким він проходить. Якщо частоти розподілені випадковим чином по клітинках таблиці, це майже завжди говорить про відсутність зв’язку між груповими характеристиками (або про те, що взаємозалежність між ними незначна); з іншого боку, якщо частоти тяжіють до однієї з діагоналей і центру таблиці, утворюючи свого роду еліпс, це майже завжди говорить про те, що між характеристиками х і у існує майже лінійний зв’язок.

Можна зробити висновок про прямий лінійний зв’язок із розташування, яке рухається по діагоналі від верхнього лівого кута до нижнього правого кута, тоді як розташування, яке рухається в протилежному напрямку від нижнього лівого кута до верхнього правого кута, означає протилежне.

Метод кореляційних таблиць може бути використаний і для таблиць спряженості різної розмірності. Найпростіша розмірність – 2 × 2 (таблиця «чотирьох полів»), коли за альтернативною ознакою («так» – «ні», «чоловіча стать» – «жіноча» тощо) виділяються 2 групи ознак.

## 2.2 Логістична регресія

Статистична модель, яка зазвичай використовується для моделювання двійкової залежної змінної за допомогою логістичної функції. Інша назва логістичної функції – сигмоїдна функція, яка визначається так:

$$F(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

Ця функція допомагає моделі логістичної регресії стискати значення від (–k, k) до (0, 1). У випадку визначення сигмоїдної функції, інтервал (–k, k) означає, що вхідне значення ознаки x у функцію має знаходитись у межах від –k до k. Змінна k є деяке число, яке визначає обмеження для вхідних змінних, тільки тоді функція буде правильно працювати та повертати значення у інтервалі від 0 до 1.

Логістична регресія в основному використовується для задач бінарної класифікації; однак його можна використовувати для багатокласової класифікації. На рисунку наведено приклад нелінійної роздільності двох класів:

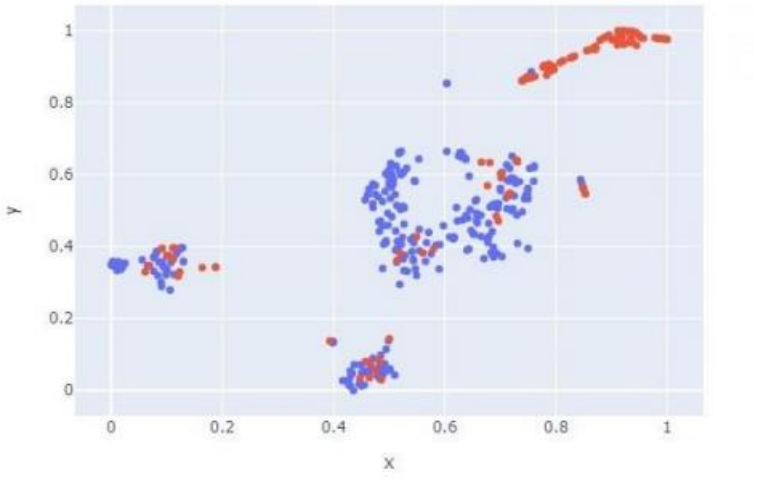



Рисунок 2.1 – Нелінійно розподілені два класи

Рівняння логістичної регресії складається з логарифмів шансів, які потім проходять через сигмоподібну функцію. Ця функція бере результат лінійного рівняння та перетворює його на шанс, що знаходиться десь між 0 і 1. Крім того, ми маємо можливість вибрати межу рішення та використовувати цю

ймовірність у процесі категоризації. Наприклад, уявімо, що ми намагаємося спрогнозувати, чи буде післязавтра дощ, використовуючи наданий набір даних.

Якщо після використання логістичної моделі ймовірність виявиться рівною 90%, то ми точно знаємо, що післязавтра дуже ймовірно, що буде дощ. З іншого боку, якщо ймовірність виявиться рівною 10%, ми можемо стверджувати, що завтра не буде хмарно, і таким чином ми можемо перевести ймовірності в двійкову форму.

Для початку ми можемо зробити припущення, що p(x) є лінійною функцією. Однак проблема полягає в тому, що p має бути ймовірністю в діапазоні від 0 до 1, але p(x) є необмеженим лінійним рівнянням. Це джерело труднощів. Щоб вирішити цю проблему, зробимо припущення, що log p(x) є лінійною функцією x. Крім того, щоб обмежити його в діапазоні [0,1], буде використано логіт–перетворення. У результаті ми розглянемо логарифм p(x), поділений на (1–p(x)). Далі ми збираємося переконатися, що ця функція є лінійною, виконавши такі дії:

$$log\frac{\mathrm{p(x)}}{1-\mathrm{p(x)}} = \mathrm{a_0} + \mathrm{ax}$$

Після вирішення для p(x):

$$P(x) = \frac{e^{a_0+a}}{e^{a_0+a}+1}$$

Ми можемо вибрати певний поріг, наприклад 0,5, щоб перетворити логістичну регресію в лінійний класифікатор. Тепер частоту неправильної класифікації можна звести до мінімуму, якщо ми зробимо прогноз, що y=1, коли p менше 0,5, і y=0, коли p більше 0,5. Класами в цьому сценарії є 1 і 0.

Логістична регресія здатна робити прогнози щодо ймовірностей, тому ми можемо використовувати ймовірність, щоб підібрати її. Таким чином, прогнозований клас для кожної окремої точки даних навчання x дорівнюватиме y. Імовірність y дорівнює або p, якщо y=1, або 1–p, якщо y=0. Імовірність тепер можна виразити такою формулою:

$L(\alpha_0, \alpha) = \prod_{l=1}^{n} p(x_i)^{y_i}(1 - p(x_i)^{1-y_i})$

Множення можна перетворити на суму, взявши логарифм:

$$l(\alpha_0, \alpha) = \sum_{i=0}^{n} y_i log p(x_i) + (1 - y_i) log 1 - p(x_i) =$$

$$= \sum_{i=0}^{n} log 1 - p(x_i) \; + \; \sum_{i=0}^{n} y_i \, log \frac{p(x_i)}{1-p(x_i)}$$

Далі, після встановлення значення p(x):

$$l(\alpha_0, \alpha) = \sum_{i=0}^{n} -log 1 + e^{\alpha_0 + \alpha} + \sum_{i=0}^{n} y_i(\alpha_0 + \alpha_. x_i)$$

У випадку логістичної регресії використовується градієнтний підйом, який протилежний градієнтному спаду. Тому наступним кроком буде взяти максимум функції ймовірності, про яку йшлося раніше.

Роздивимось метод максимальної правдоподібності (Maximum Likelihood Estimation, MLE). Стратегія максимізації значення функції правдоподібності для оцінки параметрів розподілу ймовірностей. Метою цієї стратегії є підвищення ймовірності того, що спостережувані дані відбуваються. MLE можна визначити диференціюванням рівняння, наведеного вище, відносно різних параметрів і встановленням результату на нуль. Наприклад, похідну по одній із складових параметра alpha, яка позначається a_j, можна записати так:

$$\frac{\partial l}{\partial \alpha_j} = \sum_{i=0}^{n} (y_i - \rho \, (x_i; \alpha_0, \alpha)) x_{ij}$$

Точність моделі оцінюється на основі даних із тестового набору, щоб уникнути проблеми переобладнання. Але навіть у цьому сценарії все ще існує можливість зміщення оцінки, тому вони використовують метод, який називається перехресною перевіркою, який іноді пишеться як перехресна перевірка, а іноді просто перехресна перевірка. Контроль такого типу може здійснюватися за допомогою техніки відкладених даних (також відомої як

метод блокування) або контролю за допомогою k–блоків (також відомого як метод k–кратної перехресної перевірки).

### 2.3 Розбиття вибірки на тестову та тренувальну

Підхід із затримкою даних вимагає відокремлення тестових вибірок від навчальних у співвідношенні 70:30 (найчастіше), 60:40 або 80:20. З іншого боку, оцінка похибки буде дещо шумною, незважаючи на те, що вона буде досить близькою до похибки моделі при застосуванні до свіжих даних. Навчальна та тестова вибірки випадковим чином розбиваються багато разів, а потім параметр помилки усереднюється. Це допомагає зменшити вплив шуму. Однак, поки процедура повторюється, кожна точка даних входитиме в тестову підмножину різну кількість разів, що може призвести до зміщення. Бажано вдатися до управління через k–блоки, а не вступати в таку боротьбу з шумом, оскільки це не завжди раціонально.

Дані довільно розбиваються на k різних підмножин, які не перекриваються одна з одною (5, 10 або 20). Після цього модель навчається з використанням k–1 підмножин даних, а решта даних використовується для тестування. Всього процес виконується k разів. Потім отримана оцінка усереднюється після проходження циклічного пошуку кожної з k підгруп.

Відпрацьовано методику оцінки якості продукції. Давайте зараз розглянемо деякі показники якості. На них не повинні впливати параметри моделі та вони повинні давати можливість зробити об’єктивну оцінку точності передбачення мітки класу.

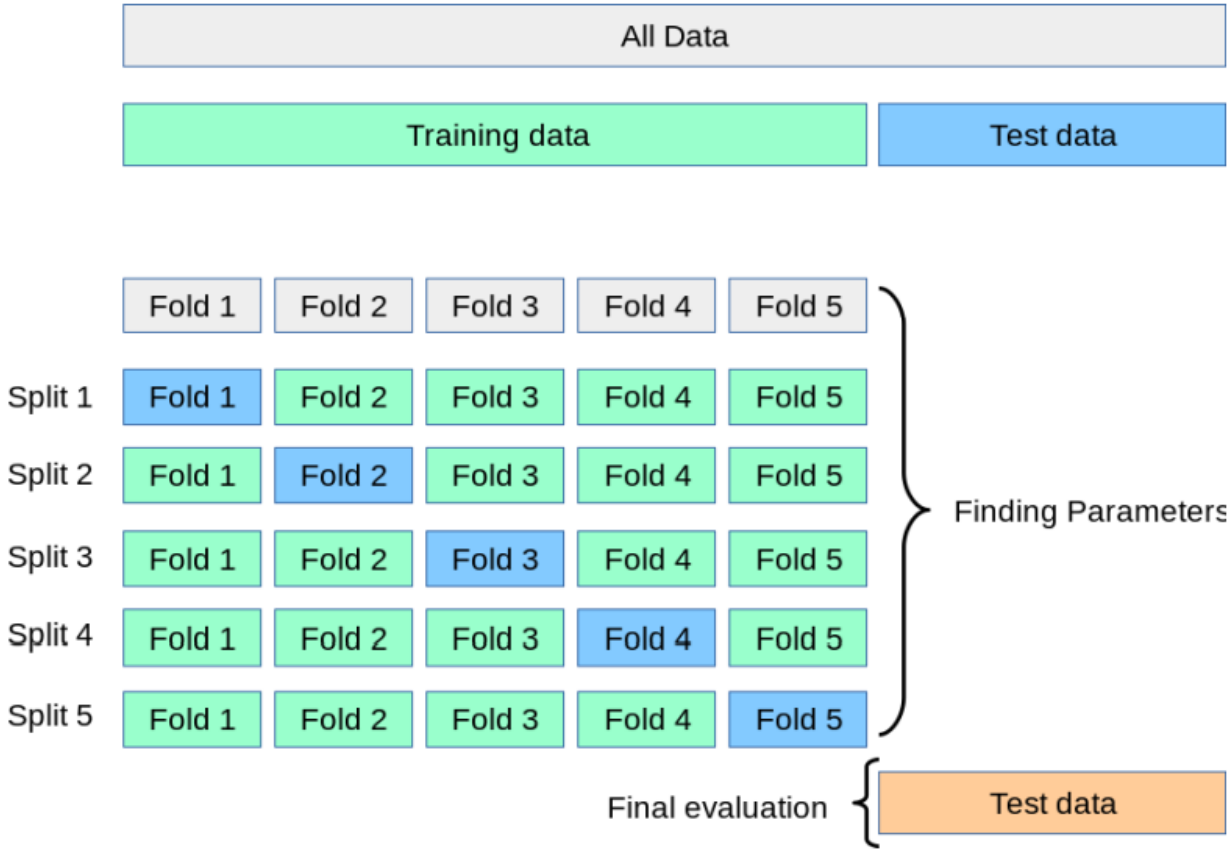



Рисунок 2.2 – Розподіл вибірки на тестову та тренувальну

Щоб обчислити різні метрики якості, спочатку потрібно побудувати матрицю плутанини. Ця матриця дає вам можливість організувати загальну кількість правильно передбачених міток, а також кількість неправильно призначених міток класу. Кількість речей, які належать до кожної категорії, відображається в матриці як числове значення в кожному стовпці.

