МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"



**КУРСОВА РОБОТА**

з дисципліни «Машинне навчання»

на тему: **ЗАСТОСУВАННЯ АЛГОРИТМІВ КЛАСТЕРИЗАЦІЇ У ЗАДАЧАХ СЕГМЕНТАЦІЇ ПОКУПЦІВ**

Студента 314 групи спеціальності 122 “Комп’ютерні науки”

Печьонкін Максим Дмитрович

Керівник:

к. е. н., доц. Бойко Н.І.

Кількість балів:\_\_\_\_\_\_\_\_\_\_\_ Оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

**Львів – 2023**

**ЗМІСТ**

ВСТУП………………………………………………………………………….3

РОЗДІЛ 1. Аналіз літературних джерел………………………………………7

РОЗДІЛ 2. Аналіз матеріалів та методів…………………………………….11

2.1 Опис набору даних……………………………………………………….11

2.2 Опис та обгрунтування вибраних моделей……………………….……..26

2.3 Опис методів обробки NaN значень…………………………..................37

2.4 Визначення похибки ……………………………………..……………….39

РОЗДІЛ 3. ЕКСПЕРИМЕНТИ……………………………………………….40

3.1 Опис бібліотеки sklearn …………………………………………………. 41

3.2 Реалізація методів імпутації і методів машинного навчання з датасетом Heart Attack Dataset …………………………………………………………..41

3.3 Реалізація методів імпутації і методів машинного навчання з датасетом Covid-19 Dataset ……………………………………………………………..46

3.4 Опис методів імпутації ………………………………………………… 51

РОЗДІЛ 4. ОБГОВОРЕННЯ РЕЗУЛЬТАТІВ ЕКСПЕРИМЕНТІВ…………56

4.1 Опис і порівняння результатів використовуючи Heart Attack Dataset ..56

4.2 Опис і порівняння результатів використовуючи Covid-19 Dataset …….61

4.3 Опис і порівняння результатів використовуючи Diabetes Health Indicators Dataset ………………………………………………………………………...66

ВИСНОВКИ…………………………………………………………………..72

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ…………………………………….73

ДОДАТКИ……………………………………………………………………..74

**ВСТУП**

Вивчення інформаційних потоків та їхньої якості набуває великого значення для бізнесу, науки та суспільства. Розвиток нових технологій та збільшення обсягів даних вимагає вдосконалення методів їх обробки. Одна з найважливіших проблем в аналізі даних є робота з відсутніми значеннями та невизначеністю, особливо у великих наборах даних.

Ця проблема є важливою у багатьох сферах, таких як: бізнес, наукові дослідження, фінанси, медицина.

**Актуальність дослідження роботи**

1. **Збільшення обсягів даних:** у світі зростає обсяг інформації дуже швидко, і щоб отримати точні і надійні результати необхідно розробляти ефективні методи обробки даних.
2. **Підвищення вимог до якості даних:** точність даних має важливе значення для наукових досліджень. Щоб отримати якісні дані, необхідно усунути невизначеності та пропущені значення.
3. **Інтеграція даних з різних джерел:** організації використовують дані з різних джерел і форматів. Необхідно враховувати різні підходи до обробки пропущених значень та невизначеності для ефективної обробки даних.
4. **Необхідність прийняття обґрунтованих рішень:** важливо приймати обґрунтовані рішення, які базуються на точних і надійних даних.
5. **Розвиток штучного інтелекту та машинного навчання:** через швидкий розвиток штучного інтелекту та машинного навчання, вивчення підходів до обробки даних стає важливою передумовою для розробки ефективних алгоритмів та моделей.

Отже, вивчення та вдосконалення способів обробки даних, особливо в роботі з відсутніми даними та невизначеністю, має велике значення в сучасному світі. Для зростаючих обсягів інформації підвищені вимоги до якості даних, а також потребу в об'єднанні даних з різних місць.

Розроблення ефективних методів обробки даних не тільки забезпечує точність і надійність даних, але й сприяє прийняттю впевнених рішень у різних галузях і загальному прогресу суспільства та бізнесу.

**Метою даної курсової роботи** є аналіз і порівняння різних сучасних підходів до обробки пропущених значень та визначення невизначеності у великих наборах даних.

Щоб досягти цієї мети, поділимо завдання на такі основні завдання:

1. Аналіз та порівняння сучасних існуючих методів для обробки пропущених значень.
2. Розробка і вдосконалення алгоритмів, які забезпечують велику швидкість і точність для обробки пропущених значень та визначення невизначеності у великих наборах даних.
3. Тестування та оптимізація розробленої програми.
4. Порівняння та аналізування отриманих результатів з існуючими методами. Виявлення можливостей застосування отриманих результатів у реальних умовах.

**Об’єктом дослідження** є порівняння іпокращення методів, які використовуються для обробки пропущених значень, визначення їх ефективності і можливостей у застосуванні у практичних ситуаціях.

**Предметом дослідження** є методи та алгоритми, які використовуються для обробки пропущених значень.

Використовуються такі **методи дослідження**:

* **Аналіз літературних джерел:** перегляд уже наявних наукових робіт та досліджень, які розглядають методи обробки пропущених даних та визначення невизначеності у великих обсягах інформації.
* **Емпіричні дослідження:** використання фактичних даних для порівняння різних методів обробки пропущених даних та оцінки їх ефективності в реальних умовах.
* **Статистичний аналіз:** використання статистичних методів для порівняння результатів різних методів обробки пропущених значень та визначення невизначеності великих наборів даних.
* **Моделювання:** створення математичних моделей для вивчення впливу різних методів обробки даних на точність та достовірність результатів.

**Практичне значення досліджен**ня різних підходів до обробки пропущених значень та визначення невизначеності у великих наборах даних полягає в наступному:

* **Покращення якості даних:** розробка ефективних методів роботи з відсутніми даними та невизначеністю може допомогти підвищити якість даних, що використовуються в наукових дослідженнях, бізнес-аналітиці, фінансовому аналізі та інших сферах.
* **Зниження ризику помилок:** використання найефективніших методів дозволить уникнути помилок у аналізі даних і ухилення від неправильних рішень на основі неповних або неточних даних.
* **Підвищення ефективності аналізу:** оптимальний вибір методів обробки даних підвищить ефективність аналітичних процесів, зменшить час, необхідний для обробки даних, та покращить точність результатів.
* **Підтримка прийняття раціональних рішень:** надійні та точні дані є ключовими для прийняття раціональних та обґрунтованих управлінських, наукових та стратегічних рішень, що сприяє успішному функціонуванню організацій та проектів.
* **Застосування у реальних умовах:** результати дослідження можуть бути використані у практичних задачах обробки даних у різних сферах діяльності, що сприятиме підвищенню ефективності та конкурентоспроможності.

**Апробація з дослідження** з обробки відсутніх значень та визначення невизначеності, у великих наборах даних містить: тестування розробленого методу на реальних даних.

Робота буде складатися з наступних розділів:

● Вступ

● Загальний розділ

● Аналіз матеріалів та методів

● Експерименти

● Результати експериментів дослідження

● Висновки

1. **ЗАГАЛЬНИЙ РОЗДІЛ**

У розділі буде проаналізовано наступні літературні джерела: аналіз наукових статей, що стосуються теми курсової роботи.

* 1. **Аналіз літературних джерел**

Список літературних джерел, які використовувались в роботі.

1. *Little, Roderick JA, and Donald B. Rubin. "Statistical analysis with missing data." John Wiley & Sons, 2019.* – фокусується на проблемі відсутніх даних та способах її вирішення, а також детально описує методи статистичного аналізу.
2. *Gelman, Andrew, et al. "Bayesian data analysis." CRC Press, 2021*. –розглядається байєсівський підхід до аналізу даних, який є корисним для визначення невизначеності в даних.
3. *Rubin, Donald B. "Multiple imputation for nonresponse in surveys." John Wiley & Sons, 2004*. – присвячена методам множинних підстановок, які зазвичай використовуються в соціології та інших дослідженнях для роботи з пропущеними даними. Книга містить практичні поради та методологічні підходи для роботи з пропущеними значеннями.
4. *Van Buuren, Stef. "Flexible imputation of missing data." CRC Press, 2012*. – описує гнучкі методи відновлення відсутніх даних, які дозволяють ефективно уникнути втрати інформації під час аналізу. Це важливе джерело інформації для дослідників, зацікавлених у розробці методів обробки неповних даних.
5. *Carpenter, James R., and Michael G. Kenward. "Missing data in clinical trials: a practical guide." CRC Press, 2020*. – фокусується на аналізі пропущених даних у клінічних дослідженнях і надає практичні поради щодо їх обробки. Це джерело є корисним для фахівців у медичній галузі, які стикаються з проблемою відсутніх даних у дослідженнях.
6. *Enders, Craig K. "Applied missing data analysis." Guilford Press, 2010.* – зосереджується на аналізі відсутніх даних у клінічних випробуваннях і надає практичні поради, як працювати з відсутніми даними.
7. *Graham, J. W. (2012). Missing data: Analysis and design. Springer.* – містить огляд того, як аналізувати та планувати дослідження з пропущеними даними. Він допомагає зрозуміти принципи роботи з відсутніми даними та їхній вплив на аналіз результатів дослідження.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Літератур-не джерело № | Методи | Тип завдань | Переваги | Недоліки |
| 1 | Множинні підстановки | Соціологічні дослідження, аналіз опитувань | Ефективне врахування пропущених даних | Вимагається вибір правильного моделювання |
| 2 | Байєсівський підхід | Дослідження з великою кількістю параметрів | Враховує невизначе-ність і враховує апріорні знання | Вимагає досить глибоких знань у байєсівській статистиці |
| 3 | Множинні підстановки | Соціологічні дослідження, аналіз опитувань | Практичні поради і методологічні підходи | Необхідний великий набір даних |
| 4 | Гнучкі методи відновлення | Аналіз даних з різноманітними типами пропущених даних | Ефективно уникає втрати інформації під час аналізу | Є обчислюва-льно складними і зазвичай вимагають більшої обробки даних |
| 5 | Аналіз пропущених даних у клінічних дослідженнях | Клінічні дослідження | Практичні поради щодо обробки пропущених даних | Обмежена область застосування |
| 6 | Аналіз пропущених даних у клінічних випробуваннях | Клінічні дослідження | Надає практичні поради для роботи з відсутніми даними | Обмежена область застосування |
| 7 | Аналіз та планування досліджень з пропущеними даними | Дослідження з пропущеними даними | Допомагає зрозуміти принципи роботи з відсутніми даними | Загальне порівняння методів |

У машинному навчанні, існують сучасні, різні методи, які обробляють відсутні дані. Ця література надає конкретні поради та методологічні підходи для кращого розуміння та вирішення проблеми відсутніх даних і невизначеності. Для науковців, аналітиків даних, які працюють з великими обсягами даних і мають справу з відсутніми значеннями та проблемами невизначеності є важливо знати: Байєсівський аналіз, методи множинного відновлення, гнучкі методи підстановки та інші підходи, описані в цих джерелах.

* 1. **Постанова задачі**

*Задача:* вивчення різних підходів до обробки пропущених значень та визначення невизначеності у великих наборах даних.

Для досягнення цієї цілі, потрібно виконати наступні етапи:

* **Постановка завдання:** на цьому етапі визначається основна мета дослідження, яка полягає у вивченні різних підходів до обробки пропущених значень та невизначеності у великих масивах даних. На цьому етапі також визначаються вимоги до якості даних і вибираються параметри для оцінки ефективності методів обробки.
* **Аналіз існуючих методів:** проводиться огляд та аналіз існуючих методів обробки пропущених значень та невизначеності, таких як методи імплікації, статистичні підходи та машинне навчання. Це дозволить визначити найбільш підходящий метод для поставленого завдання.
* **Підготовка та підготовка даних:** на цьому етапі збираються дані, необхідні для дослідження, в тому числі великі масиви даних, що містять пропущені значення. Дані проходять попередню обробку, включаючи видалення шуму і пропусків, для підготовки до подальшого аналізу.
* **Вибір і розробка методів обробки:** на основі аналізу наявних методів і характеристик даних обирається конкретний метод обробки, який найкраще відповідає цілям дослідження. Розробляються алгоритми для реалізації обраних методів обробки.
* **Реалізація та тестування методу:** на цьому етапі реалізується обраний метод обробки даних та проводяться тестові експерименти для оцінки його ефективності та точності. Будь-які виявлені недоліки оптимізуються для покращення результатів.
* **Оцінка ефективності та валідація:** ефективність і точність обраного методу оцінюється за допомогою таких показників, як точність, повнота, F-міра і коефіцієнт Жаккара. Результати тестуються на нових даних для перевірки стабільності та відтворюваності підходу.

**2. АНАЛІЗ МАТЕРІАЛІВ ТА МЕТОДІВ**

Спочатку необхідно підготувати дані і збір датасету, цей етап є найважливішим. Датасет – це основа для будь-якого аналізу даних. Він містить інформацію, яка використовується для роботи з моделями машинного навчання. Тому наявність пропущених значень у датасеті може значно ускладнити аналіз та викликати невизначеність у результатах. Будуть використовуватися існуючі датасети на сайті Kaggle, який містить різні великі набори даних з пропущеними значеннями.

Буде проаналізовано різні підходи для обробки пропущених значень та визначення невизначеності у великих наборах даних, і різні математичні моделі для передбачення цільових стовпців, такі як: дерево рішень, KNN, логістична регресія, Наївний Баєс, підсилення градієнту. В наступному розділі детальніше проаналізуємо датасети, які були використанні у цій курсовій роботі.

Вибір датасетів має великий вплив на метод обробки пропущених значень. І для цього було вибрано пару датасетів, які є суттєво відрізняються один від одного, вони також містять великі набори даних і мають NaN значення.