Таблиця 2.3 – Матриця помилок моделі

| | | Передбачені моделлю класи | |
|---|---|---|---|
| | | A(x)=1 | A(x)=-1 |
| Реальні класи | Y=1 | True positive | False negative |
| | Y=-1 | False positive | True negative |

Існує дві категорії помилок, які можуть виникнути під час класифікації: хибнопозитивні (False Positive) та хибнонегативні (False Negative). У галузі статистики існує два типи помилок: помилки типу I, при яких нульова гіпотеза

помилково відхиляється, і помилки типу II, при яких нульова гіпотеза приймається неправильно.

Загальний підсумок усього, що перевірено, дорівнює сумі всіх значень у матриці помилок.

$$N = TP + FP + FN + TN$$

### 2.4. Перевірка якості моделі з використанням матриці помилок та ROC графіку

Матрицю помилок можна використовувати для створення багатьох надзвичайно поширених показників, включаючи надійність (найбільш використовуваний показник), запам'ятовуваність і точність (завжди йдуть парами), а також інші показники, більш специфічні для інших галузей дослідження.

Відсоток правильних відповідей представлено як надійність або правильність у деяких інших джерелах.

$$accuracy = \frac{TP + TN}{N}$$

Оцінка моделі, заснована лише на її надійності, буде достатньою для нас, якщо класи мають однакове значення; однак це не дозволяє нам брати до уваги різницю у значущості між FP і FN, яка є важливою в багатьох галузях, включаючи медицину. Це досягається використанням вичерпності та точності.

Здатність ідентифікувати певний клас називається повнотою. Ця характеристика в медицині називається чутливістю (sensitivity). Високий рівень чутливості означає, що хворі люди, у яких справді є захворювання, були правильно ідентифіковані.

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

Точність (здатність відрізняти цей клас від інших):

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

Специфіка (specificity), найчастіше застосовується в медичній статистиці. Висока специфічність дозволяє відсіяти людей, які справді не мають цього захворювання.

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

Коефіцієнт кореляції Метьюса (у статистиці відомий як фікоефіцієнт) може застосовуватися як міра якості для бінарної класифікації при високому дисбалансі класів. Приймає значення в діапазоні [–1,1], де 1 – ідеальне передбачення, 0 – випадкове передбачення, –1 – повне розходження між прогнозом і спостереженням.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

На додаток до показників, розглянутих вище, матрицю неточності можна використовувати для обчислення великої кількості інших показників, які є унікальними для відповідних тематичних областей; наприклад, коефіцієнт помилкових відкриттів (FDR) для геноміки.

І повнота, і точність є взаємовиключними поняттями. У будь–який момент часу відкликання може бути збільшено до 1, незважаючи на дуже низьку точність. F–міра, яка є гармонійним середнім вимірювань повноти та точності:

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}}, \alpha \in [0,1]$$

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}, \beta^2 = \frac{(1-\alpha)}{\alpha}, \beta^2 \in [0, \infty)$$

Звичайна залежність повноти і точності показано малюнку 8, але таке буває який завжди, у деяких завданнях необхідно віддавати перевагу повноті ($\beta > 1$) чи точності ($\beta < 1$). В іншому випадку можна використовувати збалансований F–міру ($\beta = 1$):

$$F_1 = \frac{2PR}{P + R},$$

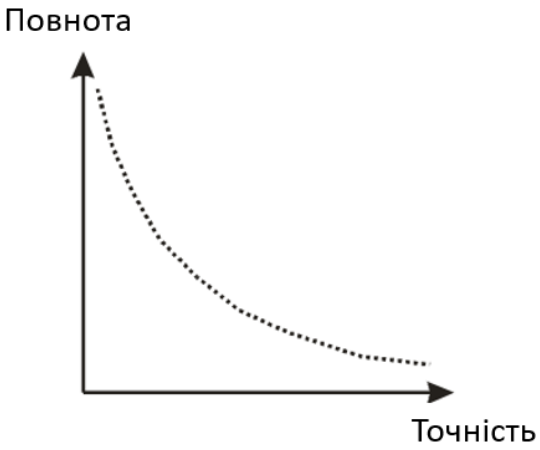



Рисунок 2.4 – Графік залежності повноти та точності

Інший дуже популярний спосіб оцінити якість класифікаційних моделей – це ROC–криві (ROC – робоча характеристика приймача, метод був розроблений для військових радарів у 1941 році, звідси й назва).

У випадку бінарної класифікації ймовірність того, що об’єкти належать до класів, виводиться з самих об’єктів; незважаючи на це, для того, щоб елемент було призначено до класу, потрібно спочатку визначити порогове значення, яке відрізнятиме класи один від одного. Інтуїтивно зрозуміло, що ви повинні вибрати поріг, який дорівнює 0,5, але на практиці це не завжди досяжно, особливо для вибірки, незбалансованої за класами. У цьому конкретному сценарії поріг має бути налаштований таким чином, щоб він потрапляв у певний клас. Крім того, потреби бізнес–процесу, в який вбудовано модель, можуть потенційно вплинути на значення порогу.


Рисунок 2.5 – Аналіз ймовірності схвалення або відмови пропозації

Через це ми можемо використовувати криву ROC для оцінки якості моделі без обмеження певним числом для порогового значення. ROC означає робочу характеристику приймача. Кожна точка на кривій представляє інше значення, яке служить порогом. І FPR (який відображається на горизонтальній осі), і TPR (який відображається на вертикальній осі) зростають, коли порогове значення знижується.

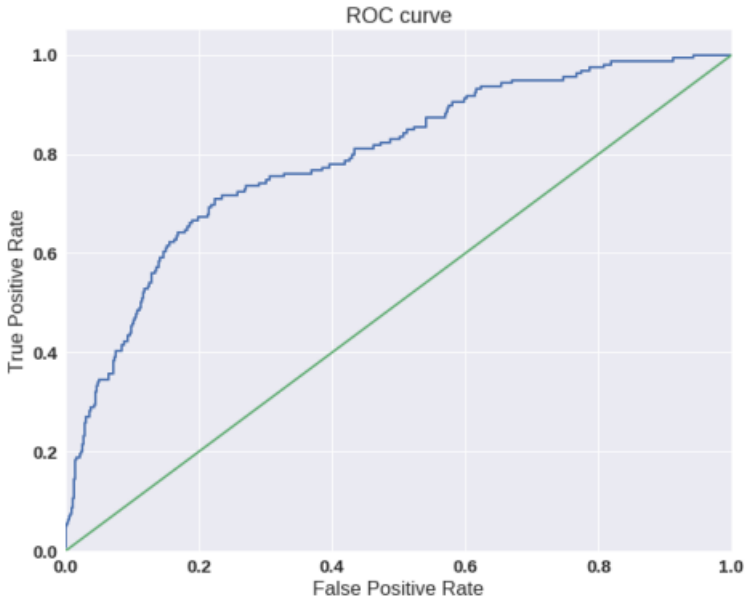



Рисунок 2.6 – ROC графік для оцінки якості моделі

TPR (true positive rate) показує чутливість алгоритму класифікації та дорівнює recall. FPR (false positive rate) відображає частку об'єктів negative класу, передбачених неправильно:

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

Крива ROC була розроблена для використання для бінарної класифікації; однак у ситуації багатокласової класифікації кожен клас має власну криву ROC.

Оскільки може бути важко візуально оцінити багато моделей на кривих ROC, замість цього використовується індикатор площі під кривою (AUC). Ця індикація забезпечує чисельне наближення значення. Площа під кривою робочих характеристик приймача (AUC) порівнянна з імовірністю того, що

класифікатор надасть більшу вагу випадково вибраному позитивному елементу, ніж випадково вибраному негативному об’єкту.

$$A = \int_{-\infty}^{\infty} y(T)x'(T)\,dT = \int_{-\infty}^{\infty} TPR(T)FPR'(T)\,dT =$$

$$= \int_{-\infty}^{\infty} TPR(T)P_0(T)\,dT = \langle TPR \rangle$$

В ідеальному випадку $A$ = 1. Чим вона більша, тим алгоритм краще працює. Якщо $A$ = 0.5 (що на ROC–кривій відображається збігом графіка та діагоналі), то наша модель, по суті, здійснює випадкове вгадування. Якщо $A$ < 0.5, то класифікатор діє з точністю навпаки.

# РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПОБУДУВАННЯ ЛОГІСТИЧНОЇ РЕГРЕСІЇ

Дані, що використовуються у роботі були отримані з відкритого ресурсу. Характеристики обчислюються з оцифрованого зображення аспірату тонкої голки (FNA) утворення молочної залози, дані зібрані у окремій таблиці з 32 колонками та 569 записами. Цільова ознака – діагноз (доброякісна або злоякісна пухлина), розподіл значень: 357 доброякісних значень та 212 злоякісних.

Для кожного клітинного ядра обчислюється десять дійсних характеристик: радіус (середнє значення відстаней від центру до точок на периметрі), текстура (стандартне відхилення значень сірого), периметр, площа, гладкість (локальні зміни довжини радіуса), компактність (периметр^2 / площа – 1,0), увігнутість (вираженість увігнутих ділянок контуру), увігнуті точки (кількість увігнутих ділянок контуру), симетрія, фрактальна розмірність ("апроксимація берегової лінії" – 1).

Для кожного зображення було обчислено середнє значення, стандартну помилку та «найгірше» або найбільше (середнє з трьох найбільших значень), у результаті чого було отримано 30 ознак. Наприклад, поле 3 – середній радіус, поле 13 – радіус SE, поле 23 – найгірший радіус. Усі значення ознак перекодуються чотирма значущими цифрами.

Пакет Python із відкритим кодом під назвою NumPy пропонує швидкі, попередньо скомпільовані функції для популярних математичних і числових операцій. Щоб створити пакети високого рівня, їх об’єднують. Вони пропонують можливості, які можна порівняти з можливостями MatLab. Великі масиви та матриці можна обробляти просто за допомогою NumPy (числовий Python). Завдяки широкому спектру корисних алгоритмів, включаючи мінімізацію, перетворення Фур’є, регресію та інші прикладні математичні підходи, SciPy (Scientific Python) розширює можливості numpy.

Об’єкт масиву є основною функцією numpy. За винятком вимоги, щоб компоненти масиву належали до одного типу даних, наприклад float і int, масиви та списки в Python можна порівнювати. Порівняно зі списками, масиви дозволяють обробляти величезні обсяги даних значно швидше і, головне, набагато ефективніше.

Pandas – це модуль Python, який використовується для статистичного аналізу та обробки даних. Це бібліотека з відкритим вихідним кодом, яка сприяє різноманітній обробці даних і є швидкою та простою у використанні. Вони охоплюють суперечки, статистичний аналіз, злиття та багато інших процесів. У цій статті ми розглянемо використання бібліотеки Pandas для обчислення підсумкової статистики.

Pandas спрощує виконання багатьох трудомістких і повторюваних завдань, пов’язаних із роботою з даними, зокрема:

- Очищення даних
- Заповнення даних
- Нормалізація даних
- Зливається і приєднується
- Візуалізація даних
- Статистичний аналіз
- Перевірка даних
- Завантаження та збереження даних

### 3.1. Реалізація дослідницького аналізу, огляд інструментів для детального аналізу даних

```
# імпортування необхідних бібліотек Python
import numpy as np
import pandas as pd
```

Пакет комп'ютерної мови Python для візуалізації 2D і 3D даних називається Matplotlib. Отримані зображення можна використовувати як ілюстрації в книгах.