* 1. **Опис набору даних**

Вибір правильного датасету відіграє важливу роль для обробки пропущених значень та визначення невизначеності у великих наборах даних, оскільки:

1. Датасет відіграє велику роль у обробці NaN значень та визначенні невизначеності, оскільки він є основою для аналізу даних та впливає на точність та достовірність результатів.
2. Дані в датасеті повинні бути якісними і достовірними, він повинен бути без помилок або неточностей, та відображати реальну ситуацію.

Через правильний датасет ми отримаємо найкращу обробку NaN значень, та змогу перевірки найкращих методів для обробки NaN значень.

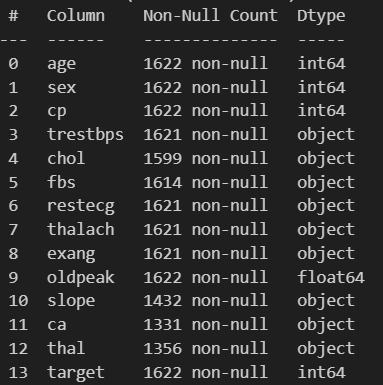
* + 1. **Heart Attack Dataset**

Набір даних складається з трьох менших датасетів. Усі вони знаходиться на платформі Kaggle у відкритому доступі. Цей набір містить дані про стан здоров’я людини, і як ці показники впливають на здоров’я людини.

|  |  |  |  |
| --- | --- | --- | --- |
| Статистика датасету | | Тип змінних | |
| Кількість стовпців | 14 | Категорійні | 9 |
| Кількість змінних | 1622 | Числові | 5 |

**Таблиця 1** Статистика датасету

У таблиці 1 показується скільки у наборі даних є категоріальних і числових стовпців. Разом 3 датасети містять 1622 рядків даних і 14 стовпців, з яких 9 стовпців є категоріальними, а 5 числовими.



**Рисунок 1** Інформація про датасет

Рисунок 1 дає інформацію про тип стовпців, імена стовпців і їх кількість і кількість пропущених значень у кожному стовпці. Кожен стовпець пояснює характеристику людини: її вік, стать, тип болю в грудях, артеріальний тиск, рівень цукру в крові, стани серця, кількість великих судин.

Інформація про значення стовпців у наборі даних пояснено у таблиці 2:

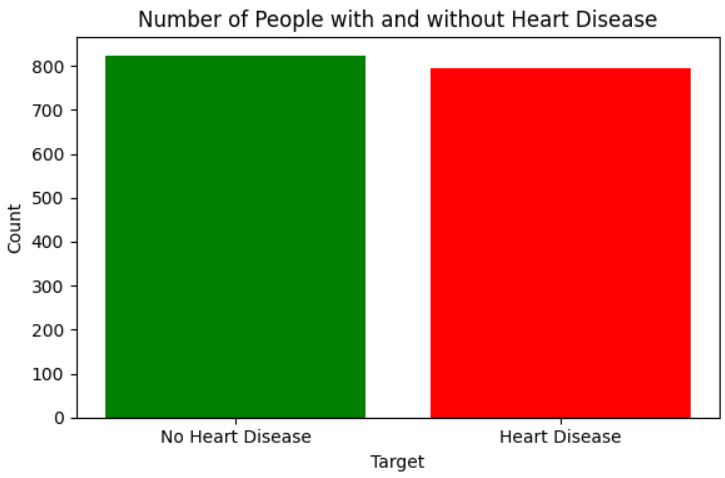
|  |  |
| --- | --- |
| **Ім’я колонки** | **Пояснення значення** |
| Age | Вік |
| Sex | Стать |
| Cp | Тип болю в грудях:  1 – типова стенокардія  2 – атипова стенокардія  3 – неангіальний біль  4 – без симптомний |
| Trbps | артеріальний тиск у стані спокою |
| Chol | холестеральний в мг/дл, отриманий за допомогою датчика ІМТ. |
| Fbs | рівень цукру в крові > 120 мг/дл, де 1 = так; 0 = ні |
| Restecg | Результати електрокардіографії в спокої |
| Thalach | Максимальна досягнута частота серцевих скорочень |
| Exng | стенокардія фізичного навантаження, де 1 = так, 0 = ні |
| Oldpeak | показник зміни електрокардіографічних хвиль після фізичного навантаження |
| Slp | нахил піка СТ-сегменту під час навантаження |
| Caa | кількість великих судин, значення від 1 до 3 |
| Thall | досягнута максимальна частота серцевих скорочень |
| Target | цільова колонка, 0 – немає ймовірності серцевого нападу, 1 = є ймовірність серцевого нападу |

**Таблиця 2** Пояснення характеристик датасету

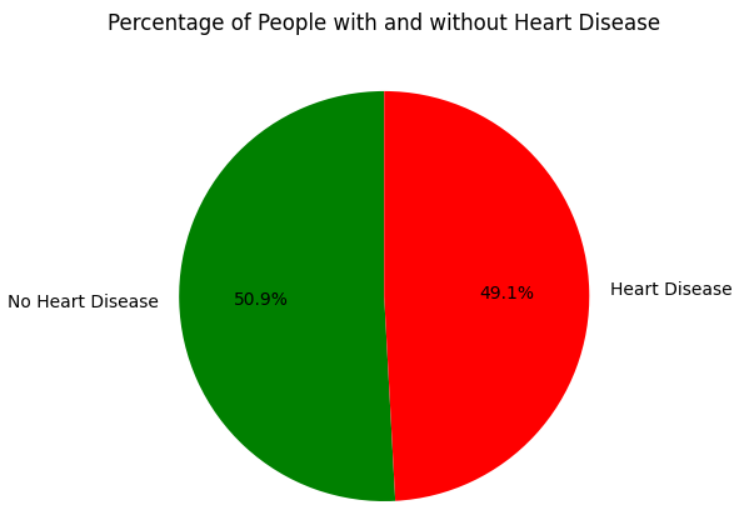
У таблиці 2 показується характеристика стовпців, за що вони відповідають. Датасет є збалансованим, оскільки усі ознаки мають одинаковий вплив на результат.

Цільова колонка цього набору даних є Target, де будемо передбачувати чи можливий у людини інфаркт чи ні. Де 0 – не можливо, а 1 – можливий інфаркт.

У рисунку відображено кількість людей які мали інфаркт і не мали інфаркту.

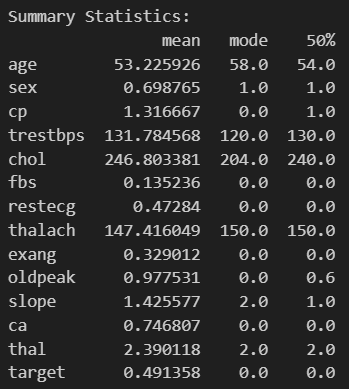


**Рисунок 2** Кількість людей з і без інфаркту, стовпчаста діаграма



**Рисунок 3** Кількість людей з і без інфаркту, кругова діаграма

На рисунку 3 і рисунку 4 показано діаграми, які мають співвідношення людей з інфарктом і без інфаркту. У датасеті є більше людей, які не мають інфаркт. Оскільки їхня частка в датасеті 50.9%. Пацієнти які мають інфаркт у датасеті займають 49.1% всього набору даних.



**Рисунок 4** Опис метрик

На рисунку 4 показано середнє значення, мода і медіана кожного стовпця у датасеті. Як бачимо на Рисунку 4 середній вік людини за середнім значенням досягає 53 роки, за модою 58 років, а за медіаною 54 роки. Також дуже відрізняються такі ствопці один від одного: chol, oldpeak, cp. Стать – переважають 1, тобто чоловіки.

Цей набір даних аналізує фактори ризику серцевого нападу на основі цих медичних даних та допомагає визначити зв'язок між різними параметрами та ймовірністю виникнення серцевого нападу. Також цей датасет не містить пропущені значення.

* + 1. **Covid-19 Dataset**

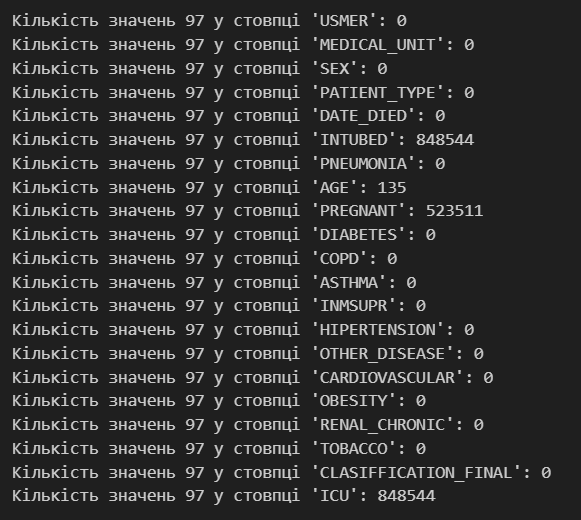
Даний набір даних є у відкритому доступі на сайті Kaggle. Цей датасет містить інформацію про здоров’я людини під час захворювання на коронавірус у 2020 році.

|  |  |  |  |
| --- | --- | --- | --- |
| Статистика датасету | | Тип змінних | |
| Кількість стовпців | 21 | Категорійні | 8 |
| Кількість змінних | 1048575 | Числові | 13 |

**Таблиця 3** Статистика датасету

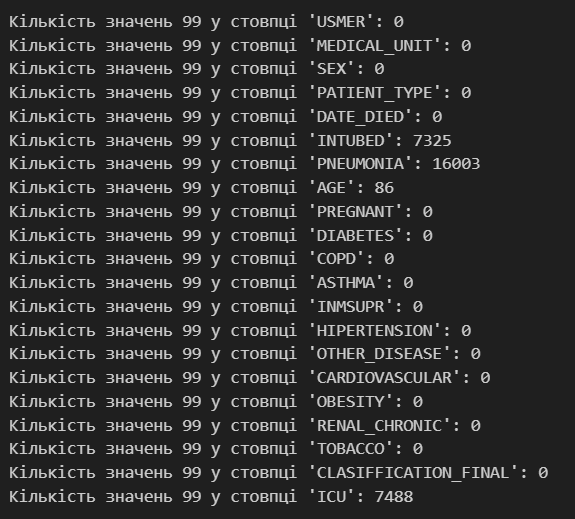
Таблиця 3 дає інформацію скільки у наборі даних є категоріальних і числових стовпців. Набір даних містить 1048575рядків даних і 21 стовпців, з яких 8 стовпців є категоріальними, а 13 числовими. Порівняно з першим датасетом Heart Attack Dataset, він є більш об’ємним в рази, а саме цей датасет більший в 646 разів за попередній. Також даний набір даних має на 7 більше стовпців ніж попередній датасет, але він містить на 1 категоріальний стовпець менше, і на 8 більше числових стовпців.

У датасеті пропущені значення не є NaN, а замінені на числа 97 і 99, а для дати на дату 9999-99-99.



**Рисунок 5** Інформація про датасет

Рисунок 5 демонструє кількість пропущених значень, які замінені на число 97 у кожному стовпці.

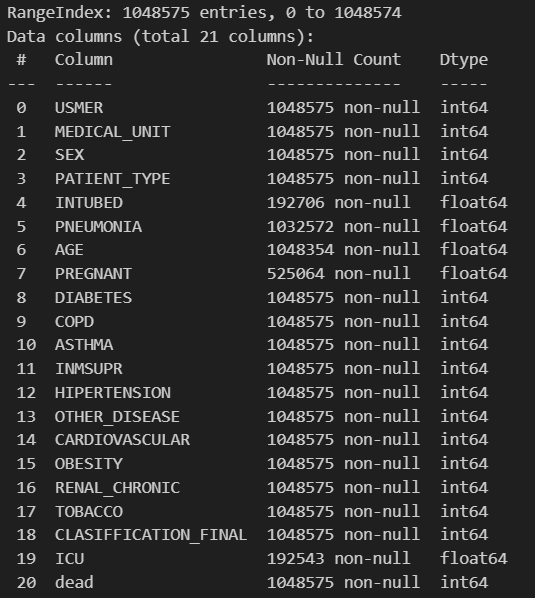


**Рисунок 6** Інформація про датасет

Рисунок 6 демонструє кількість пропущених значень, які замінені на число 99 у кожному стовпці.

Переведемо значення 97, 99 і 9999-99-99 в NaN значення, перевіримо заальну кількість пропущених значень.

Рисунок 7 дає інформацію про стовпці, їх імена і кількість.



**Рисунок 7** Інформація про датасет

Рисунок 7 показано 21 стовпець їх тип і кількість пропущених значень у кожному стовпці. Найбільше пропущених значень є у стовпцях з назвою ‘ICU’, ‘INTUBED’, ‘PREGNANT’. Також є невеликі пропущені значення в колонках з назвою ‘AGE’, ‘PNEUMONIA’.