Разом із NumPy, SciPy та IPython Matplotlib є універсальною програмою, яка пропонує функції, подібні до MATLAB. Тепер пакет підтримує ряд графічних бібліотек, включаючи PyGTK і wxWindows.

Програмне забезпечення підтримує численні типи графіків і діаграм, зокрема:

- Графіки (англ. line plot)
- Діаграми розсіювання (англ. scatter plot)
- Стовпчасті діаграми (англ. bar chart) та гістограми (англ. histogram)
- Кругові діаграми (англ. pie chart)
- Діаграми стебло–листя (англ. stem plot)
- Контурні графіки (англ. contour plot)
- Поля градієнтів (англ. Quiver)
- Спектральні діаграми (англ. spectrogram)

Пакет Seaborn від Python дозволяє створювати статистичні візуальні ефекти. Він тісно поєднується зі структурами даних Pandas і побудований на Matplotlib. Ви можете перевірити та зрозуміти свої дані за допомогою Seaborn. Потужний і надзвичайно настроюваний, Matplotlib. Теми за замовчуванням у Seaborn допомагають уникнути накладання сюжетів. Matplotlib використовує Numpy і Pandas для побудови кількох типів графіків. Розширена версія Matplotlib, Seaborn, будує графіки за допомогою Matplotlib, Numpy та Pandas.

Plotly – це бібліотека Python для інтерактивної візуалізації даних. Plotly неймовірно потужний у поясненні та дослідженні даних, він надає функцію, якої немає в жодній іншій бібліотеці візуалізації – інтерактивність. Це дозволяє користувачам взаємодіяти з графіками на дисплеї, забезпечуючи кращий досвід оповідання. Збільшення та зменшення масштабу, відображення

балів, панорамування графіків та інші інтерактивні дії, що спрощують аналіз даних.

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls
import plotly.figure_factory as ff
```

Бібліотека машинного навчання scikit–learn на основі мови програмування Python доступна для безкоштовного завантаження. Вона містить низку методів, таких як машина опорних векторів, метод випадкового лісу, алгоритм збільшення градієнта, метод k–середніх і DBSCAN, які призначені для класифікації, регресії та кластерного аналізу даних.

Для високошвидкісної лінійної алгебри та операцій з масивами пакет scikit–learn активно використовує бібліотеку NumPy, яка в основному розроблена на мові програмування Python.

Для підвищення продуктивності деякі ключові алгоритми написані на мові програмування Cython. Обгортка, написана на Cython для бібліотеки LIBSVM, використовується для реалізації машини опорних векторів, тоді як оболонки для бібліотеки LIBLINEAR використовуються для реалізації логістичної регресії та лінійної машини опорних векторів. Слід зазначити, що за таких обставин розширення таких процедур за допомогою мови програмування Python може бути неможливим.

Пакет scikit–learn добре працює з кількома іншими бібліотеками програмування Python, включаючи SciPy, Pandas і Matplotlib для роботи з об’єктами DataFrame, а також Matplotlib і plotly для візуалізації даних.

```
from sklearn.feature_selection import RFE
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import VotingClassifier
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.model_selection import learning_curve, train_test_split
from sklearn.metrics import precision_score, confusion_matrix, recall_score
from sklearn.metrics import roc_curve, accuracy_score, precision_recall_curve
```

На початку проведення аналізу необхідно створити датасет, який використовується у роботі.

```
# Читання датасету для подальшого його аналізу
data = pd.read_csv('../input/data.csv')
```

Видалення нульових значень – це процес очищення даних, який може допомогти підвищити якість аналізу та моделювання. Це дозволяє усунути порожні або неінформативні рядки або стовпці, що може вплинути на точність моделі.

```
check_nulls = pd.DataFrame(len(data['id']) - data.isnull().sum(), columns = ['Count'])

trace = go.Bar(x = check_nulls.index, y = check_nulls['Count'] ,opacity = 0.5, marker=dict(color = 'black'))
layout = dict(title =  "Перевірка на нульові значення у датасету")

fig = dict(data = [trace], layout=layout)
py.iplot(fig)
```

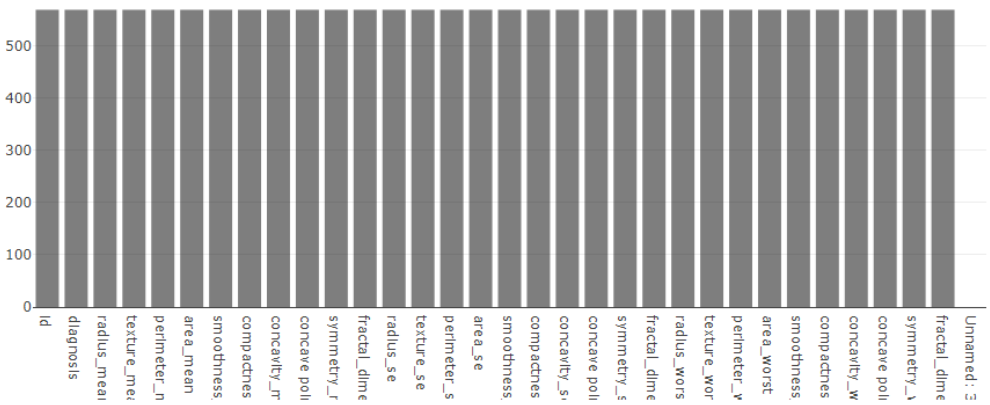



Рисунок 3.1 – Перевірка на нульові значення датасету

Усі змінні повні, лише «Unnamed: 32» повністю нульовий. Можливо, помилка у наборі даних, або це технічне поле, що було видалено при публікації датасету. Подібні змінні мають бути видалені з вибірки, це ми зробимо на наступному кроці.

```
# Видалення непотрібні змінних, що не пройшли перевірку якості, або не мають цінності під час аналізу.
data = data.drop(['Unnamed: 32','id'],axis = 1)

# Переназначуємо ціль M(malignant - злоякісна) = 1, B(benign - доброякісна) = 0.
data.diagnosis.replace(to_replace = dict(M = 1, B = 0), inplace = True)
```

Head() – функція, яка відображає перші n рядки набору даних.

Метод Pandas DataFrame describe () використовується для обчислення деяких статистичних даних, таких як процентиль, середнє значення та стандартне відхилення різних числових значень DataFrame. Він використовується для аналізу як числових, і об'єктних серій, і навіть DataFrame, що має набори стовпців змішаних типів даних.

```
# Head функція для огляду перших значень датасету та
#візуального ознайомлення з вибіркою та її змінними
data.head()
```

| | diagnosis | radius_mean | texture_mean |
|---|---|---|---|
| 0 | 1 | 17.99 | 10.38 |
| 1 | 1 | 20.57 | 17.77 |
| 2 | 1 | 19.69 | 21.25 |
| 3 | 1 | 11.42 | 20.38 |
| 4 | 1 | 20.29 | 14.34 |

```
# describe функція для знаходження
#базових статистичних показників вибірки
data.describe()
```

| | diagnosis | radius_mean | texture_mean |
|---|---|---|---|
| count | 569.000000 | 569.000000 | 569.000000 |
| mean | 0.372583 | 14.127292 | 19.289649 |
| std | 0.483918 | 3.524049 | 4.301036 |
| min | 0.000000 | 6.981000 | 9.710000 |
| 25% | 0.000000 | 11.700000 | 16.170000 |
| 50% | 0.000000 | 13.370000 | 18.840000 |
| 75% | 1.000000 | 15.780000 | 21.800000 |
| max | 1.000000 | 28.110000 | 39.280000 |

Рисунок 3.2 – Огляд перших значень датасету

Рисунок 3.3 – Базові статистичні показники вибірки

Для подальшої роботи алгоритму необхідно розділити датасет на 2 піддатасети, де M – злоякісна пухлина, а B – доброякісна пухлина.

```
# розподіляємо на 2 датасети. M - злоякісна пухлина, B - доброякісна пухлина.
M = data[(data['diagnosis'] == 1)]
B = data[(data['diagnosis'] == 0)]
```

Досліджуємо розподіл датасету за цільовою змінною, що у нашому випадку – діагноз захворювання. Насамперед, проаналізуємо кількісне співвідношення змінних.

```
#------------COUNT-----------------------
trace = go.Bar(x = (len(M), len(B)), y = ['злоякісна', 'доброякісна'], orientation = 'h', opacity = 0.8, marker=dict(
        color=[ '#d63118', '#d9c91c'],
        line=dict(color='#000000',width=1.5)))

layout = dict(title =  'Кількість змінних за діагнозами')

fig = dict(data = [trace], layout=layout)
py.iplot(fig)
```

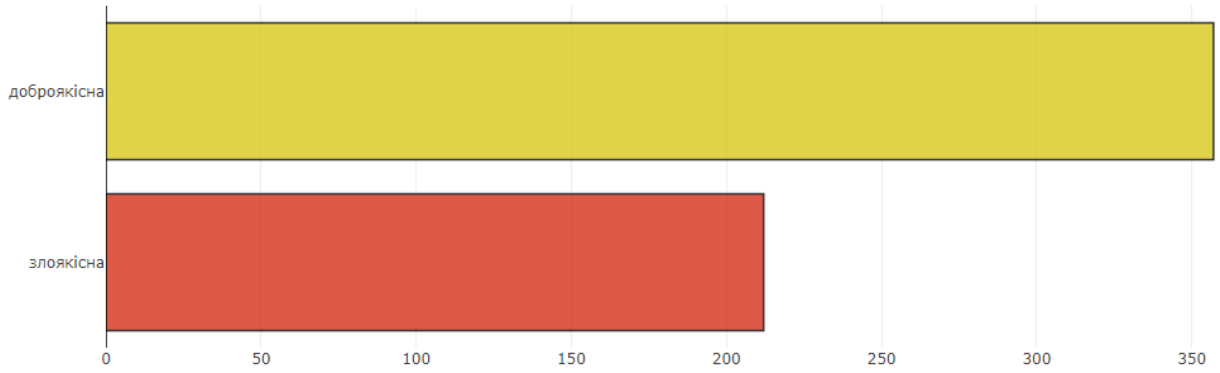



Рисунок 3.4 – Кількість змінних за діагнозами

Оцінимо також відсоткове співвідношення доброякісного та злоякісного показнику. Цей етап початкового аналізу допомагає зрозуміти з якими даними ми працюємо, це особливо важливо при роботі з великим обсягом даних.

```
#------------PERCENTAGE-------------------
trace = go.Pie(labels = ['доброякісна','злоякісна'], values = data['diagnosis'].value_counts(),
               textfont=dict(size=15), opacity = 0.8,
               marker=dict(colors=['#d9c91c', '#d63118'],
                           line=dict(color='#000000', width=1.5)))

layout = dict(title =  'Розподіл змінної діагнозу')

fig = dict(data = [trace], layout=layout)
py.iplot(fig)
```

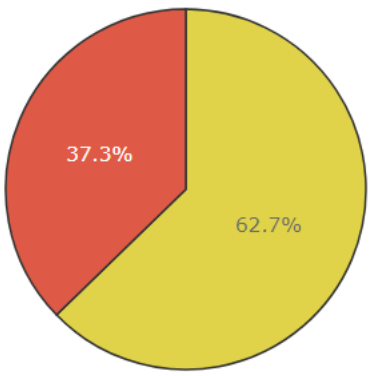



Рисунок 3.5 – Розподіл змінної діагнозу

Функція EDA_plot_builder створює графік, за яким можна оцінити кожну зі змінних датасету.