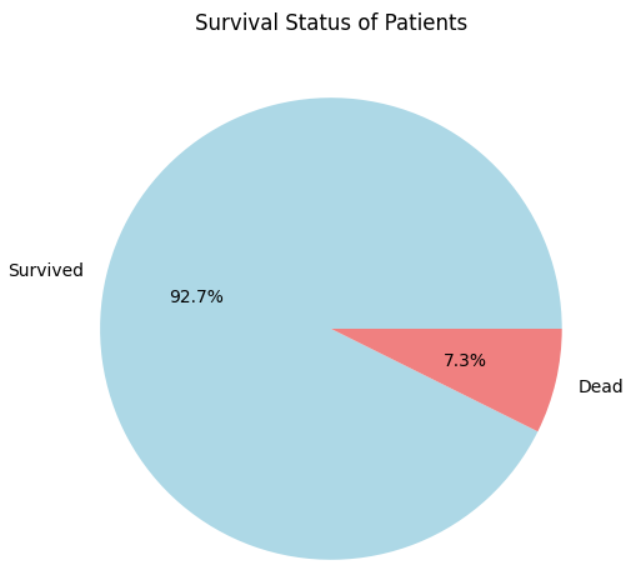
Інформація про значення стовпців у наборі даних пояснено у таблиці 4:

|  |  |
| --- | --- |
| **Ім’я колонки** | **Пояснення значення** |
| Sex | Стать  0 – жінка  1 – чоловік |
| Age | Вік людини |
| Сlassification | Р езультати тесту на COVID. Значення 1-3 означають, що пацієнт був діагностований з COVID різних ступенів важкості. 4 або більше означає, що пацієнт не є носієм COVID або тест недостовірний. |
| Patient type | Тип догляду за пацієнтом в медичному закладі.  1 – виписаний додому  2 – госпіталізований. |
| Pneumonia | Чи має пацієнт запалення повітряних мішків. |
| Pregnancy | Чи вагітний пацієнт. |
| Diabetes | Чи має пацієнт діабет. |
| COPD | Хронічна обструктивна хвороба легень. |
| Asthma | Показник астми. |
| Inmsurp | Чи має пацієнт імунодепресію. |
| Hypertension | Показник гіпертонії. |
| Сardiovascular | Показник, чи має пацієнт захворювання серця чи судин. |
| Renal chronic | Чи має пацієнт хронічну ниркову хворобу. |
| Other disease | Інші захворювання. |
| Obesity | Пацієнт має ожиріння чи ні. |
| Tobacco | Показник тютюну |
| USMER | Показує, чи отримував пацієнт лікування в медичних одиницях першого, другого чи третього рівня. |
| Medical unit | Тип установи Національної медичної системи, яка надавала догляд. |
| Intubed | Чи був пацієнт підключенй до апарату штучної вентиляції легень. |
| Icu | Показує, чи був пацієнт прийнятий до відділення інтенсивної терапії. |
| Date died | Показує чи пацієнт помер. Якщо пацієнт помер, вказує дату смерті; якщо ні, то значення 9999-99-99. |

**Таблиця 4** Пояснення характеристик датасету

Цільова колонка є date died, оскільки ця колонка дає ключове значення, чи людина при певному стані здоров’я смертельно хвора чи може виздоровіти.

У рисунку відображено процентне співвідношення – кількість людей які померли і не померли від коронавірусу.



**Рисунок 8** Кількість людей які померли і не померли від коронавірусу

Рисунок 8, Набір даних містить 92.7% не померлих людей і 7.3% померлих людей. Хоч і різниця між даними є дуже великою, але це не робить датасет не збалансованим, оскільки це відображає реальну ситуацію в світі.



**Рисунок 9** Опис метрик

На рисунку 9 показано середнє значення, мода, медіана кожного стовпця у датасеті. У датасеті сильно відрізняються результати від кожної метрики. У колонці ‘Age’ відрізняється вік між модою і середнім значенням з медіаною, він містить 30 і 40, 41 рік. Також сильно відрізняється ‘MEDICAL\_UNIT’ – медіана відрізняється від моди і середнього значення на 4 позначки, що має великий вплив на результати. У цьому датасеті результати середнього значення, медіани і моди є в більшості одинаковими порівняно з результатами першого датасету. Результати середнього значення є дробовими, оскільки ми сумуємо усі дані, і ділимо на їх кількість у стовпці.

Він містить 1048575 різних випадків, рядків. У цьому датасеті ми визначатимемо, які симптоми коронавірусу призводять до смерті. Ключова цільова колонка є date\_died. Оскільки через цю колонку ми знаємо, як впливають симптоми на людину.

* + 1. **Diabetes Health Indicators Dataset**

Цей датасет містить інформацію про здоров'я та показники хронічних захворювань серед американців, яка була зібрана за допомогою опитування BRFSS2015, проведеного Центром контролю та профілактики захворювань (CDC).

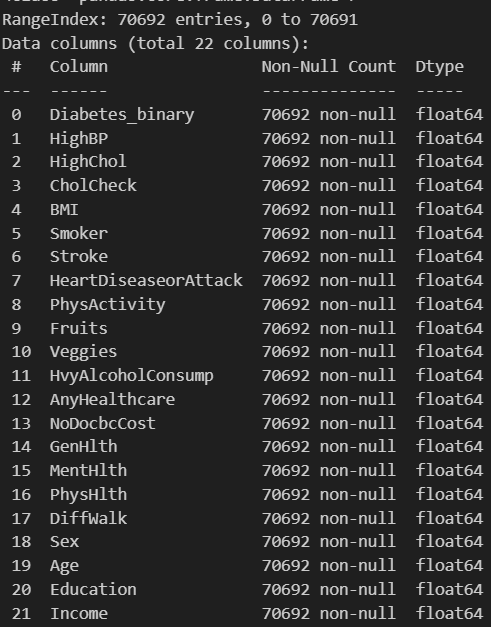
Датасет містить відповіді від 70692 осіб та включає 21 змінну ознаку, яка визначає наявність діабету серед опитаних.

|  |  |  |  |
| --- | --- | --- | --- |
| Статистика датасету | | Тип змінних | |
| Кількість стовпців | 21 | Категорійні | 10 |
| Кількість змінних | 70692 | Числові | 11 |

**Таблиця 5** Статистика датасету

У таблиці 5 вираховано з набору даних скільки стовпців є є категоріальні і числові. Датасет містить відповіді від 70 692 осіб та включає 21 змінну, з яких 10 стовпців є категоріальними, а 11 числовими, які визначають наявність діабету серед опитаних. Даний датасет є більшим за перший датасет який містив 1622 рядки і меншим за другий датасет який містить 1048575 даних про хворих людей на корона вірус. Він містить найбільшу кількість категоріальних стовпців – 10.

Рисунок 10 дає інформацію про тип стовпців, імена стовпців і їх кількість і кількість пропущених значень у кожному стовпці.



**Рисунок 10** Інформація про датасет

На рисунку 10 зображено назву кожного стовпця, його тип і кількість пропущених значень. У цьому датасеті немає пропущених значень, їх вставлятимемо штучно: 5%, 10%, 15 від загального обсягу даних для порівняння результатів взалежності від пропущених значень.

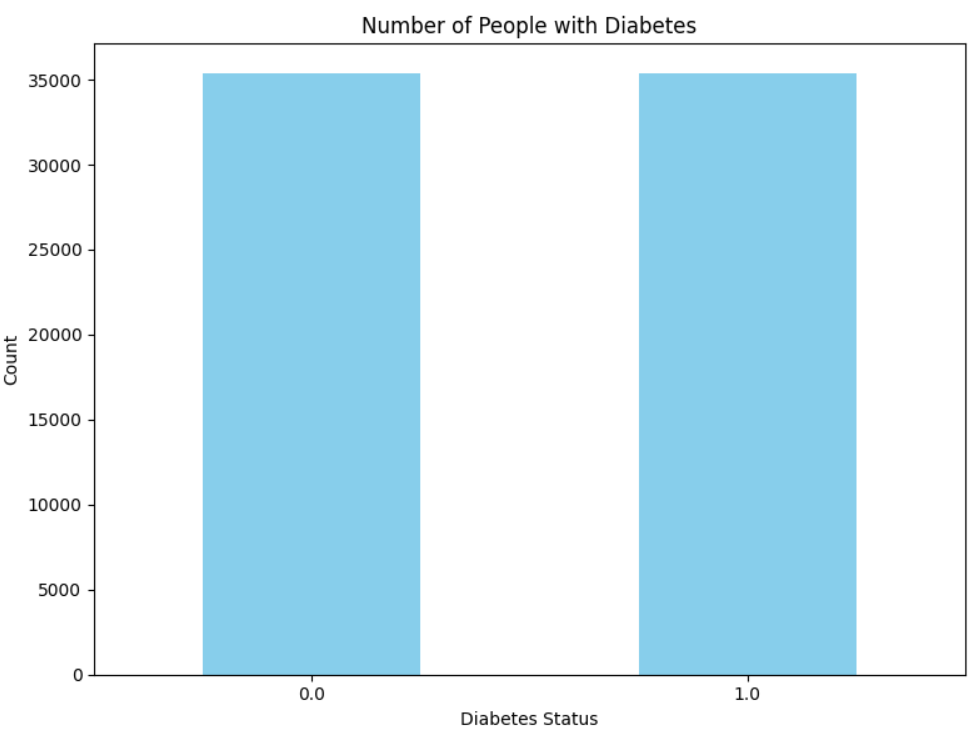
Інформація про значення стовпців у наборі даних пояснено у таблиці 2:

|  |  |
| --- | --- |
| **Ім’я колонки** | **Пояснення значення** |
| Diabetes\_binary | Показник наявності діабету. Має 2 класи: 0 - відсутність діабету або тільки під час вагітності, 1 – діабет. |
| HighBP | Показник наявності високого артеріального тиску. |
| HighChol | Показник наявності високого рівня холестерину. |
| CholCheck | Показник проведення перевірки рівня холестерину. |
| BMI | Індекс маси тіла, який використовується для оцінки ступеня ожиріння або худоби учасників. |
| Smoker | Чи пацієнт курить чи ні. |
| Stroke | Показник наявності інсульту. |
| HeartDiseaseorAttack | Показник наявності серцево-судинних захворювань або серцевого нападу. |
| PhysActivity | Показник фізичної активності. |
| Fruits | Кількість споживаних фруктів. |
| Veggies | Кількість споживаних овочів. |
| AnyHealthcare | Показник доступності будь-якої медичної допомоги. |
| HvyAlcoholConsump | Показник важкого споживання алкоголю. |
| NoDocbcCost | Показник відсутності витрат на медичну допомогу. |
| GenHlth | Загальний стан здоров'я. |
| MentHlth | Психічний стан. |
| PhysHlth | Фізичне здоров’я. |
| DiffWalk | Показник складності ходьби. |
| Sex | Стать 1 – жінка, 2 – чоловік. |
| Age | Вік. |
| Education | Рівень освіти. |
| Income | Дохід пацієнта. |

**Таблиця 6** Пояснення характеристик датасету

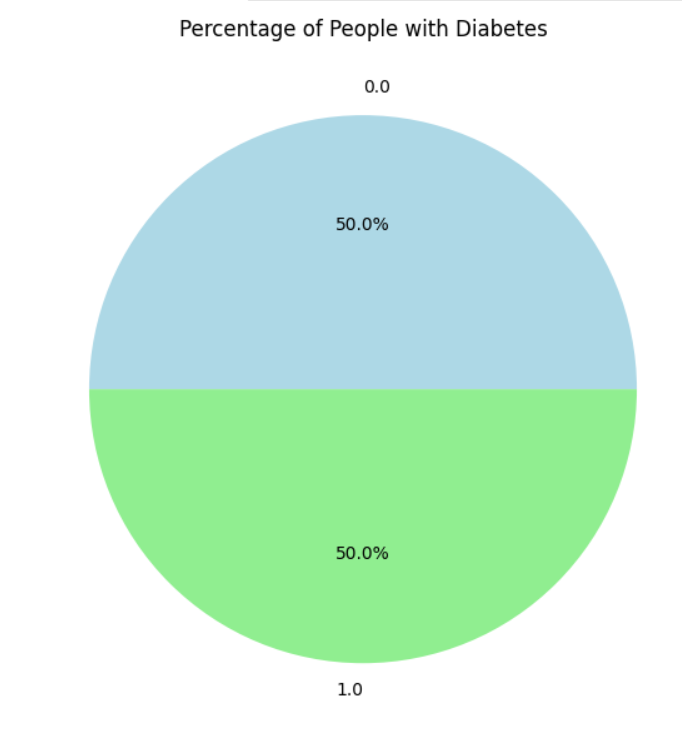
У таблиці 6, цільова колонка є Diabetes\_binary, де будемо передбачувати чи можливий діабет. Де 0 – немає діабету, а 1 – можливий діабет.

У рисунку відображено кількість людей які мали діабет і не мали діабету в датасеті.



**Рисунок 11** Кількість людей з діабетом, стовпчаста діаграма

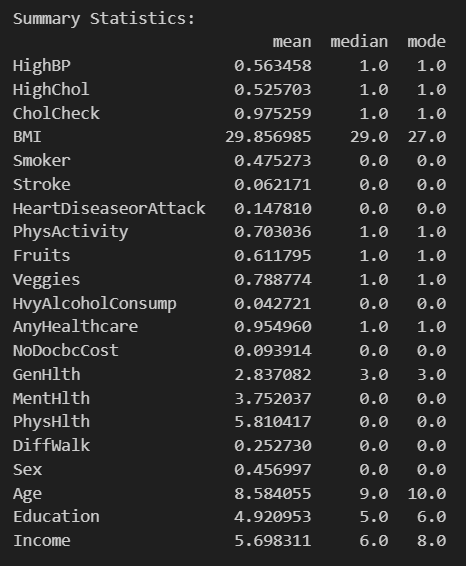
Рисунок 11 показує стовпчасту діаграму, де кількість людей які хворі на діабет, де 0 – немає діабету, а 1 – є діабет. Бачимо, що даний датасет є збалансованим, містить одинакову кількість здорових і хворих людей.



**Рисунок 12** Кількість людей з діабетом, кругова діаграма

Рисунок 12 відображає кругову діаграму, де кількість людей які хворі на діабет, де 0 – немає діабету, а 1 – є діабет. Показано в відсотковому відношенні, де 50% датасету є люди з діабетом, а інші 50% людей без діабету.

На рисунках 11 і 12 видно, що набір даних є збалансованим. Кількість людей які мають діабет в наборі даних є 50% від загальної кількості людей.