```
def EDA_plot_builder (feature_name, size_bin):
    M_df = M[feature_name]
    B_df = B[feature_name]
    hist_data = [M_df, B_df]

    group_labels = ['злоякісна', 'доброякісна']
    colors = ['#d63118', '#d9c91c']

    fig = ff.create_distplot(hist_data, group_labels,
                             colors = colors, show_hist = True,
                             bin_size = size_bin, curve_type='kde')
    fig['layout'].update(title = feature_name)
    py.iplot(fig, filename = 'Density plot')
```

Графік розподілу змінної з датафрейму – це візуалізація даних, що показує, як часто кожне значення змінної зустрічається у вибірці. На графіці осі X розташовуються значення змінної, а осі Y – їх частота.

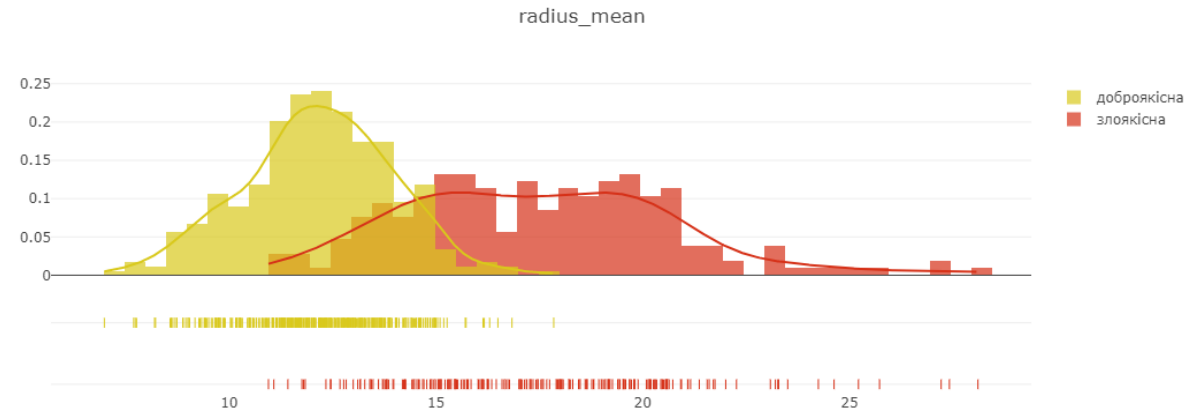



Рисунок 3.6 – Приклад розподілу змінної radius_mean

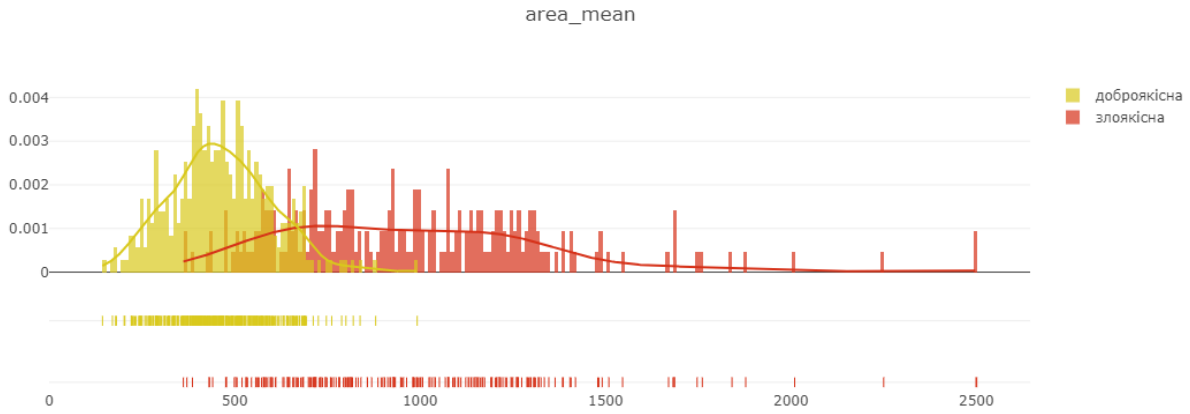


Рисунок 3.7 – Приклад розподілу змінної area_mean

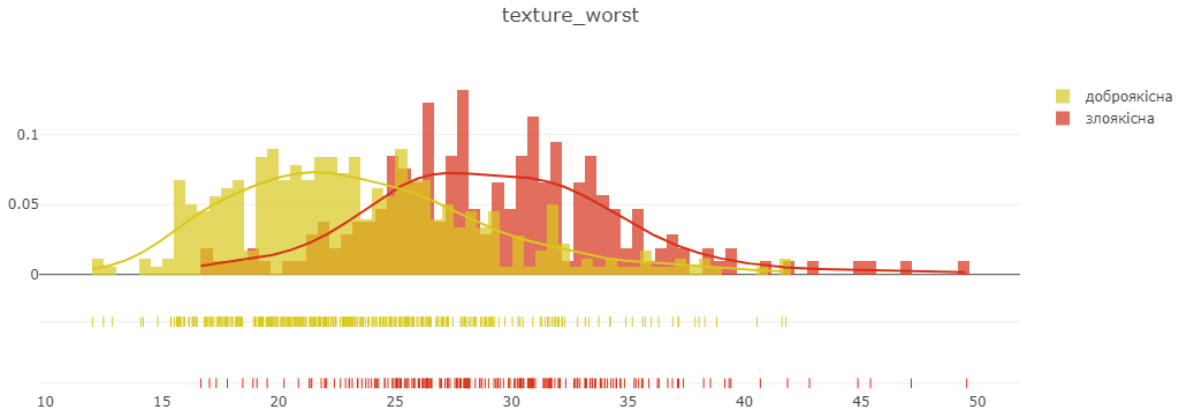



Рисунок 3.8 – Приклад розподілу змінної texture_worst

Графіки розподілу змінних із фреймів даних забезпечують безліч переваг, починаючи від сприяння візуалізації даних і закінчуючи розпізнаванням викидів і аномалій і закінчуючи порівнянням різних розподілів змінних. Можна навіть створити графіки для перевірки статистичних гіпотез або для ілюстрації результатів аналізу даних для кращого розуміння спеціалістів сфери. Зі змінними графіками розподілу розуміння даних стає легшим і більш повним, оскільки можна визначити закономірності та залежності, використовувати дані для створення обґрунтованих висновків.

### 3.2 Перевірка кореляції ознак даних за допомогою кореляційної матриці

```
#Створюємо кореляційну матрицю
correlation = data.corr()
matrix_cols = correlation.columns.tolist()
corr_array  = np.array(correlation)
```

Кореляційна матриця – це таблиця, яка ілюструє кореляцію між парами змінних у наборі даних. У графі кореляційної матриці кожен стовпець і рядок представляють змінну, а на перетині стовпців і рядків знаходиться коефіцієнт кореляції між відповідними змінними. Колір на графіку часто

використовується для вказівки на силу кореляції, причому яскравіший колір означає сильнішу кореляцію, а більш тьмяний колір – слабшу. Цей тип графіка допомагає візуально оцінити силу та напрямок зв’язку між змінними, що може бути корисним для розуміння взаємозв’язків у даних і для вибору найбільш значущих змінних для аналізу.

```
#Створюємо візуалізацію кореляції
trace = go.Heatmap(z = corr_array, x = matrix_cols, y = matrix_cols,
                   xgap = 3, ygap = 3, colorscale='RdBu', colorbar=dict())

layout = go.Layout(dict(title = 'Кореляційна матриця для змінних вибірки',
                        autosize = False,
                        height  = 720,
                        width   = 800,
                        margin  = dict(r = 0 ,l = 210,
                                       t = 25,b = 210),
                        yaxis   = dict(tickfont = dict(size = 9)),
                        xaxis   = dict(tickfont = dict(size = 9)),
                       )
                  )
fig = go.Figure(data = [trace],layout = layout)
py.iplot(fig)
```

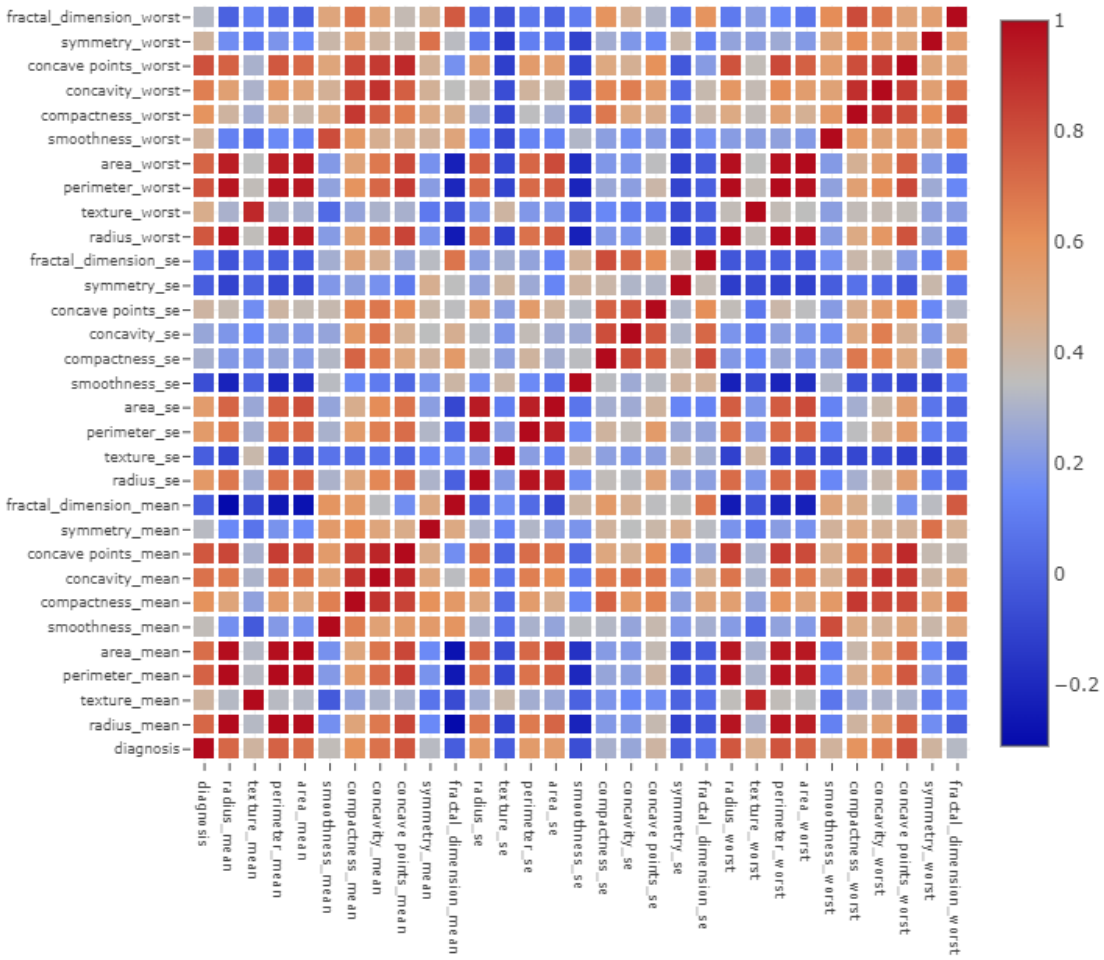



Рисунок 3.9 – Кореляційна матриця для змінних вибірки

Роздивимось більш детально пари змінних та перевіримо їх кореляцію. Кореляційний графік зображує зв’язок між двома відмінними ознаками шляхом нанесення точок на двовимірну площину. Якщо точки близькі до утворення прямої лінії, між ознаками існує висока лінійна кореляція. З іншого боку, якщо точки розподілені далеко одна від одної, не утворюючи певного шаблону, тоді характеристики корелюють у меншій мірі або зовсім не існують.