**Рисунок 13** Опис метрик

На рисунку 13 показано середнє значення, мода, медіана кожного стовпця у цьому датасеті. Дані одинакові у стовпцях медіани і моди, крім середнього значення.

Основна цільова змінна Diabetes\_binary має 2 класи:

0 – відсутність діабету або тільки під час вагітності,

1 – діабет.

**2.2 Опис та обгрунтування вибраних моделей**

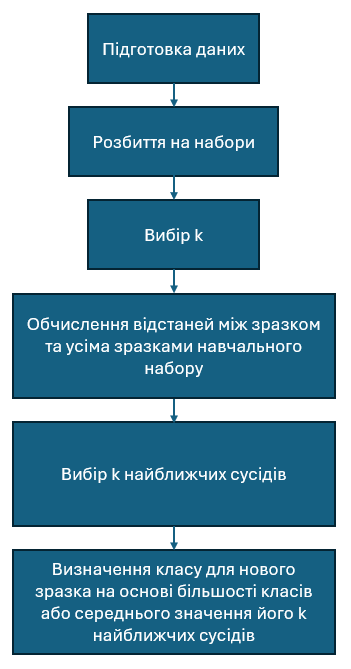
Ми використаємо різні математичні моделі, для того, щоб досягти максимальну ефективність для тренування, передбачування моделей і виявлення найкращої обробки NaN значень. А саме: КНН, логістична регресія, дерева рішень, Наївний Баєс, підсилення градієнта (gradient boosting). Пояснимо різницю між кожним методом машинного навчання:

**2.2.1 KNN**

Алгоритм k-найближчих сусідів (KNN) – це один з найпопулярніших алгоритмів машинного навчання для класифікації та регресії. Він базується на припущенні, що схожі об'єкти зазвичай належать до одного класу або мають схожі значення цільової змінної.

**Послідовність роботи алгоритму KNN**:

1. Підготовка даних.
2. Розбиття даних на навчальний та тестові набори.
3. Вибір значення k (кількість найближчих сусідів).
4. Обчислення відстаней між новим зразком та усіма зразками навчального набору.
5. Вибір k найближчих сусідів.
6. Визначення класу для нового зразка на основі більшості класів (тобто для класифікації) або середнього значення (для регресії) його k найближчих сусідів.



**Рисунок 14** Послідовність алгоритму KNN

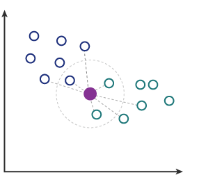
Рисунок 14 показує загальну послідовність алгоритму KNN.

**Переваги алгоритму KNN:**

* Простота реалізації.
* Ефективність у вирішені задач класифікації з невеликим набором атрибутів.
* Можна використовувати для задач регресії і класифікації.
* Не вимагається попереднє навчання моделі
* Адаптивний до змін
* Може моделювати складні нелінійні залежності між ознаками та цільовими змінними.

**Недоліки алгоритму KNN:**

* Обчислювально витратний для великого набору даних.
* КНН дуже чутливий до шуму в даних, що може призвести до неправильної класифікації або прогнозів.



**Рисунок 15** Зображення класифікації точки

Рисунок 15 відображає приклад пошуку класифікації точки серед інших відомих точок.

Отже, KNN - це ефективний алгоритм з простим механізмом прийняття рішення, але його ефективність залежить від обсягу даних, наявності шуму та вибору оптимального значення параметра k.

**2.2.2 Логістична регресія**

Логістична регресія – це метод для аналізу даних, що використовується для прогнозування ймовірностей бінарних подій. Вона є одним з основних алгоритмів класифікації у машинному навчанні, особливо для випадків, коли маємо два можливих результати або класи. Основна ідея логістичної регресії полягає в тому, щоб побудувати математичну модель, яка оцінює ймовірність того, що зразок належить до певного класу. Ця ймовірність виражається у вигляді значень від 0 до 1, де 0 означає неможливість події, а 1 - абсолютна достовірність.

**Послідовність роботи логістичної регресії:**

1. **Підготовка даних**: підготувати і обробити дані для датасету, перевірити вірність даних і наявність NaN значень.
2. **Розділення даних:** розділити дані на навчальний набір та тестовий набір для оцінки ефективності моделі.
3. **Побудова моделі:** використовуючи навчальний набір, побудувати математичну модель логістичної регресії, що відображає зв’язок між залежними та незалежними змінними.
4. **Навчання моделі:** застосувати метод максимальної ймовірності або інші методи оптимізації для навчання моделі на навчальних даних.
5. **Оцінка моделі:** через тестовий набір оцінити точність та ефективність моделі на прогнозування нових зразків.

**Формула алгоритму:**

*P(y = 1|x) =*

де

*P(y = 1|x) –* ймовірність, що вихідний результат *𝑦* дорівнює 1 (позитивний клас), при умові вхідних ознак *𝑥*.

e – число Ейлера

- параметри моделі, які підбираються під час навчання.

*x1*, *x2, …, xn* – вхідні ознаки, які використовуються для прогнозування.

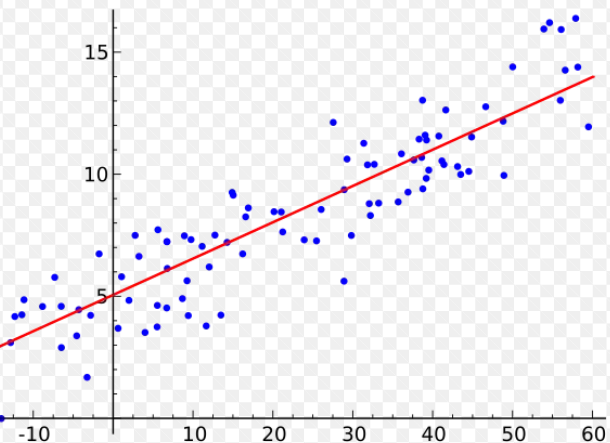
*y* – бінарний результат (категорія), який прогнозується 0 або 1.

**Переваги логістичної регресії:**

* Це простий і потужний алгоритм для бінарної класифікації.
* Логістична регресія може враховувати надлишкові ознаки або мультиколінеарність, що дозволяє уникнути перенавчання та покращити загальну ефективність моделі.
* Логістична регресія дає змогу встановлювати власні пороги для прийняття рішень. Це є великою перевагою, у тих задачах, коли необхідно мати баланс між чутливістю та специфічністю.

**Недоліки логістичної регресії:**

* Вона може бути не ефективною у моделюванні складних нелінійних залежностей в даних, через те що це лінійна модель.
* Аномалії в даних можуть впливати на точність та надійність результатів логістичної регресії.
* Логістична регресія може бути обмежена у вирішенні завдань, де існують сильні нелінійні залежності між ознаками та цільовими змінними.



**Рисунок 16** Зображення прямої для логістичної регресії

На рисунку 16 намальовано пряму, за якою класифікуватимуться точки.

Логістична регресія - це потужний і широко використовуваний алгоритм класифікації, який дозволяє прогнозувати ймовірність настання події на основі вхідних ознак. Однак ефективність цього алгоритму може залежати від характеристик даних та налаштувань моделі.

**2.2.3 Дерева рішень**

Дерево рішень (Decision tree) — це непараметричний контрольований алгоритм навчання, який можна застосовувати для опрацювання як дискретних, так і безперервних даних. Він має ієрархічну структуру дерева, яка складається з кореневого вузла, гілок, внутрішніх вузлів і листових вузлів. Основне завдання алгоритму, це розподілити набір даних на підмножини на основі найважливішого атрибута у цих даних.

**Основні сфери застосування дерев рішень:**

* Класифікація даних
* Регресивний аналіз даних

**Покроковий опис алгоритму обрахунку дерев рішень:**

1. Підготувати дані. Для усіх значень знайти значення Gini(D).
2. Знайти Ginik(D) для кожного атрибута k: Ginik(D) = ∑ (Di / D ) \* Gini(Di)
3. Розбити за мінімальним значенням індексу Gini. Тобто елемент з найменшим значенням Gini, буде коренем нашого дерева, а всі подальші гілки і листки додаватимуться за зростанням значення індексу Gini.

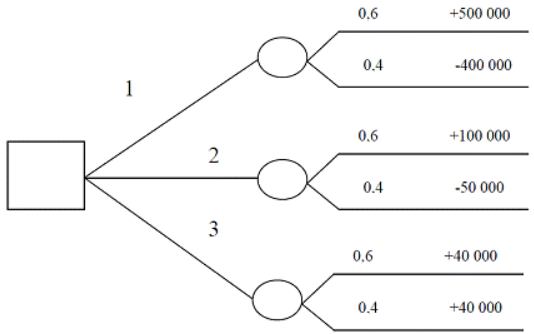
**Переваги алгоритму дерева рішень:**

* Дерева рішень легко інтерпретувати, оскільки вони можуть бути візуалізовані у вигляді деревоподібних структур з правилами прийняття рішень.
* Дерева рішень не потребують нормалізації даних, оскільки вони розглядають кожну ознаку окремо та нечутливі до масштабування даних.
* Дерева рішень можна використовувати як для класифікації, так і для регресійних завдань.
* Дерева рішень можуть ефективно обробляти великі обсяги даних без значних витрат часу на підготовку даних.

**Недоліки алгоритму дерева рішень:**

* Дерева рішень можуть перенавчатися на тренувальних даних і демонструвати високу точність на навчальному наборі, але низьку узагальнюючу здатність на нових даних.
* Дерева рішень виявляють обмежену здатність моделювати складні нелінійні залежності між ознаками та цільовими змінними.
* Невеликі зміни у навчальному наборі можуть призводити до значних змін у структурі та результативності дерева рішень.

Отже, дерева рішень – це простий та зрозумілий спосіб моделювання рішень, який може бути застосований до різних завдань машинного навчання.



**Рисунок 17** Класифікація дерева рішень

НА рисунку 17 показано класифікацію числа, до інших чисел. Де гілка 1 має межі від 100 до 500, гілка 2 має межі від 50 до 100, гілка 3 має межі від 40 до 0.

**2.2.4 Наївний Баєс**

Наївний Баєс - це простий, але потужний алгоритм машинного навчання, який використовує ймовірність для прийняття рішень. Його основна ідея полягає в тому, що він визначає ймовірність того, що певний об'єкт належить до певного класу, враховуючи його атрибути чи характеристики. Наприклад, він може визначати ймовірність того, що лист - це спам або не спам, відповідно до його вмісту та властивостей. Це дає можливість автоматично розподіляти об'єкти в різні категорії, що є корисним для багатьох задач, таких як класифікація документів, виявлення шахрайств, медичні діагнози тощо.

**Покроковий опис алгоритму Наївного Баєса:**

1. Спочатку потрібно підготувати дані, включаючи навчальний набір з об'єктами та відповідними класами чи категоріями, які потрібно передбачити. Обробити дані, щоб вони були коректними і не містили NaN значення.
2. Для кожного класу обчислюють ймовірності віднесення об'єкта до цього класу, використовуючи формулу теореми Баєса. Це включає розрахунок апостеріорної ймовірності P(C|x) - ймовірності класу С при умові, що відомі атрибути x.
3. Наївний Баєс припускає незалежність між атрибутами, тобто він вважає, що значення кожного атрибуту впливає на клас незалежно від інших атрибутів. Це спрощує обчислення ймовірностей, але це припущення не завжди відповідає реальним даним. Навчаємо модель.
4. Після навчання моделі і обчислення ймовірностей для кожного класу відбувається прогнозування класу для нових об'єктів. Об'єкт призначається класу з найвищою апостеріорною ймовірністю.

**Формула**

*P(Ck|x1,x2, …, xn) =*

де:

*P(Ck|x1,x2, …, xn)* – ймовірність належності прикладу *x* до класу *Ck*, при умові, що відомі ознаки *x1,x2, …, xn* .

*P(Ck)* – ймовірність того, що вибраний приклад належить класу *P(Ck)* без урахування будь-яких ознак.

– умовна ймовірність ознак при умові, що приклад належить класу *Ck .*

– ймовірність вектору ознак у загальному випадку, незадежно від класу.

**Переваги алгоритму Наївного Баєса:**

* Простий та ефективний алгоритм, що швидко навчається на великих обсягах даних.
* Особливо ефективний для роботи з категоріальними та текстовими даними, де він може показати високу точність.
* Не потрібно налаштовувати багато гіперпараметрів або враховувати складні взаємозв'язки між ознаками.

**Недоліки алгоритму Наївного Баєса:**

* Наївний Баєс базується на припущенні про незалежність між ознаками, що може бути недостатнім для деяких даних, де взаємозв'язки складніші.
* У деяких складних завданнях, де залежність між ознаками значна, наївний Баєс може показати низьку точність прогнозування.
* Якщо в навчальному наборі виникають нульові ймовірності для деяких комбінацій ознак, алгоритм може давати неправильні прогнози.
* Наївний Баєс може бути вразливим до шуму в даних та впливу незначущих ознак, що може погіршити якість прогнозів.
* Якщо взаємозв'язки між ознаками виходять за межі наївного припущення, алгоритм може втратити точність та неадекватно моделювати дані.

Наївний Баєс працює швидко та ефективно навіть з великими обсягами даних, він відносно простий у реалізації та має низький ризик перенавчання. Він добре працює з великою кількістю атрибутів та може бути використаний для категоріальних та числових даних. Однак, наївний Баєс робить дуже спрощені припущення про незалежність атрибутів, що може бути неідеальним у деяких реальних ситуаціях. Також, якщо в наборі даних є атрибути, які сильно впливають на результат, але вони неправильно припускаються як незалежні, то це може призвести до неадекватних прогнозів.

**2.2.5 Підсилення градієнту**

Підсилення градієнту – використовується для підвищення точності прогнозування моделі шляхом послідовного навчання низькорівневих моделей, кожна з яких коригує попередні помилки, що здійснюється на основі градієнта (зміни) функції втрати. Основна ідея полягає в тому, щоб послідовно створювати слабкі моделі, які компенсують недоліки попередніх моделей.

**Послідовність алгоритму:**

1. **Ініціалізація базової моделі:** початкова модель може бути простою моделлю, наприклад, константним прогнозом або середнім значенням цільової змінної.
2. **Розрахунок помилок:** розраховуємо вектор помилок, який представляє різницю між спостереженими значеннями та прогнозами початкової моделі.
3. **Навчання нової моделі:** нова модель навчається на векторі помилок, тобто спробує покращити прогнози попередньої моделі.
4. **Оновлення прогнозів:** по закінченні навчання нової моделі, прогнози попередньої моделі оновлюються, додаючи до них прогнози нової моделі з вагою (зазвичай, за допомогою швидкості навчання або кроку навчання).
5. **Розрахунок нового вектора помилок:** рахуємо новий вектор помилок, який тепер враховує виправлені прогнози.
6. **Повторення кроків 3-5:** повторюємо цей процес (зазвичай кілька разів) для навчання додаткових моделей та покращення прогнозів.
7. **Закінчення:** після завершення всіх ітерацій, ансамбль моделей об'єднується для утворення кінцевого прогнозу.

**Формула:**

*Fm(x) = Fm-1(x) +ή\* hm(x)*

де:

*Fm(x)* – прогноз моделі на *m* кроці для вхідних значень *x*.

*Fm-1(x)* - прогноз моделі на *m* кроці для вхідних значень *x*.

*ή –* швидкість навчання (learning rate), яка визначає, наскільки швидко модель навчається на кожному кроці.

*hm(x) –* слабка модель (зазвичай дерево рішень), яка навчається на помилках попередніх прогнозів *Fm-1(x).*

**Переваги алгоритму підсилення градієнту:**

* Підсилення градієнту часто дає дуже точні результати, особливо в порівнянні з іншими алгоритмами.
* Цей метод може ефективно моделювати складні взаємозв'язки в даних та робити точні прогнози.
* Може працювати без необхідності передобробки даних.
* Алгоритм може використовувати різні функції втрати в залежності від конкретної задачі.

**Недоліки алгоритму підсилення градієнту:**

* Якщо не налаштовувати гіперпараметри належним чином, підсилення градієнту може перенавчитися.
* Цей метод може вимагати більше часу та ресурсів для навчання порівняно з іншими алгоритмами.
* Підсилення градієнту може бути вразливим до шуму в даних та випадкових варіацій, що може призвести до невірних прогнозів.

Підсилення градієнту є потужним та ефективним алгоритмом для прогнозування в машинному навчанні, здатним до вирішення складних задач та досягнення високої точності. Проте його використання вимагає належного налаштування та обережності, щоб уникнути перенавчання та врахувати його вразливість до шуму.

**2.3 Опис методів обробки NaN значень**

Вибір конкретного методу обробки NaN значень повинен базуватися на характеристиках самого датасету та вимогах конкретної аналітичної задачі.

**Імпутація** – це заповнення пропущених значень на основі інших доступних даних. Наприклад: медіана, середнє значення, мода, заміна на константу, видалення NaN значень, KNN.

**Різновиди імпутації:**

* **Медіана** – це значення, яке ділить впорядкований набір чисел навпіл, так що половина чисел менша за медіану, а половина більша.

Як працює метод: для кожного стовпця з пропущеними значеннями обчислюється медіана з відомих значень, і пропущені значення заповнюються цією медіаною.

Цей метод використовується тоді, коли дані мають викиди або відносно велику варіацію, оскільки медіана менш чутлива до викидів, ніж середнє значення, то вона є ефективним вибором для числових даних з великою кількістю викидів або аномалій.

* **Середнє значення** – всі числа у стовпці додаються і діляться на кількість чисел у наборі стовпця. Цей метод підходить для числових даних, які мають нормальний розподіл і відсутність викидів. Використання середнього значення може бути ефективним для великих наборів даних з невеликою кількістю пропусків.
* **Мода** – це показник центральної тенденції у статистиці, який визначається як найбільш часто зустрічатиме значення в наборі даних. Цей метод добре підходить для категоріальних даних або даних з дискретними значеннями, де можна визначити найбільш часто зустрічаємі значення. Він ефективний, коли потрібно заповнити пропуски у категоріальних змінних або змінних з обмеженим набором значень.
* **Видалення NaN значень** – це видалення пустих значень, цей метод використовують, коли малий набір даних і мала кількість пропусків.
* **Заміна на константу** – це замінювання всіх пропущених значень на певну фіксовану константу.
* **KNN** (метод найближчих сусідів) можна використовувати для імпутації пропущених значень (NaN) у даних. Основна ідея полягає в тому, що для кожного пропущеного значення можна знайти k найближчих сусідів з відомими значеннями цієї ознаки і використати їхні значення для заповнення пропущеного.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Метод імпутації** | **Вплив викидів** | **Тип даних (числові / категоріальні)** | **Розмір набору даних** | **Ефективність** |
| Медіана | Менше впливають | Числові | Будь-який | Залежить від розподілу даних. Ефективний при наявності викидів |
| Середнє значення | Чутливий | Числові | Будь-який | Ефективний для нормально розподілених даних без викидів |
| Мода | Не впливають | Категоріальні | Будь-який | Ефективний для категоріальних даних з невеликою кількістю пропусків |
| Видалення NaN значень | Видаляється | Будь-який | Малий набір даних | Ефективний при малій кількості пропусків |
| Заміна на константу | Не впливають | Числові /  Категоріальні | Будь-який | Ефективний для наборів даних з невеликою кількістю пропусків |
| KNN | Залежить від налаштування методу | Будь-який | Будь-який | Ефективний при великих наборах даних з різноманітними типами даних |

**Таблиця 7** Порівняння методів імпутації

**2.4 Визначення похибки**

Точність – це відношення правильно класифікованих прикладів до загальної кількості прикладів у тестовому наборі даних. Формула для обчислення точності виглядає так:

*Accuracy =*

де *m –* це кількість правильно класифікованих прикладів, а *n –* загальна кількість прикладів.

У цьому розділі порівняли набори даних між собою, з’ясували які колонки і дані містять датасети. Яка кількість пропущених значень і які цільові колонки необхідно передбачити. З’ясували, які методи машинного навчання використовуватимуться для тренування моделі. Визначили методи імпутації за допомогою яких обро,лятимуться пропущені значеня.

**3 ЕКСПЕРИМЕНТИ**

Експерименти важливі для обробки NaN значень у машинному навчанні, оскільки вони дозволяють визначити оптимальний підхід до обробки відсутніх значень що в свою чергу можу суттєво вплинути на якість моделі та її прогнозування. Давайте розглянемо кілька причин, чому експерименти з обробкою прпущених значень є важливими:

1. Використання різних методів обробки NaN значень може вплинути на точність та ефективність моделі. Експерименти допомагають визначити найкращий підхід для конкретного набору даних.
2. Недоцільна обробка пропущених значень може призвести до перекосів у навчальному процесі та прогнозах моделі. Експерименти допомагають зменшити ці ризики і підібрати оптимальний спосіб обробки відсутніх значень.
3. Через те, що існують різні методи обробки NaN значень, таких як заповнення медіаною, середнім значенням, видалення рядків з NaN, заміна константою і т. д. Експерименти дозволяють порівняти ці методи та обрати найефективніший для конкретного набору даних і задачі.
4. Якщо неправильно обробитипропущені значення, тооброблені NaN значення можуть займати зайвий обсяг пам'яті та час обробки. Експерименти знаходять найефективніший спосіб обробки NaN значень, що дозволяє оптимізувати використання ресурсів.

Роль моделі відіграє велике значення в обробці пропущених значень:

1. Різні моделі можуть підходити для різних видів даних та задач. Наприклад, деякі моделі можуть бути більш чутливими до NaN значень, тоді як інші можуть краще з ними впоратися. Експерименти допомагають вибрати оптимальну модель для конкретної задачі.
2. Під час тренування моделі, яка обробляє NaN значення, важливо вибрати оптимальний метод заповнення, щоб уникнути перекосів та покращити точність прогнозів.

Узагальнюючи, експерименти з обробкою NaN значень та вибір оптимальної моделі для прогнозування значень дозволяють покращити якість та ефективність моделі машинного навчання, зменшуючи перекоси та оптимізуючи використання ресурсів.

**3.1 Опис бібліотеки sklearn**

|  |  |
| --- | --- |
| Sklearn.pipeline | Модуль дозволяє створювати послідовність операцій обробки даних та моделювання. Об’єднює кілька етапів обробки даних та моделювання в один пайплайн для зручності та ефективності коду. |
| sklearn.model\_selection | Функція для створення крос-валідації моделі, тобто для оцінки її ефективності на кількох різних тренувальних та тестових наборах даних. Приймає на вхід модель, датасет, а повертає масив результатів метрик. |
| sklearn.linear\_model | Цей клас реалізує логістичну регресію. |
| sklearn.neighbors | Реалізує метод КНН. |
| sklearn.tree | Цей клас реалізує метод машинного навчання дерево рішень. |
| sklearn.ensemble | Дозволяє використовувати градієнтне прискорення. |
| sklearn.naive\_bayes | Цей клас дає можливість використовувати метод Наївного Баєса. |

**Таблиця 8** Методи бібліотеки sklearn

У таблиці 8 показано, які методи бібліотеки ми будемо імпортувати і використовувати.

**3.2 Реалізація методів імпутації і методів машинного навчання з датасетом Heart Attack Dataset**

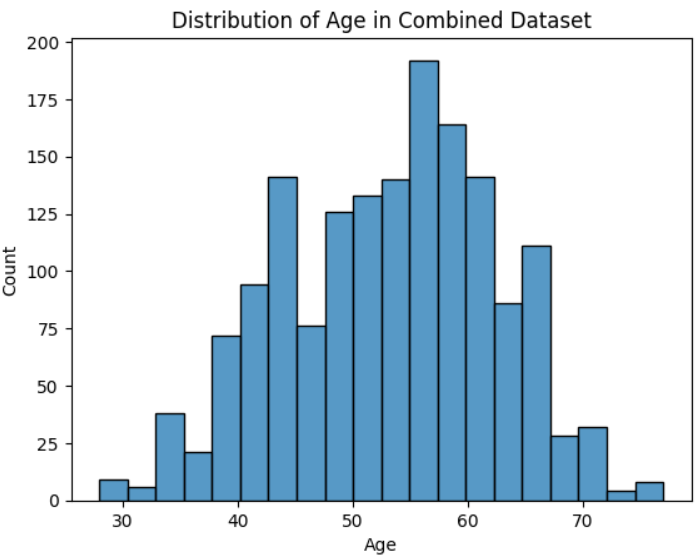
Перед робото з датасетом, перевіримо чи він є збалансованим.

Перевірка балансу датасету є важливою, оскільки незбалансовані дані можуть призводити до неправильних або перекручених результатів моделі машинного навчання. Причини для чого необхідно робити балансування даних:

1. Вплив на якість моделі – незбалансовані дані можуть призвести до виникнення проблем, таких як перекошення класів (class imbalance), де модель може бути схильною передбачати більшість прикладів як один клас, ігноруючи меншість класів. Це може призвести до низької точності та викривлення метрик якості моделі.
2. Наявність достатньої кількості даних – балансований датасет забезпечує належну кількість прикладів для кожного класу, що дозволяє моделі навчатися адекватно для всіх класів та робити коректні прогнози.
3. Зменшення перекрученості – незбалансовані дані можуть призвести до перекрученості (bias) моделі в бік домінуючого класу, що може вплинути на її здатність до генералізації на нові дані.

Перевірка балансу датасету може бути здійснена за допомогою різних методів, але ми використаємо лише один – візуалізуємо класи. Графічне відображення кількості прикладів у кожному класі може допомогти оцінити баланс датасету.

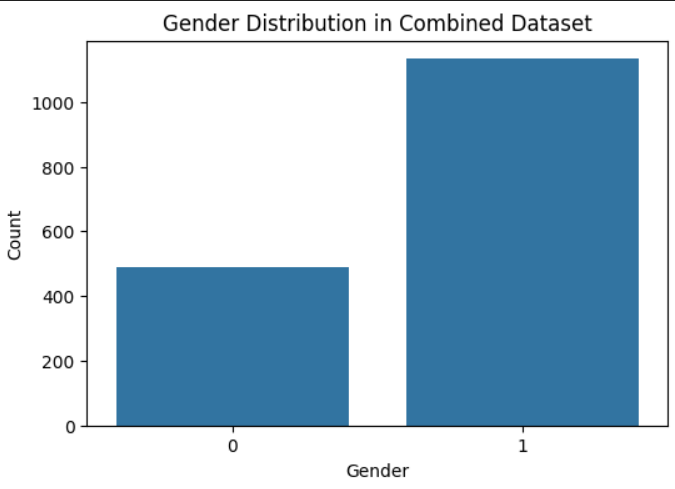
Перевірка датасету за віком:



**Рисунок 18** Діаграма за віком

На рисунку 18 зображено статистику людей за віком від 27 до 78 років. Найбільше людей мають вік від 55 до 60 років, кількість людей при цьому віці від 140 до 185.