```
def plot_cor_2_val(val1, val2) :
    trace1 = go.Scatter(
        x = M[val1], y = M[val2],
        name = 'злоякісна', mode = 'markers',
        marker = dict(color = '#d63118', line = dict(width = 1)))

    trace2 = go.Scatter(
        x = B[val1], y = B[val2],
        name = 'доброякісна', mode = 'markers',
        marker = dict(color = '#d9c91c', line = dict(width = 1)))

    layout = dict(title = val1 +" "+"vs"+" "+ val2,
                  yaxis = dict(title = val2,zeroline = False),
                  xaxis = dict(title = val1, zeroline = False))

    plots = [trace1, trace2]

    fig = dict(data = plots, layout=layout)
    py.iplot(fig)
```

Завдяки аналізу за графіком ми можемо виділити позитивно корельовані, некорельовані та негативно корельовані ознаки. Далі наведемо декілька прикладів кожного з варіантів взіємозв'язку ознак. Розглянемо приклади корельованості ознак:

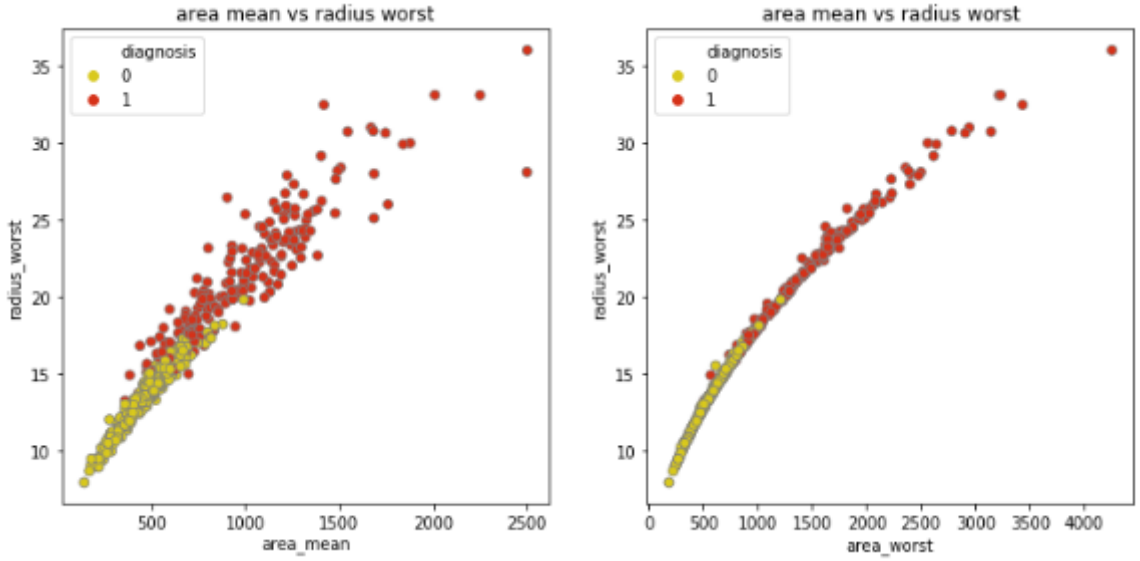



Рисунок 3.10 – Позитивно корельовані ознаки

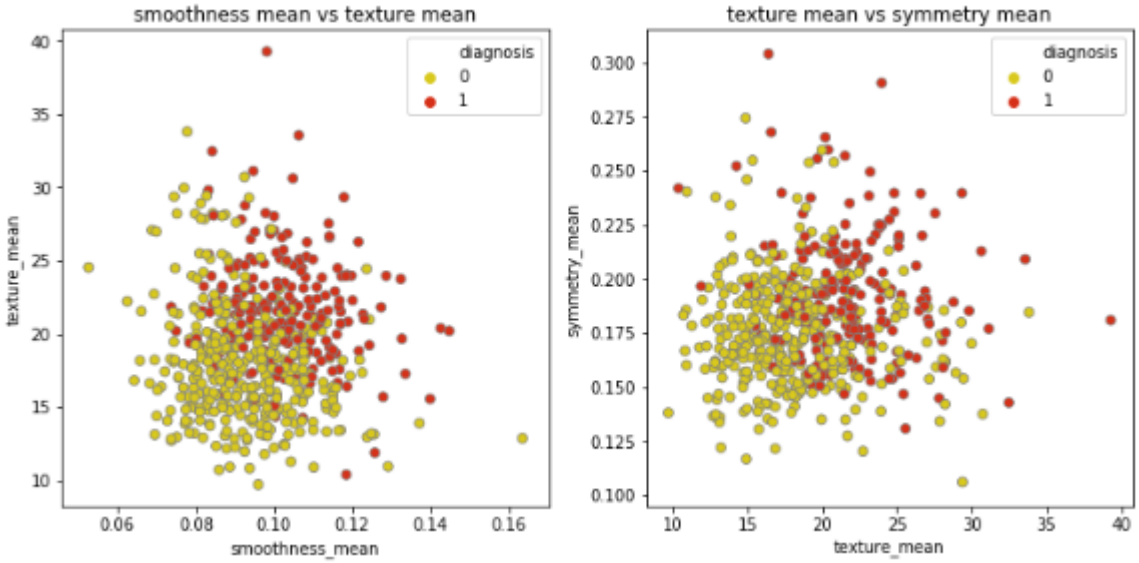



Рисунок 3.11 – Некорельовані ознаки

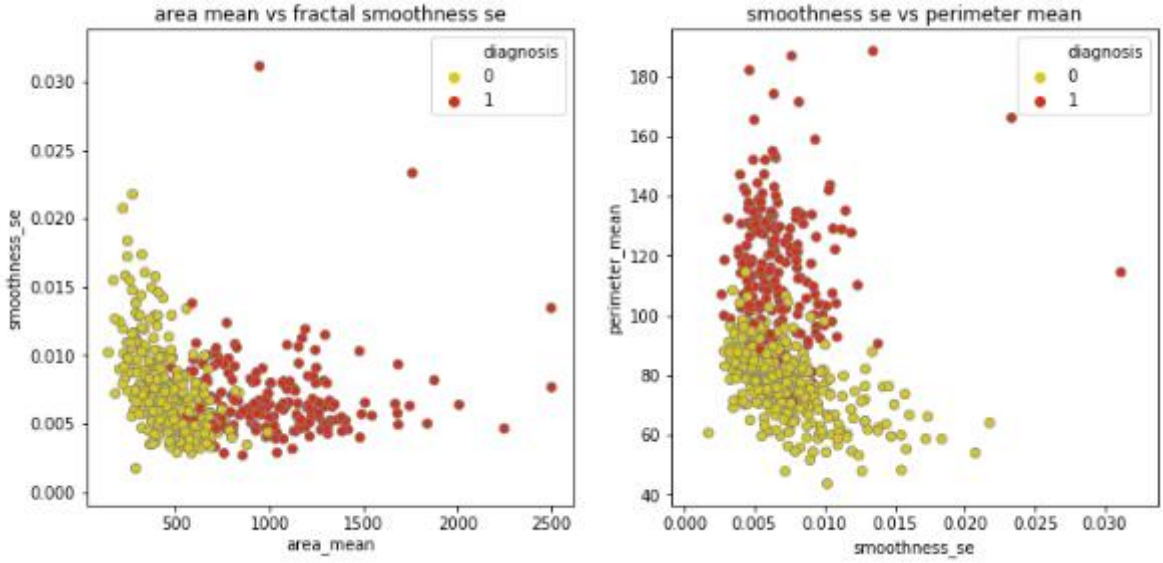



Рисунок 3.12 – Негативно корельовані ознаки

### 3.3 Реалізація методу головних компонент та перевірка зміни точності моделі

PCA (Principal Component Analysis) – це метод, який дозволяє зменшити розмірність даних, зберігаючи при цьому найбільшу кількість інформації. Він ґрунтується на пошуку нових лінійних комбінацій вихідних ознак (головних компонентів), які максимально описують мінливість даних. При цьому перша головна компонента описує найбільшу частку мінливості даних, друга наступну за нею і так далі.

Таким чином, PCA дозволяє побачити приховані закономірності даних і зменшити їх розмірність для більш ефективного аналізу. Роздивимось зміну розмірності до 6, 2 та 3 компонент. Проаналізуємо точність, з якою обрана кількість компонент відтворює дані та розглянемо оптимальні способи зображення для кожного з варіантів.

Роздивимось метод головних компонент з 6 компонентами:

```
#explained_variance
var_pca = pd.DataFrame(pca.explained_variance_ratio_)
var_pca = var_pca.T

#----------SUM AND DROP COMP [7:30]
col_list = list(v for v in chain(pca_std.columns[6:30]))
var_pca['OTHERS_COMP'] = var_pca[col_list].sum(axis=1)
var_pca.drop(var_pca[col_list],axis=1,inplace=True)
var_pca = var_pca.T
```

Формулюємо компоненти та оцінюємо наскільки точно вони описують зміну даних. Компоненти утворюються з лінійних комбінацій початкових ознак, при цьому кожен компонент представляє нову функцію, яка найкраще описує дані. Компоненти формуються таким чином, що перший компонент пояснює максимально можливу кількість дисперсії в даних, другий компонент пояснює максимально можливу дисперсію, що залишилася, і так далі.

```
labels = ['COMP1','COMP2','COMP3','COMP4','COMP5','COMP6', 'COMP7 - 30']
colors = ['red', 'blue', 'yellow', 'green', 'purple', 'orange', 'lightgrey']

trace = go.Pie(labels = labels, values = var_pca[0].values, opacity = 0.8,
               textfont=dict(size=15),
               marker=dict(colors=colors,
                           line=dict(color='#000000', width=2)))

layout = dict(title =  'PCA : компоненти та пояснена дисперсія (6 комп. = 88,8%)')
fig = dict(data = [trace], layout=layout)
py.iplot(fig)
```

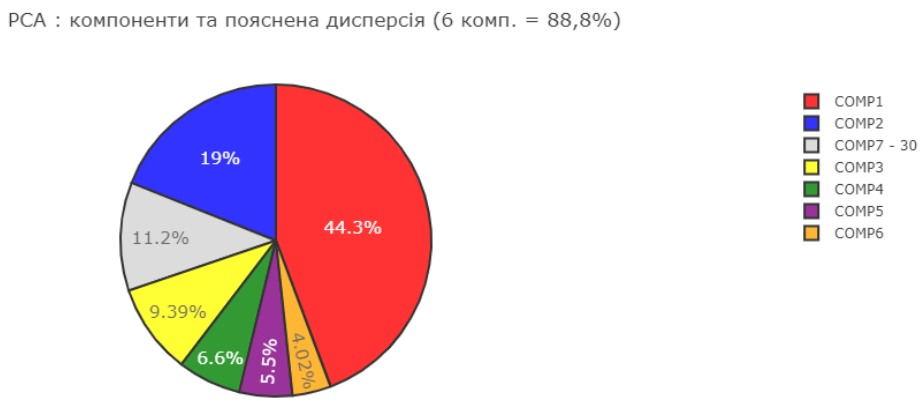



Рисунок 3.13 – PCA з 6 компонентами

Найкращим відтворенням цих даних є наступний графік, через велику кількість компонент. Він наочно відображає яку долю дисперсії пояснює кожна з компонент. Ми бачимо, що перші 6 компонент надають 88,8% точності.

Роздивимось метод головних компонент з 2 компонентами:

```
pca = PCA(n_components = 2)

pca_std = pca.fit(X_std, target_pca).transform(X_std)
pca_std = pd.DataFrame(pca_std,columns = ['COMP1','COMP2'])
pca_std = pca_std.merge(target_pca,left_index = True,right_index = True,how = 'left')
pca_std['diagnosis'] = pca_std['diagnosis'].replace({1:'злоякісна',0:'доброякісна'})
```