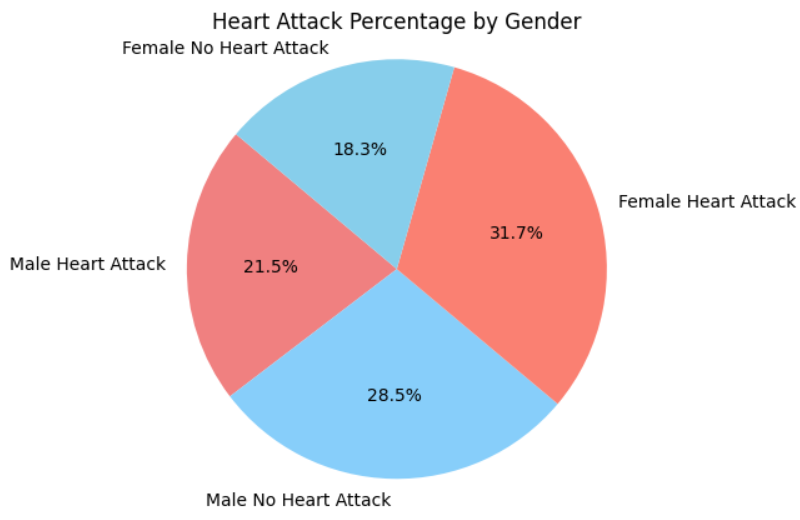
Перевірка датасету за статтю, чоловік чи жінка:



**Рисунок 19** Діаграма за статтею

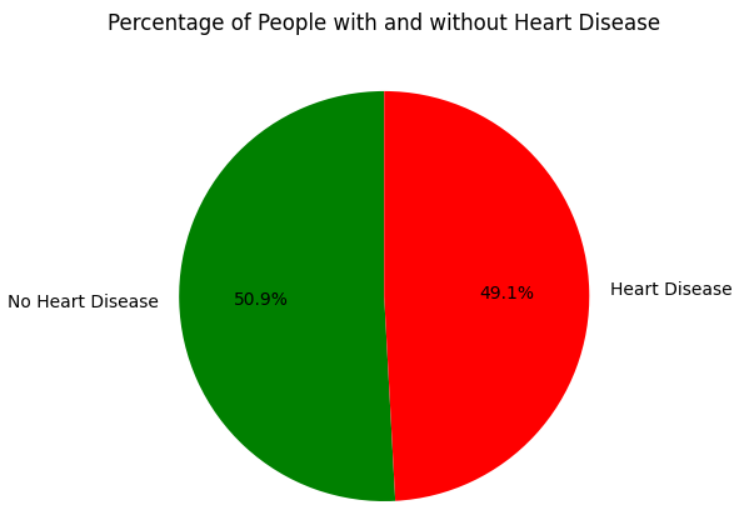
На рисунку 19 показується кількість чоловіків і жінок у датасеті, які мали або не мали інфаркту. Бачимо, що чоловіків в 2 рази більше за жінок.

Перевірка датасету, скільки інфарктів у чоловіків і жінок:

****

**Рисунок 20** Діаграма кількості чоловіків і жінок з і без інфаркту

На рисунку 20 зображено у відсотковому відношенні кількість чоловіків і жінок, які мали і не мали інфаркту. Найбільше у датасеті жінок з інфарктом, їхня відсоткова частка датасету має 31.7%. Жінок без інфаркту є частка 18.3%. Чоловіків які не мали інфаркту досягає 28.5%, а чоловіки які мали інфаркт досягає 21.5%. Тобто чоловіків, які пережили інфаркт менше в 1.5 рази ніж жінок. А жінки які не мали інфаркту в 1.5 рази менше ніж чоловіків які теж не мали інфаркту.



**Рисунок 21** Кількість людей з і без інфаркту, кругова діаграма

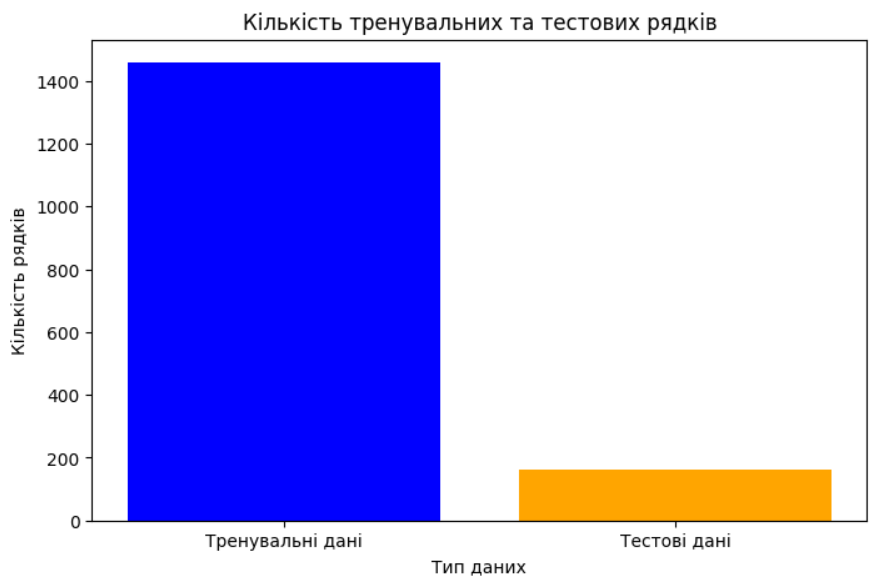
На рисунку 20 і рисунку 21 показано діаграми, які мають співвідношення людей з інфарктом і без інфаркту. У датасеті є більше людей, які не мають інфаркт. Оскільки їхня частка в датасеті 50.9%. Пацієнти які мають інфаркт у датасеті займають 49.1% всього набору даних.

З Рисунку 21 видно що даний набір даних є збалансованим, оскільки цільова колонка має майже співвідношення в 50%.

Даний датасет збалансований, то буде доречно порівнювати методи імпутації і методи машинного навчання.

Для оброблення пропущених значень у цьому датасеті, ми використаємо усі зазначенні вище методи імпутації і методи машинного навчання.

Також було перевірено, яка кількість NaN значень є кожному стовпці датасету – Рисунок 20. Це необхідно для того, щоб дізнатися, які стовпці потрібно обробляти від пропущених значень.

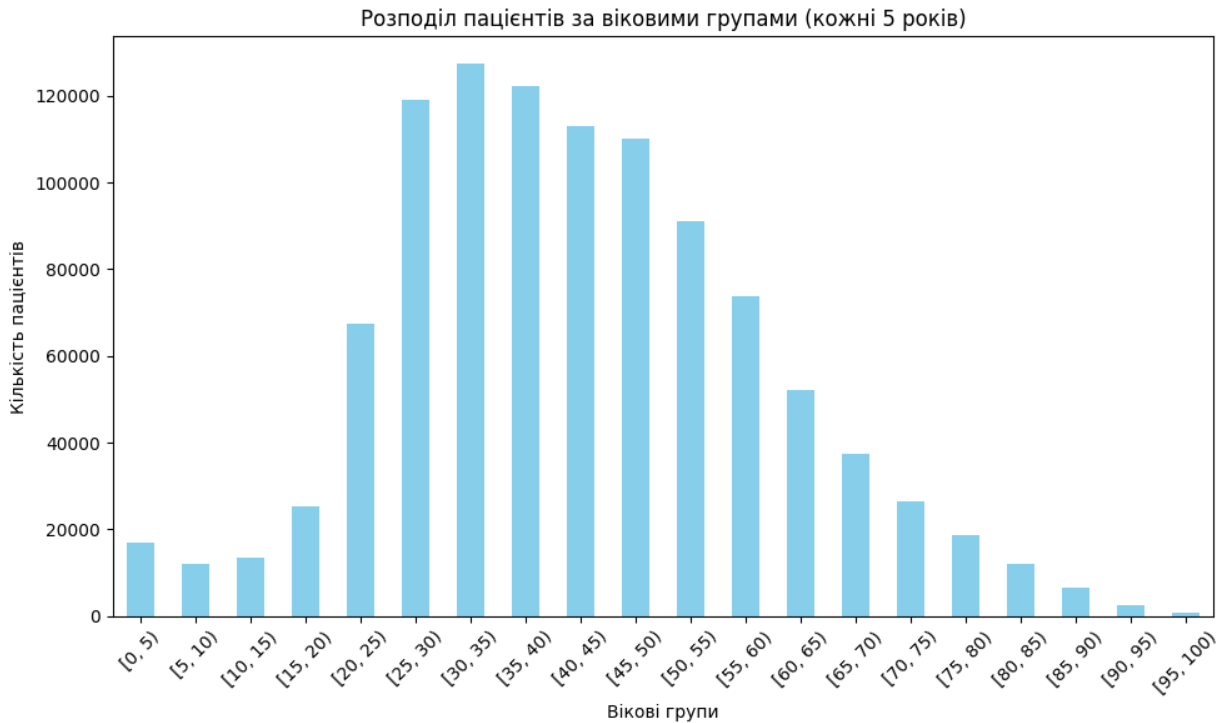


**Рисунок 22** Розділення набору даних на тренувальну і тестову

Рисунок 22 відоражає стовпчасту діаграму, на якій зображено кількість даних, які використовуватимуть для тренування моделі і кількість даних для тестування даних. Тестові дані займають 10% всього набору даних, а тренувальний набір даних займає 90% всього датасету.

**3.3 Реалізація методів імпутації і методів машинного навчання з датасетом Covid-19 Dataset**

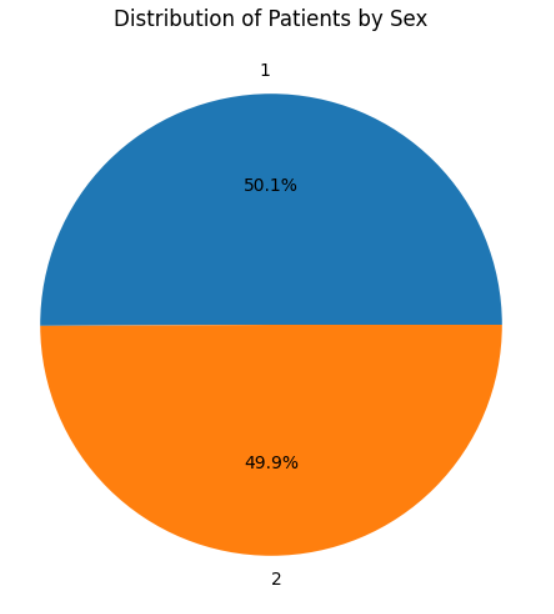
У рисунку 8 показано, що даний датасет є збалансованим. Датасет є збалансованим, оскільки це відображає реальну ситуацію. Також перевіримо кількість людей за віком і статтю:



**Рисунок 23** Відображення статистики за віком

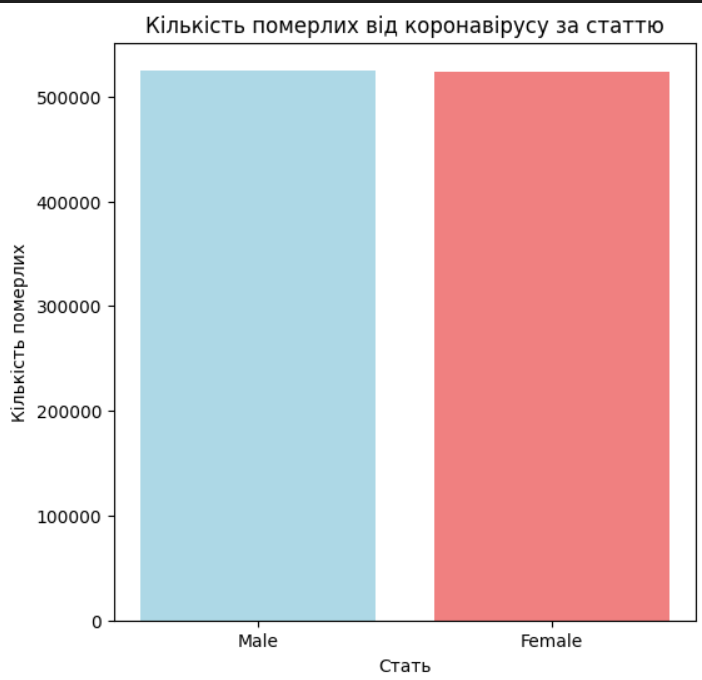
У Рисунку 23 відображається статистика за віком. У датасеті переважає кількість людей від 25 до 50 років. Найбільше захворювань у людей віком 30 – 35 років.

Перевірка датасету за статтю, чоловік чи жінка:



**Рисунок 24** Діаграма за статтею

На рисунку 24 показується кількість чоловіків і жінок у датасеті, які вижили або померли від коронавірусу. Бачимо, що чоловіків така сама кількість як і жінок. Частка чоловіків 50.1%, а жінок 49.9%.



**Рисунок 25** Кількість чоловіків і жінок померлих від коронавірусу

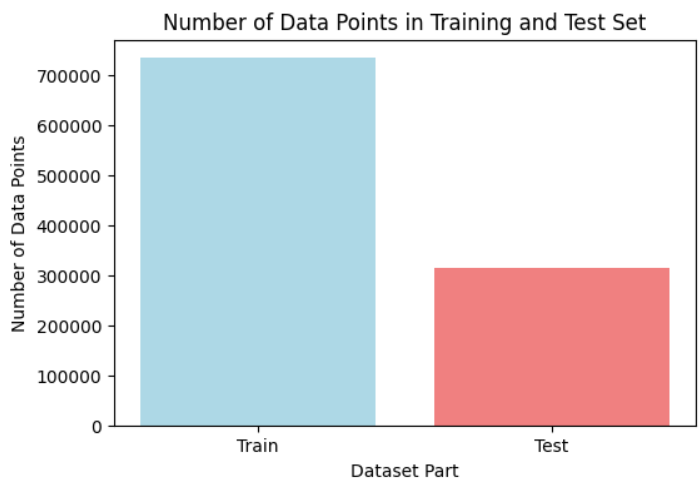
Рисунок 25 показано діаграми, які мають співвідношення чоловіків і жінок померлих від коронавірусу. Померлих чоловіків 50% і померлих жінок 50% серед померлих людей у датасеті.