```
def pca_scatter(target,color) :
    tracer = go.Scatter(x = pca_std[pca_std['diagnosis'] == target]['COMP1'] ,
                        y = pca_std[pca_std['diagnosis'] == target]['COMP2'],
                        name = target, mode = 'markers',
                        marker = dict(color = color,line = dict(width = 1))
                       )
    return tracer

layout = go.Layout(dict(title = 'Діаграма розсіювання PCA (2 порівняння = 63,3%)',
                        xaxis = dict(gridcolor = 'white',
                                     title = 'COMP1 = 44.3%',
                                     zerolinewidth=1, ticklen=5, gridwidth=2),
                        yaxis = dict(gridcolor = 'white',
                                     title = 'COMP2 = 19.0%',
                                     zerolinewidth=1, ticklen=5, gridwidth=2),
                        height = 800
                       ))

trace1 = pca_scatter('злоякісна','#d63118')
trace2 = pca_scatter('доброякісна','#d9c91c')
plots = [trace2,trace1]
fig = go.Figure(data = plots,layout = layout)
py.iplot(fig)
```

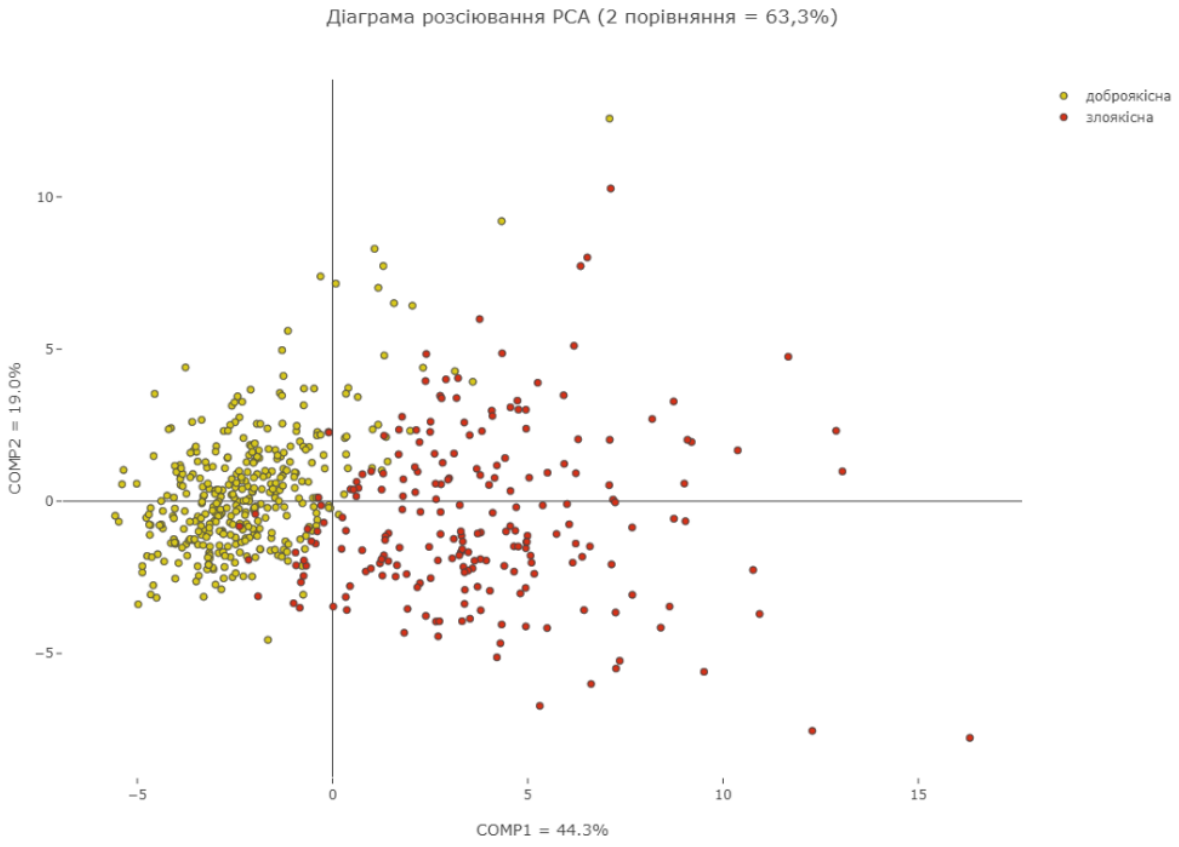



Рисунок 3.14 – PCA з 2 компонентами

У випадках, коли при використанні PCA частка поясненої дисперсії дорівнює 63,3%, можна говорити, що ці компоненти можуть бути корисними для оптимізації продуктивності та швидкості обробки та прорахунку моделі. Однак це значно зменшує точність моделі і призводить до втрати важливих ознак.

Роздивимось метод головних компонент з 3 компонентами:

```
pca = PCA(n_components = 3)
pca_std = pca.fit(X_std, target_pca).transform(X_std)

pca_std = pd.DataFrame(pca_std,columns = ['COMP1','COMP2','COMP3'])
pca_std = pca_std.merge(target_pca, left_index = True, right_index = True,how = 'left')
pca_std['diagnosis'] = pca_std['diagnosis'].replace({1:'злоякісна',0:'доброякісна'})
```

```
M_pca = pca_std[(pca_std['diagnosis'] == 'злоякісна')]
B_pca = pca_std[(pca_std['diagnosis'] == 'доброякісна')]
```

```
trace1 = go.Scatter3d(x = M_pca['COMP1'],
                      y = M_pca['COMP3'],
                      z = M_pca['COMP2'],
                      mode = "markers",
                      name = "злоякісна",
                      marker = dict(size = 4,color = '#d63118',line = dict(width = 1))
                     )
trace2 = go.Scatter3d(x = B_pca['COMP1'],
                      y = B_pca['COMP3'],
                      z = B_pca['COMP2'],
                      name = 'доброякісна',
                      mode = 'markers',
                      marker = dict(size = 4,color= '#d9c91c',line = dict(width = 1))
                     )
```

```
layout = go.Layout(dict(title = 'Діаграма розсіювання PCA (3 порівняння = 72,7%)',
                        scene = dict(camera = dict(up=dict(x= 0 , y=0, z=0),
                                                   center=dict(x=0, y=0, z=0),
                                                   eye=dict(x=1.25, y=1.25, z=1.25)),
                                     xaxis  = dict(title = 'COMP1',
                                                   gridcolor='rgb(255, 255, 255)',
                                                   zerolinecolor='rgb(255, 255, 255)',
                                                   showbackground=True,
                                                   backgroundcolor='rgb(230, 230,230)'),
                                     yaxis  = dict(title = 'COMP3',
                                                   gridcolor='rgb(255, 255, 255)',
                                                   zerolinecolor='rgb(255, 255, 255)',
                                                   showbackground=True,
                                                   backgroundcolor='rgb(230, 230,230)'
                                                  ),
                                     zaxis  = dict(title = 'COMP2',
                                                   gridcolor='rgb(255, 255, 255)',
                                                   zerolinecolor='rgb(255, 255, 255)',
                                                   showbackground=True,
                                                   backgroundcolor='rgb(230, 230,230)'
                                                  )),height = 700))

plots = [trace1,trace2]
fig  = go.Figure(data = plots,layout = layout)
py.iplot(fig)
```

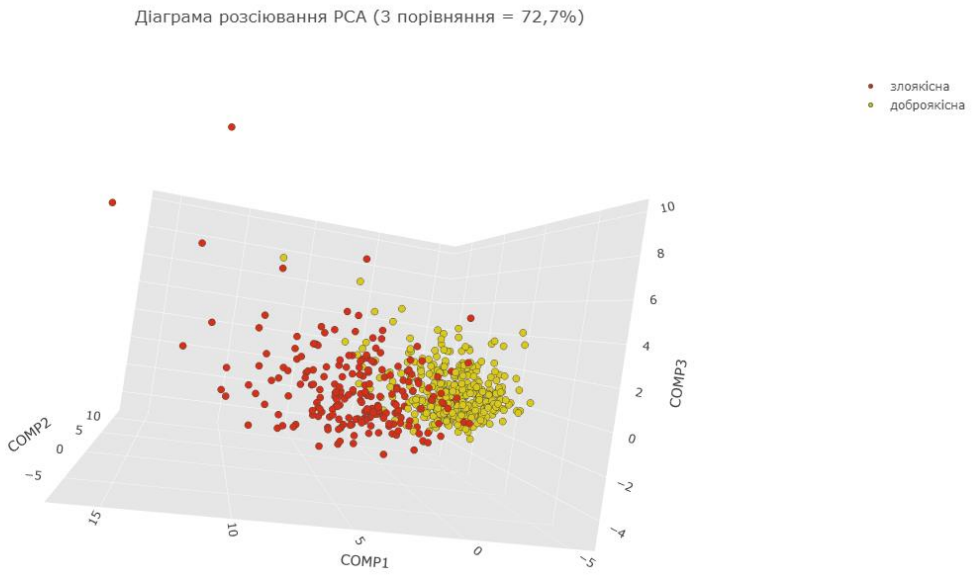



Рисунок 3.15 – PCA з 3 компонентами

### 3.4. Підготовка функцій для перевірки якості моделі

Матриця невідповідностей, також відома як матриця помилок, дозволяє візуалізувати продуктивність алгоритму:

- істинно позитивний (TP – true positive): злоякісна пухлина правильно визначена як злоякісна
- істинно негативний (TN – true negative): доброякісна пухлина правильно визначена як доброякісна
- хибнопозитивний (FP – false positive): доброякісна пухлина неправильно визначена як злоякісна
- хибнонегативний (FN – false negative): злоякісна пухлина неправильно визначена як доброякісна

```
# Функція побудови графіку матриці невідповідностей (матриця помилок)
def plot_confusion_matrix(cm, classes,
                          normalize = False,
                          title = 'Матриця невідповідностей',
                          cmap = plt.cm.Blues) :
    plt.imshow(cm, interpolation = 'nearest', cmap = cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation = 0)
    plt.yticks(tick_marks, classes)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])) :
        plt.text(j, i, cm[i, j],
                 horizontalalignment = 'center',
                 color = 'white' if cm[i, j] > thresh else 'black')

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

Реалізація цієї функції буде наведена надалі при оцінці якості алгоритму класифікації. На основі чотирьох показників (TP, FP, FN, TN) можна оцінити важливі метрики, такі як:

Accuracy – доля правильних відповідей моделі від загальної кількості відповідей.

Precision – це відсоток правильних відповідей моделі всередині класу; цей відсоток являє собою частку елементів, які фактично підпадають під цей клас, на відміну від усіх інших об’єктів, які система виділила йому.

Recall – це відсоток фактичних класифікацій, які є позитивними. Повнота показує відсоток елементів, які справді належать до позитивного класу, демонструючи, наскільки наші прогнози справджуються.

На відміну від accuracy, precision і recall можна використовувати за наявності незбалансованих зразків, оскільки вони не залежать від співвідношення класів. Знайти ідеальний баланс між цими двома критеріями часто є проблемою в реальних умовах. Зрозуміло, що чим вище точність і повнота, тим краще. Однак у реальному житті неможливо досягти максимальної точності і повноти одночасно, тому необхідно знайти певний баланс. Щоб інтегрувати інформацію про точність і повноту нашого алгоритму, ми хотіли б мати конкретну міру. Нам буде простіше вибрати, яку реалізацію використовувати в цій ситуації. Такою метрикою є F–міра.

F–мера є гармонійним середнім між точністю і повнотою. Вона прагне нуля, якщо точність чи повнота прагне нуля.

```
# Функція виводу метрик
def show_metrics():
    tp = cm[1,1]
    fn = cm[1,0]
    fp = cm[0,1]
    tn = cm[0,0]
    print('Accuracy  =     {:.3f}'.format((tp+tn)/(tp+tn+fp+fn)))
    print('Precision =     {:.3f}'.format(tp/(tp+fp)))
    print('Recall    =     {:.3f}'.format(tp/(tp+fn)))
    print('F1_score  =     {:.3f}'.format(2*(((tp/(tp+fp))*(tp/(tp+fn)))/
                                          ((tp/(tp+fp))+(tp/(tp+fn))))))
```

Компроміс між accuracy та recall для різних порогових значень зображено кривою precision–recall. Висока точність корелює з низьким рівнем хибнопозитивних результатів, тоді як високий рівень запам’ятовування корелює з низьким рівнем хибнопозитивних результатів. Велика площа під кривою вказує на високий рівень запам’ятовування та високу точність.

```
# Precision-recall графік
def plot_precision_recall():
    plt.step(recall, precision, color = 'b', alpha = 0.2,
             where = 'post')
    plt.fill_between(recall, precision, step ='post', alpha = 0.2,
                 color = 'b')

    plt.plot(recall, precision, linewidth=2)
    plt.xlim([0.0,1])
    plt.ylim([0.0,1.05])
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.title('Precision Recall Curve')
    plt.show();
```

Крива ROC (крива робочих характеристик приймача) – це графік, що показує продуктивність моделі класифікації за всіх порогів класифікації. Ця крива відображає два параметри: істинний позитивний показник та хибнопозитивний рівень.

```
# ROC curve
def plot_roc():
    plt.plot(fpr, tpr, label = 'ROC curve', linewidth = 2)
    plt.plot([0,1],[0,1], 'k--', linewidth = 2)
   # plt.xlim([0.0,0.001])
   # plt.ylim([0.0,1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curve')
    plt.show();
```

Крива навчання відображає перевірку оцінювача та бали навчання для різних навчальних вибірок. Це техніка, яка дозволяє визначити, наскільки додаткові навчальні дані можуть допомогти та чи є оцінювач більш сприйнятливим до помилок зміщення або дисперсії. Додавання більшої кількості навчальних зразків, швидше за все, збільшить узагальнення.

```
# Learning curve
def plot_learning_curve(estimator, title, X, y, ylim = None, cv = None,
                        n_jobs = 1, train_sizes = np.linspace(.1, 1.0, 5)):
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel('Training examples')
    plt.ylabel('Score')
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv = cv, n_jobs = n_jobs, train_sizes = train_sizes)
    train_scores_mean = np.mean(train_scores, axis = 1)
    train_scores_std = np.std(train_scores, axis = 1)
    test_scores_mean = np.mean(test_scores, axis = 1)
    test_scores_std = np.std(test_scores, axis = 1)
    plt.grid()
    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha = 0.1, color = "g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color = "r",
             label = "Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color = "g",
             label = "Cross-validation score")
    plt.legend(loc = "best")
    return plt
```

Перехресна перевірка – це статистичний метод для оцінювання та порівняння алгоритмів навчання, який передбачає поділ даних на два розділи: один для вивчення або навчання моделі, а інший – для перевірки моделі.

Методологічною помилкою є вивчення параметрів функції прогнозування та її оцінка на тому самому наборі даних. Модель, яка просто повторює мітки зразків, які вона щойно побачила, отримала б хороші оцінки, але не могла б зробити жодних прогнозів щодо даних, які ще не були побачені. Переобладнання – це термін для цієї обставини. Щоб запобігти цій проблемі, під час проведення експерименту з машинним навчанням (під наглядом) частину доступних даних зазвичай резервують як тестовий набір (X_test, y_test).

Перехресна перевірка – це техніка для оцінки прогнозних моделей шляхом поділу вихідної вибірки на навчальний набір для навчання моделі та тестовий набір для її оцінки.

```
# Cross validation metrics
def cross_val_metrics(model) :
    scores = ['accuracy', 'precision', 'recall']
    for sc in scores:
        scores = cross_val_score(model, X, y, cv = 5, scoring = sc)
        print('[%s] : %0.5f (+/- %0.5f)'%(sc, scores.mean(), scores.std()))
```

Визначаємо наші зміні X та Y. Після цього нам необхідно використати метод масштабування StandardScaler. Його принцип роботи полягає у перетворенні значень ознак таким чином, щоб вони мали середнє значення 0 та стандартне відхилення 1.

### 3.5 Побудова логістичної регресії та пошук гіперпараметру

```
# Визначаємо X та Y
y = np.array(data.diagnosis.tolist())
data = data.drop('diagnosis', 1)
X = np.array(data.as_matrix())
```

З якою ціллю ми використовуємо стандартизацію даних? Наприклад, змінна з діапазоном від 0 до 100 переважатиме змінну з діапазоном від 0 до 1. Використання цих змінних без стандартизації фактично надає змінній із більшим діапазоном більшу вагу в аналізі.

```
# Нормалізація
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

Бібліотека Python sklearn пропонує нам функцію StandardScaler() для стандартизації значень даних у стандартному форматі. Відповідно до наведеного вище синтаксису ми спочатку створюємо об’єкт функції StandardScaler(). Крім того, ми використовуємо fit_transform() разом із призначеним об’єктом, щоб трансформувати дані та стандартизувати їх.

У машинному навчанні дуже важливо розділити вибірку на тестові та навчальні групи. Це виконується для оцінки ефективності моделі на незалежних даних і перевірки її узагальненості.

```
# Розподіл вибірки на тестову та тренувальну
random_state = 42
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
                                                    random_state = random_state)
```

GridSearchCV – це метод налаштування гіперпараметрів для пошуку найкращих значень для конкретної моделі. Як уже зазначалося, на продуктивність моделі сильно впливає значення її гіперпараметрів. Зауваживши, що немає способу заздалегідь визначити оптимальні значення для гіперпараметрів, бажано вивчити кожне мислиме значення, перш ніж вирішити, які з них є найкращими. Ми використовуємо GridSearchCV для автоматизації налаштування гіперпараметрів, оскільки виконання цього вручну може зайняти багато часу та ресурсів.

Пакет model_selection Scikit–learn (або SK–learn) має метод під назвою GridSearchCV. Тому дуже важливо згадати, що бібліотека Scikit Learn повинна бути встановлена на комп'ютері. Ця функція допомагає циклічно переглядати попередньо визначені гіперпараметри та адаптувати ваш оцінювач (модель) до вашого навчального набору. Отже, зрештою, ми можемо вибрати найкращі параметри з перерахованих гіперпараметрів.

```
# Знайти найкращі гіперпараметри (точність)
log_clf = LogisticRegression(random_state = random_state)
param_grid = {
             'penalty' : ['l2','l1'],
             'C' : [0.001, 0.01, 0.1, 1, 10, 100, 1000]
             }
```

```
CV_log_clf = GridSearchCV(estimator = log_clf, param_grid = param_grid , scoring = 'accuracy', verbose = 1, n_jobs = -1)
CV_log_clf.fit(X_train, y_train)

best_parameters = CV_log_clf.best_params_
print('Оптимальними параметрами для використання даної моделі є:', best_parameters)
```

Як зазначалося раніше, ми надаємо функції GridSearchCV попередньо визначені значення для гіперпараметрів. Для цього ми визначаємо словник, у якому перераховано кожне можливе значення для заданого гіперпараметра. Ось його ілюстрація.

```
{ 'C': [0.1, 1, 10, 100, 1000],
  'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
  'kernel': ['rbf','linear','sigmoid']  }
```

Тут деякими гіперпараметрами моделі SVM є C, гамма та ядра. Важливо, що для інших гіперпараметрів використовуватимуться значення за замовчуванням.

Технологія перехресної перевірки використовується GridSearchCV для оцінки моделі для кожної комбінації змінних, наданих у словнику. У результаті, після використання цієї функції, ми можемо визначити точність і втрати для кожного набору гіперпараметрів і вибрати комбінацію, яка пропонує найбільшу продуктивність.

Отримані результати пошуку оптимальних гіперпараметрів моделі:

```
Fitting 3 folds for each of 14 candidates, totalling 42 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
Оптимальними параметрами для використання даної моделі є: {'C': 0.1, 'penalty': 'l2'}
[Parallel(n_jobs=-1)]: Done  42 out of  42 | elapsed:    0.4s finished
```

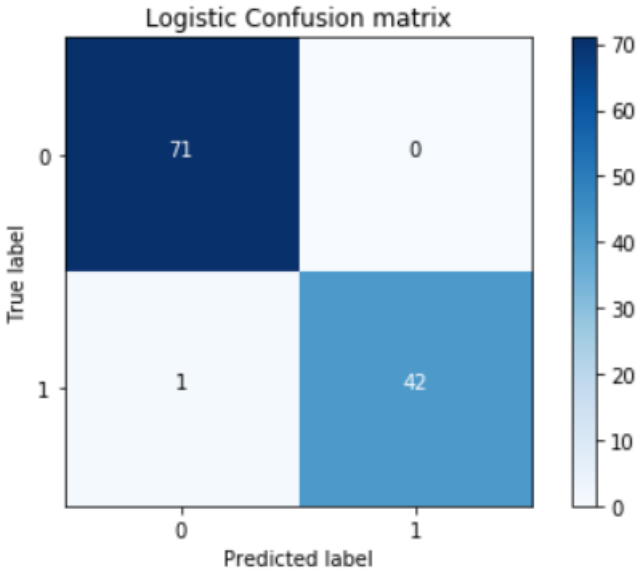
Будуємо логістичну регресію зі знайденими гіперпараметрами.

```
#Log with best hyperparameters
CV_log_clf = LogisticRegression(C = best_parameters['C'],
                                penalty = best_parameters['penalty'],
                                random_state = random_state)

CV_log_clf.fit(X_train, y_train)
y_pred = CV_log_clf.predict(X_test)
y_score = CV_log_clf.decision_function(X_test)
```

Виводимо матрицю помилок та основні метрики якості побудованої моделі класифікації для перевірки точності. Ми використовуємо формули, що були створені на попередніх кроках.

```
# Confusion maxtrix & metrics
cm = confusion_matrix(y_test, y_pred)
class_names = [0,1]
plt.figure()
plot_confusion_matrix(cm,
                      classes=class_names,
                      title='Logistic Confusion matrix')
plt.savefig('6')
plt.show()

show_metrics()
```

Результат роботи логістичної регресії з обраними гіперпараметрами:




```
Accuracy  =     0.991
Precision =     1.000
Recall    =     0.977
F1_score  =     0.988
```

Рисунок 3.16 – Матриця помилок логістичної регресії

Модель працює адекватно на тренувальній вибірці. Лише один випадок був обраний невірно, що вплинуло на значення показника Recall. Але жоден показник, що э негативним не був обраний позитивним, тому метрика якості Precision дала 100% якості. Перевіримо точність модель за ROC графіком.

```
# ROC curve
fpr, tpr, t = roc_curve(y_test, y_score)
plot_roc()
```

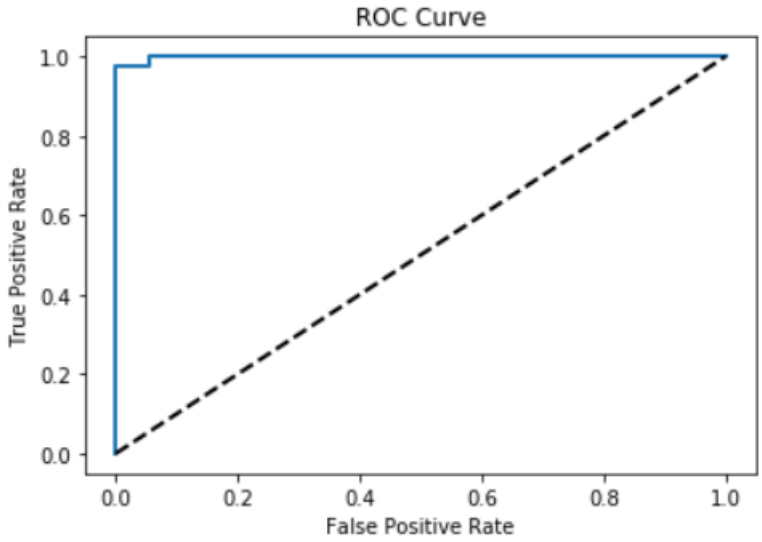



Рисунок 3.17 – ROC графік побудованої логістичної регресії

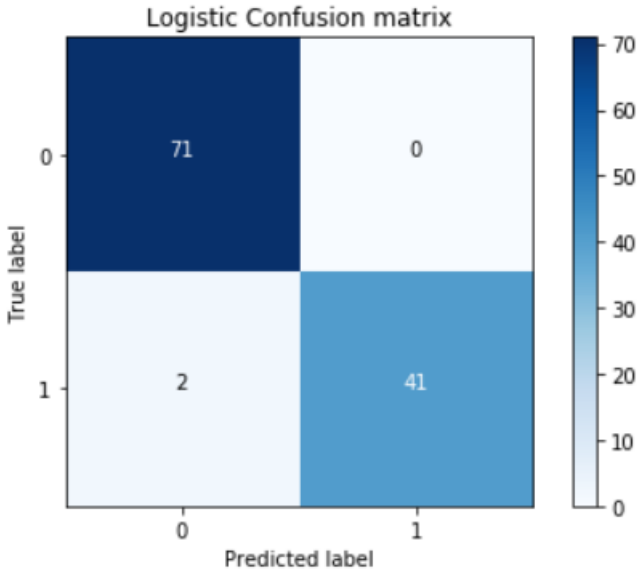
### 3.6 Побудова модифікованої версії логістичної регресії з використанням рекурсивного виключення ознак. Оцінка якості та порівняння з оригінальною версією моделі

Розглянемо спосіб покращення якості моделі з використанням рекурсивного виключення ознак (Recursive Feature Elimination, RFE). Цей метод використовується для пошуку найбільш важливих ознак у наборі даних за допомогою ітеративного процесу. Таким чином ми можемо видалити найслабшу функцію, або функції. Функції ранжуються за показником feature_importances_.