З Рисунку 25 видно що даний набір даних має є збалансованим за статтею, оскільки цільова колонка має майже співвідношення в 50%.

Даний датасет збалансований, то буде доречно порівнювати методи імпутації і методи машинного навчання.

Для оброблення пропущених значень у цьому датасеті, ми використаємо усі зазначенні вище методи імпутації і методи машинного навчання.

Також було перевірено, яка кількість NaN значень є кожному стовпці датасету – Рисунок 1. Це необхідно для того, щоб дізнатися, які стовпці потрібно обробляти від пропущених значень.

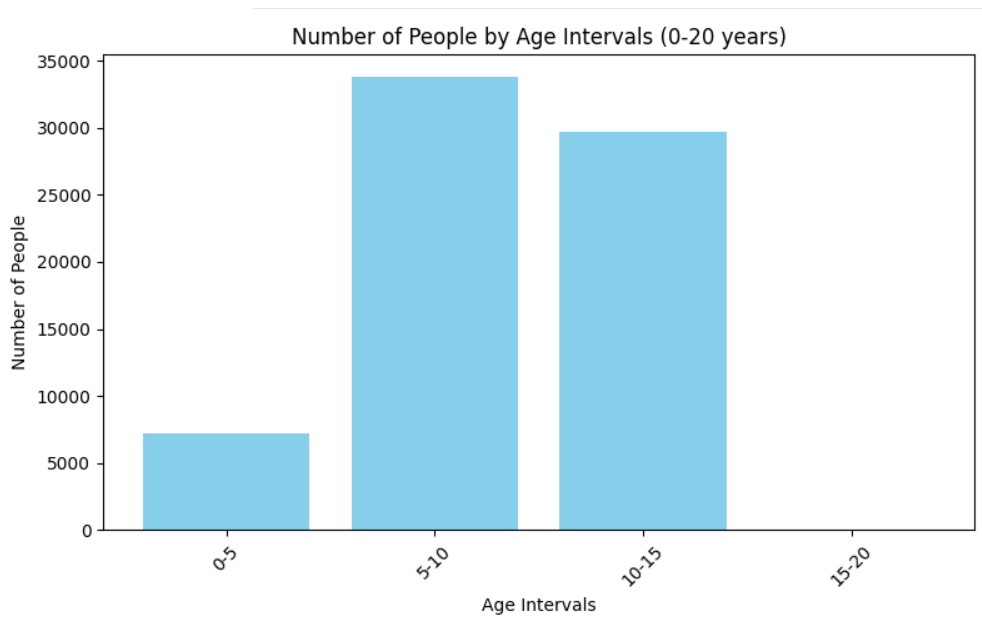


**Рисунок 26** Розділення набору даних на тренувальну і тестову

Рисунок 26 відоражає стовпчасту діаграму, на якій зображено кількість даних, які використовуватимуть для тренування моделі і кількість даних для тестування даних. Тестові дані займають 30% всього набору даних, а тренувальний набір даних займає 70% всього датасету.

**3.3 Реалізація методів імпутації і методів машинного навчання з датасетом Diabetes Dataset**

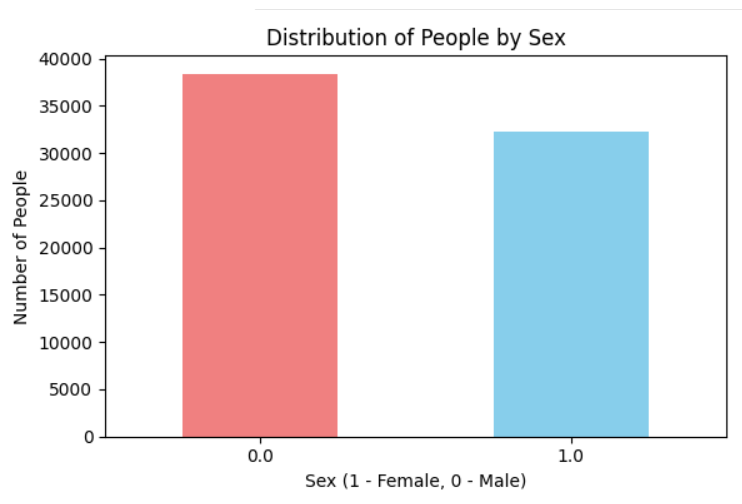
У рисунку 11 показано, що даний датасет є збалансованим. Датасет є збалансованим, оскільки він містить 50% людей з діабетом і 50% людей без діабету. Розберемо такі колонки, як вік і стать:



**Рисунок 27** Відображення статистики за віком

У Рисунку 27 відображається статистика за віком. У датасеті переважає кількість дітей від 5 до 10 років. Найменше дітей від 0 до 5 років, які мають діабет.

Перевірка датасету за статтю, чоловік чи жінка:



**Рисунок 28** Діаграма за статтею

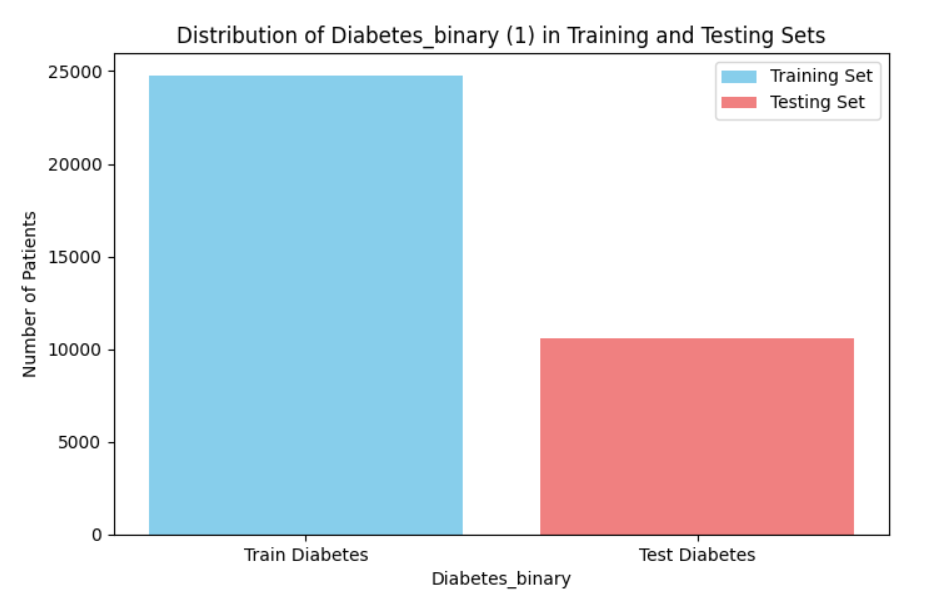
На рисунку 28 відображається діаграма – кількість чоловіків і жінок у датасеті, які мають і не маю діабет. Бачимо, що чоловіків така сама кількість як і жінок. Частка чоловіків 32306, а жінок 38386, тобто переважають жінки.

З Рисунку 7 і 6 видно що даний набір даних має є збалансованим за статтею, оскільки цільова колонка має майже співвідношення в 50%.

Даний датасет збалансований, і буде доречно порівнювати методи імпутації і методи машинного навчання.

Для оброблення пропущених значень у цьому датасеті, ми використаємо усі зазначенні вище методи імпутації і методи машинного навчання.

Також було перевірено, яка кількість NaN значень є кожному стовпці датасету – Рисунок 1. Це необхідно для того, щоб дізнатися, які стовпці потрібно обробляти від пропущених значень.



**Рисунок 29** Розділення набору даних на тренувальну і тестову

Рисунок 29 відоражає стовпчасту діаграму, на якій зображено кількість даних, які використовуватимуть для тренування моделі і кількість даних для тестування даних. Тестові дані займають 30% всього набору даних, а тренувальний набір даних займає 70% всього датасету.

**3.4 Опис методів імпутації**

Після отриманих значень про стовпці, зробимо псевдо код, який оброблятиме пропущенні значення різними методами імпутації. Після цього, зберігатиме окремий датасет з певним методом імпутації для подальшого навчання моделей.

Псевдокод для обробки пропущених значень за допомогою медіани, середнього значення, моди, NaN значенням і заміною на константу зі значенням 0:

Input: combined\_dataset (початковий датасет), columns\_to\_impute (список стовпців для імпутації)

# Створення копій початкового датасету для кожного методу імпутації

median\_dataset = combined\_dataset.copy()

mean\_dataset = combined\_dataset.copy()

mode\_dataset = combined\_dataset.copy()

constant\_dataset = combined\_dataset.copy()

deleted\_nan\_dataset = combined\_dataset.copy()

# Цикл для кожного методу імпутації

for column in columns\_to\_impute:

    # Імпутація за допомогою медіани

    median\_value = combined\_dataset[column].median()

    median\_dataset[column] = median\_dataset[column].fillna(median\_value)

    # Імпутація за допомогою середнього значення

    mean\_value = combined\_dataset[column].mean()

    mean\_dataset[column] = mean\_dataset[column].fillna(mean\_value)

    # Імпутація за допомогою моди

    mode\_value = combined\_dataset[column].mode()[0]

    mode\_dataset[column] = mode\_dataset[column].fillna(mode\_value)

    # Видалення рядків з NaN значеннями

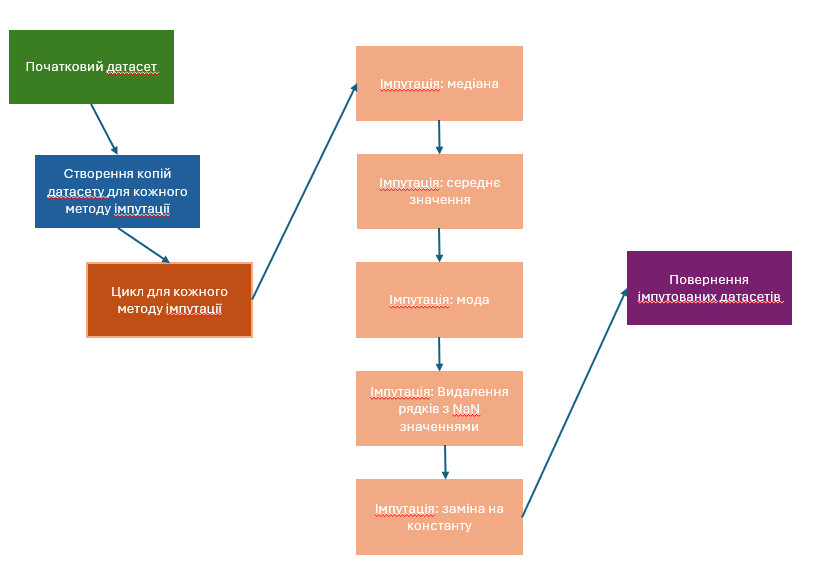
    deleted\_nan\_dataset = deleted\_nan\_dataset.dropna(subset=[column])

    # Імпутація за допомогою константи

    constant\_value = 0  # Вибір деякої константи зі значенням 0

    constant\_dataset[column] = constant\_dataset[column].fillna(constant\_value)

Послідовність агоритму для обробки пропущених значень:



**Рисунок 30** Кроки обробки пропущених значень

У Рисунку 30 показується послідовність дій для оброблення датасету різними методами імпутації. Спочатку створюються копії початкового датасету, це необхідно для збереження незмінного першого набору даних після методу імпутацій. Після того робиться цикл, де створюється конкретний датасет для певного виду імпутації і після того виконуються всі імпутації. У кінці виводяться усі оброблені набори даних.

Перевірили чи датасет є збалансованим і обробили набір даних від пропущених значень різними методами, розбили датасет на 5 датасетів, які оброблені різними методами імпутації.

Наступний крок – це перевірка результатів виконання програми різними методами машиного навчання, ткаих як КНН, логістична регресія, дерево рішень, градієнтне прискорення, Наївний Баєс:

models = [

    ('KNeighbors Classifier', KNeighborsClassifier()),

    ('Logistic Regression', LogisticRegression(max\_iter=1000, solver='liblinear', random\_state=42)),

    ('Decision Tree Classifier', DecisionTreeClassifier(random\_state=42)),

    ('Gradient Boosting', GradientBoostingClassifier(random\_state=42)),

    ('Naye base Classifier', GaussianNB())

]

datasets = [

    ('Median Imputation', X\_train\_median, X\_test\_median, y\_train\_median, y\_test\_median),

    ('Mean Imputation', X\_train\_mean, X\_test\_mean, y\_train\_mean, y\_test\_mean),

    ('Mode Imputation', X\_train\_mode, X\_test\_mode, y\_train\_mode, y\_test\_mode),

    ('Deleted NaN Imputation', X\_train\_deleted\_nan, X\_test\_deleted\_nan, y\_train\_deleted\_nan, y\_test\_deleted\_nan),

    ('Constant Imputation', X\_train\_constant, X\_test\_constant, y\_train\_constant, y\_test\_constant)

]

best\_model = None

best\_accuracy = 0.0

for dataset\_name, X\_train, X\_test, y\_train, y\_test in datasets:

    print(f"Dataset: {dataset\_name}")

    print("---------------------")

    for model\_name, model in models:

        pipeline = Pipeline([('model', model)])

        scores = cross\_val\_score(pipeline, X\_train, y\_train, cv=5)

        pipeline.fit(X\_train, y\_train)

        y\_pred = pipeline.predict(X\_test)

        accuracy = accuracy\_score(y\_test, y\_pred)

        print(f"Model: {model\_name}")

        print("Dataset Used: ", dataset\_name)

        print("Test Accuracy: ", accuracy)

        print()

        if accuracy > best\_accuracy:

            best\_accuracy = accuracy

            best\_model = (dataset\_name, model\_name, pipeline)

print("Best Model: ", best\_model)

**Рисунок 31** Псевдо код для навчання різними машинними моделями

Рисунок 31 є псевдокодом. Спочатку створюється масив моделей машинного навчання, а саме: КНН, логістична регресія, дерево рішень, градієнтне прискорення і Наївний Баєс. Після цього створюється масив наборами даних, які оброблені методами: середнього значення, мода, медіана, видалення пропущених значень, заміна на константу.