```
#Logistic regression with RFE
log_clf = LogisticRegression(C = best_parameters['C'],
                             penalty = best_parameters['penalty'],
                             random_state = random_state)

selector = RFE(log_clf)
selector = selector.fit(X_train, y_train)

y_pred = selector.predict(X_test)
y_score = selector.predict_proba(X_test)[:,1]
```

Оцінимо якість оптимізованої моделі, використовуючи аналогічні функції.

```
# Confusion maxtrix & metrics
cm = confusion_matrix(y_test, y_pred)
class_names = [0,1]
plt.figure()
plot_confusion_matrix(cm,
                      classes=class_names,
                      title='Logistic Confusion matrix')
plt.show()

show_metrics()

# ROC curve
fpr, tpr, t = roc_curve(y_test, y_score)
plot_roc()
```




```
Accuracy  =     0.982
Precision =     1.000
Recall    =     0.953
F1_score  =     0.976
```

Рисунок 3.18 – Матриця помилок модифікованої логістичної регресії

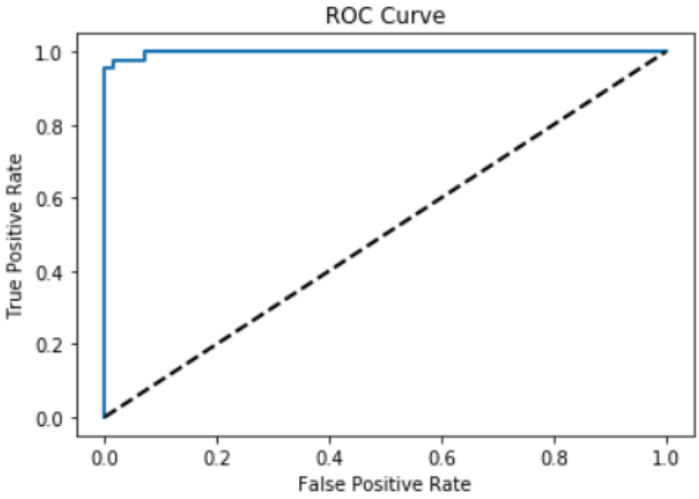



Рисунок 3.19 – ROC графік модифікованої логістичної регресії

Через зменшення розмірності датасету, оптимізована модель гірше показую себе на тренувальній вибірці, проте результати все одно відмінні. Показник Precision залишається на рівні 100%, але вже два насправді позитивні значення були спрогнозовані як негативні. Рекурсивне зменшення ознак допомагає при роботі з великим обсягом даних. У дипломній роботі розглядається тренувальний датасет, тому цей метод не є необхідним. Проте, якщо побудовану модель використовувати на великому обсягу даних, з ще більшою кількістю ознак, їх зменшення буде необхідною мірою для швидкої роботи алгоритму та видалення неінформативних, корельованих або шумових ознак.

Перевіримо які ознаки були оцінені як важливі, а які були видалені при оптимізації моделі:

```
# support and ranking RFE
print(selector.support_)
print(selector.ranking_)
```

```
[ True  True  True  True False False False  True False False  True False
 False  True False False False False False False  True  True  True  True
  True False  True  True  True False]
[ 1  1  1  1  8 16  4  1 15  9  1 14  2  1 11  3 13 12  6  5  1  1  1  1
  1  7  1  1  1 10]
```

Порівняємо роботу двох моделей за графіком навчання поетапно, щоб проаналізувати їх роботу.

```
#Графік навчання для логістичної моделі з найкращими гіперпараметрами
plot_learning_curve(CV_log_clf, 'Графік навчання для логістичної моделі', X, y, (0.85,1.05), 10)
plt.savefig('7')
plt.show()
```

```
#Графік навчання для логістичної моделі з RFE
plot_learning_curve(selector, 'Графік навчання для логістичної моделі з RFE', X, y, (0.85,1.05), 10)
plt.show()
```

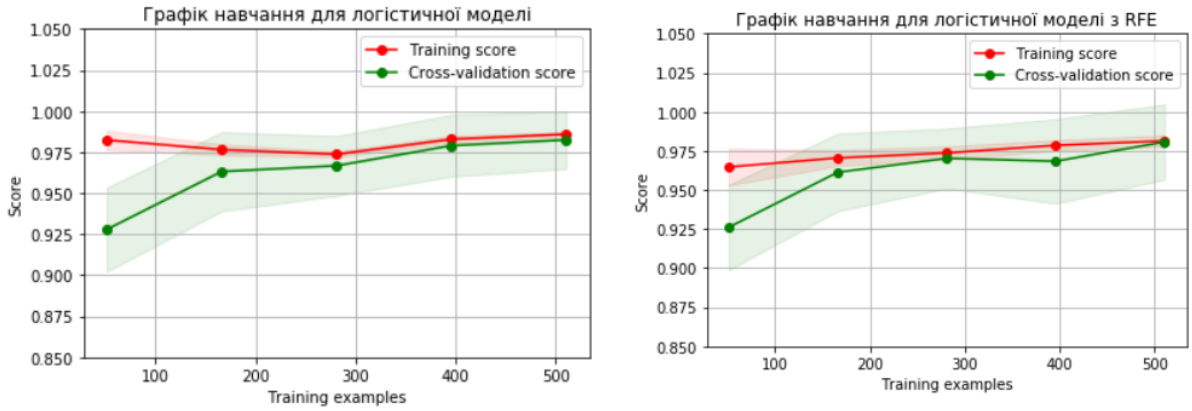



Рисунок 3.20 – Порівняння побудованої логістичної регресії та її модифікованої версії з використанням графіку навчання

Та наведемо оцінку точності кросс валідації:

```
# Кросс валідація для логістичної регресії
cross_log = cross_val_metrics(CV_log_clf)
```

```
[accuracy] : 0.98242 (+/- 0.00560)
[precision] : 0.99036 (+/- 0.01181)
[recall] : 0.96235 (+/- 0.01131)
```

```
# Кросс валідація для логістичної регресії з RFE
cross_selector = cross_val_metrics(selector)
```

```
[accuracy] : 0.97367 (+/- 0.00778)
[precision] : 0.98094 (+/- 0.01754)
[recall] : 0.94817 (+/- 0.01753)
```

Маючи лише 15 ознак і 5 етапів, ми отримали точність 97,4 зі стандартним відхиленням 0,78. На нашому датасету звичайна логістична регресія, без RFE показує кращі результати.

# ВИСНОВКИ

В нашій роботі був розглянутий та проаналізований метод класифікації ракової пухлини з використанням логістичної регресії та пошуку гіперпараметру. Ми провели детальний дослідницький аналіз та покроково розібрали можливості початкового аналізу та підготовки даних. Ознаки вибірки були перевірені на кореляцію.

Ми розробили алгоритм зменшення розмірності даних з використанням методу головних компонент до 2, 3 та 6 компонент. Проведена перевірка впливу на точності та повноту даних після роботи алгоритму та надана оцінка цього впливу.

Після підготовки вибірки до аналізу, розподіленню та тренувальну та тестову підвибірки, пошуку гіперпараметру була побудована модель логістичної регресії. На її основі був створений модифікований алгоритм логістичної регресії з використанням рекурсивного виключення ознак (RFE). Цей процес є необхідний при роботі з більшим обсягом даних.

Були створені функція метрик (матриця помилок, кросс–валідація) та відповідні графіки (ROC графік, графік навчання) для перевірки та порівняння якості двох моделей.

У роботі було розроблено відповідне програмне забезпечення, під час роботи над яким було використано відповідні пакети прикладного програмного забезпечення – Matplotlib, Seaborn, Plotly, Numpy, scikit–learn, Pandas.

У результаті перевірки було виявлено той факт, що модифікована логістична регресія з використанням рекурсивного виключення ознак показує точність 97,4% зі стандартним відхиленням 0,78. Цю модель можна оцінити як досить точну для використання, стандартна логістична регресія показала трохи кращу точність у 98,2% зі стандартних відхиленням 0,05. Цей результат природній на малій вибірці даних, що розглядався у кваліфікаційній роботі.

Але при збільшені датасету, використання оптимізації розмірності даних буде необхідним кроком для швидкої роботи алгоритму.

Отже, можна стверджувати, що в результаті проведеної роботи мету було досягнуто, поставлені завдання були виконані. Запропонована модель логістичної регресії для прогнозування раку знайти своє широке застосування у медицині та підготовці більш детального лікування пацієнта на базу проведених аналізів.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. А.В. Кугаевских, Д.И. Муромцев, О.В. Кирсанова. Классические методы машинного обучения, 2022 – 13–34 с.

2. Валитова, Л. А. Связь процесса генерирования данных и результирующего распределения социально–экономического показателя / Л.А. Валитова. – М.: Проспект, 2016. – 224 с.

3. Ноэль, Элизабет Массовые опросы: Введение в теорию демоскопии / Элизабет Ноэль. – М.: Прогресс, 1978. – 384 с.

4. Бергер, А. Б. Microsoft SQL Server 2005 Analysis Services. OLAP и многомерный анализ данных / А.Б. Бергер. – М.: БХВ–Петербург, 2007. – 659 с.

5. Наследов, Андрей IBM SPSS Statistics 20 и AMOS. Профессиональный статистический анализ данных / Андрей Наследов. – М.: Питер, 2013. – 416 с.

6. Риордан, Дж. Введение в комбинаторный анализ / Дж. Риордан. – М.: 1997. – 990 с.

7. Alan. Agresti: Categorical Data Analysis. Wiley–Interscience, Nowy Jork, 2002 – 105–113 с.

8. William H. Green: Econometric Analysis, fifth edition. Prentice Hall, 2003 – 56–73 с.

9. Ильин В.А., Позняк Э.Г. Основы математического анализа: в 2 ч. – М.: Наука. Физматлит, 1998

10. Павлов Ю.Н. Теория информации для бакалавров. –М.: Издательство МГТУ им. Н.Э. Баумана, 2016

11. Флах П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных. – М.: ДМК Пресс, 2015

12. Han J. et al. Mining Frequent Patterns without Candidate Generation: A Frequent–Pattern Tree Approach // Data Mining and Knowledge Discovery. 2004. Vol. 8, № 1. P. 53–87.

13. Ester M. et al. A Density–Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise // Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. 1996. P. 226–231.

14. Bezdek J.C. Pattern recognition with fuzzy objective function algorithms. New York: Plenum Press, 1981. 256 p.

15. Cross–validation: evaluating estimator performance [Электронный ресурс]. – Режим доступа: https://scikitlearn.org/stable/modules/cross_validation.html

16. SVM [Электронный ресурс]. – Режим доступа: https://staesthetic.files.wordpress.com/2014/02/svm.png?w=1060

17. Comparing different clustering algorithms on toy datasets [Электронный ресурс]. – Режим доступа: https://scikitlearn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

18. Classifier comparison [Электронный ресурс]. – Режим доступа: https://scikitlearn.org/stable/auto_examples/classification/plot_classifier_comparison.html

19. .Friedman J.H. Stochastic gradient boosting // Computational Statistics & Data Analysis. 2002. Vol. 38, № 4. P. 367–378.

20. Rabiner L.R. A tutorial on hidden Markov models and selected applications in speech recognition // Proc. IEEE. 1989. Vol. 77, № 2. P. 257–286.