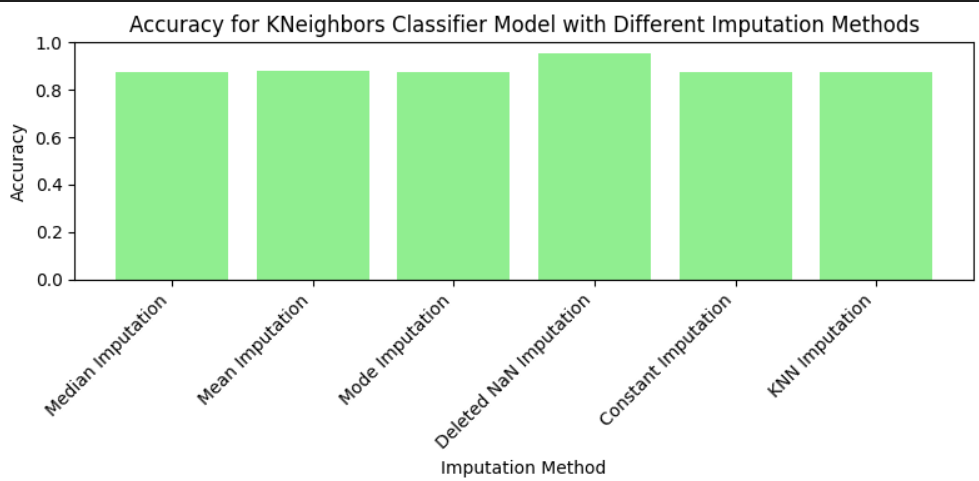
У розділі вияснили, чи доречно використовувати набори даних для порівняння результатів обробки пропущених значень, тобто перевірили чи дані наборів даних є збалансованими. Порівняли тренувальну і тестову вибірки датасетів і також порівняли датасети між собою. Також пропущені значення обробили різними методами імпутації. Використали різні методи машинного навчання для тренування моделей з різною обробкою пропущених значень.

**4 ОБГОВОРЕННЯ РЕЗУЛЬТАТІВ ЕКСПЕРИМЕНТІВ**

У цьому розділі ми детально розглянемо результати експериментів, проведених з різними методами обробки пропущених значень та застосуванням різних алгоритмів машинного навчання на різних наборах даних. Ми оцінюватимемо, які комбінації методів обробки даних та алгоритмів машинного навчання дають найкращі результати для 3 наборів даних.

**4.1 Опис і порівняння результатів використовуючи Heart Attack Dataset**

Порівняння усіх методів імпутації для певного методу машинного навчання.

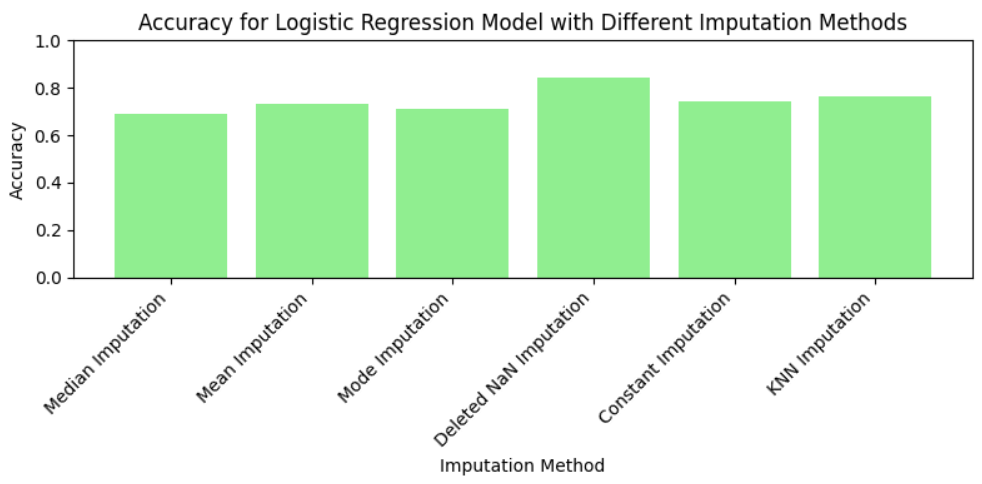


**Рисунок 32** Діаграма точності для методу КНН

|  |  |
| --- | --- |
| Модель машинного навчання КНН | |
| Види імпутації | Точність тренування |
| Медіана | 87.65% |
| Середнє значення | 88.27% |
| Мода | 87.65% |
| Видалення NaN значень | 95.48% |
| Заміна на кностанту | 87.65% |
| КНН | 87.65% |

**Таблиця 9** Результати точності методу КНН з 6-тьма методами імпутації.

Рисунок 32 показує діаграму, на якій є порівняння результатів використовуючи метод машинного навчання КНН з 6 методами обробки пропущених значень. Таблиця 9 показує точність кожного методу ообреблення пропущених значень. Рисунок 32 і Таблиця 9 показують найефективніший метод обробки пропущених значень для методу КНН у цьому датасеті є видалення пропущених значень яка досягає точності 95,48%. Найменш ефективним методом обробки пропущених значень є методи: медіана, мода, константа і КНН – з точністю 87.6%.

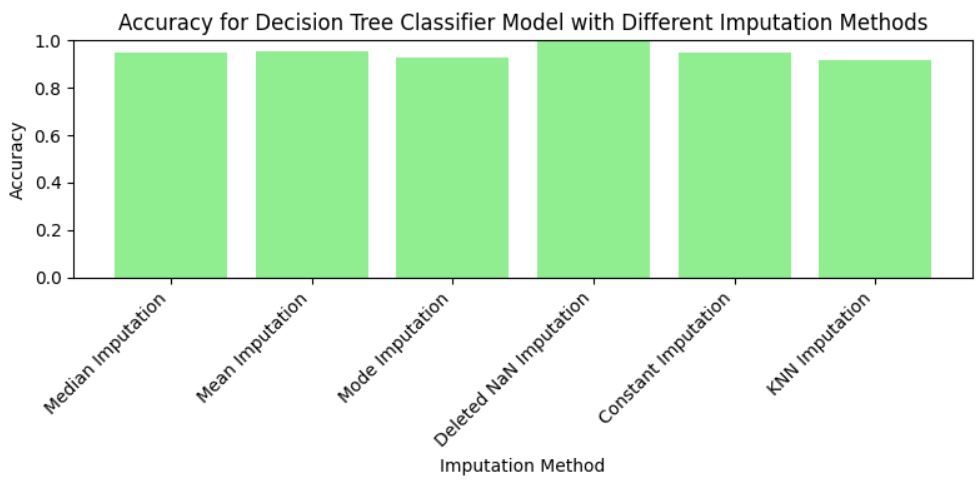
****

**Рисунок 33** Діаграма точності логістичної регресії

|  |  |
| --- | --- |
| Модель машинного навчання Логістична регресія | |
| Види імпутації | Точність тренування |
| Медіана | 69.13% |
| Середнє значення | 73.45% |
| Мода | 70.98% |
| Видалення NaN значень | 84.21% |
| Заміна на кностанту | 74.07% |
| КНН | 76.65% |

**Таблиця 10** Результати точності логістичної регресії з 6-тьма методами імпутації

Рисунок 33 є діаграмою, яка містить порівняння усіх методів оброблення пропущених значення використовуючи логістичну регресію. Рисунок 33 і Таблиця 10 демонструють, що найкраща точність досягається при видалені пропущених значень, а саме 84.21% точності, найгірша точність спостерігається у медіани і моди – 69.13% і 70.98% відповідно.

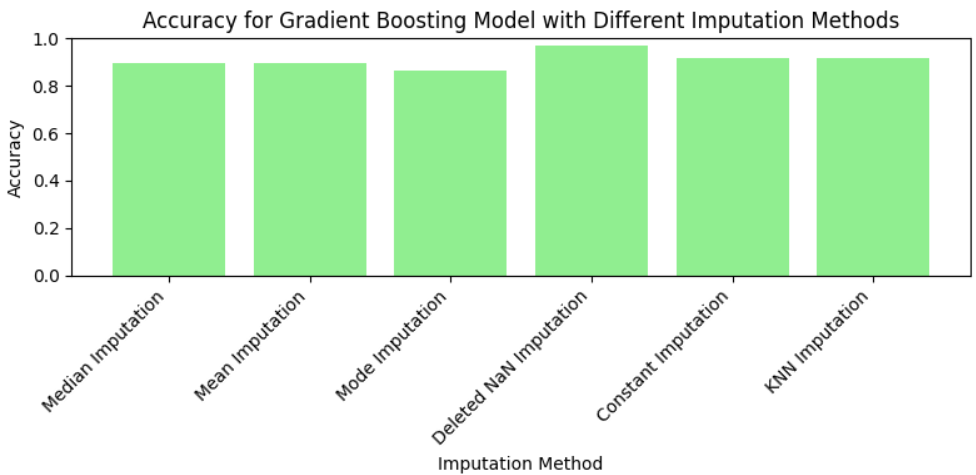
****

**Рисунок 34** Діаграма точності для дерева рішень

|  |  |
| --- | --- |
| Модель машинного навчання Дерево рішень | |
| Види імпутації | Точність тренування |
| Медіана | 95.06% |
| Середнє значення | 95.67% |
| Мода | 92.59% |
| Видалення NaN значень | 100.00% |
| Заміна на кностанту | 95.06% |
| КНН | 91.97% |

**Таблиця 11** Результати точності дерева рішень зі шістьма методами імпутації

У Рисунку 34 показано діаграму порівнянь усіх пропущених значень з методом машинного навчання: дерево рішень. Таблиця 11 показує точність кожного методу обробки пропущених значень для методу дерева рішень. Проаналізувавши рисунки і таблиці, вияснено, що найкраща точність досягається використовуючи видалення пропущених значень – 100% точності. Методи обробки пропущених значень такі як: медіана, середнє значення, заміна на константу – мають точність +- 95.3%. Найменш ефективними є мода і КНН – їхня точність 92.59% і 91.97%.

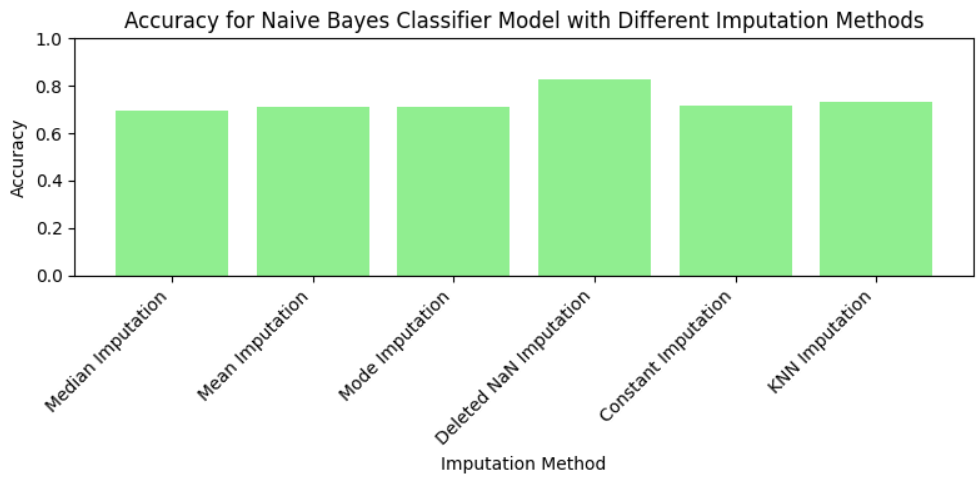
****

**Рисунок 35** Діаграма точності для градієнтного прискорення

|  |  |
| --- | --- |
| Модель машинного навчання Градієнтне прискорення | |
| Види імпутації | Точність тренування |
| Медіана | 89.50% |
| Середнє значення | 89.50% |
| Мода | 86.41% |
| Видалення NaN значень | 96.99% |
| Заміна на кностанту | 91.97% |
| КНН | 91.97% |

**Таблиця 12** Результати моделі машинного навчання – градієнтного прискорення 6 методами імпутації.

Рисунок 35 є діаграмою, яка відображає різні методи імпутації і їхню точність при виконанні з градієнтним прискоренням. Таблиця 12 надає результати при виконані алгоритму градієнтного прискорення. Рисунок 35 і Таблиця 12 надають порівняти найоптимальніші методи оброблення пропущених значень. Найоптимальніший метод є видаленням пропущених значень, який досягає точності в 96.99%, а найменш ефективний метод є мода, який має точність 86.41%.



**Рисунок 36** Діаграма точності для Наївного Баєса

|  |  |
| --- | --- |
| Модель машинного навчання Наївний Баєс | |
| Види імпутації | Точність тренування |
| Медіана | 69.75% |
| Середнє значення | 70.98% |
| Мода | 70.98% |
| Видалення NaN значень | 82.70% |
| Заміна на кностанту | 71.60% |
| КНН | 73.34% |

**Таблиця 13** Результати Наївного Баєса з різними методами імпутації.

Рисунок 36 являється діаграмою, яка містить різні методи обробки пропущених значень і їхню точність з використанням методу машинного навчання Наївного Баєса. Таблиця 13 Даний метод має найкращі показники з поєднанням методу видалення пропущених значень, і найгіршу точність з методом медіани 69.75%.

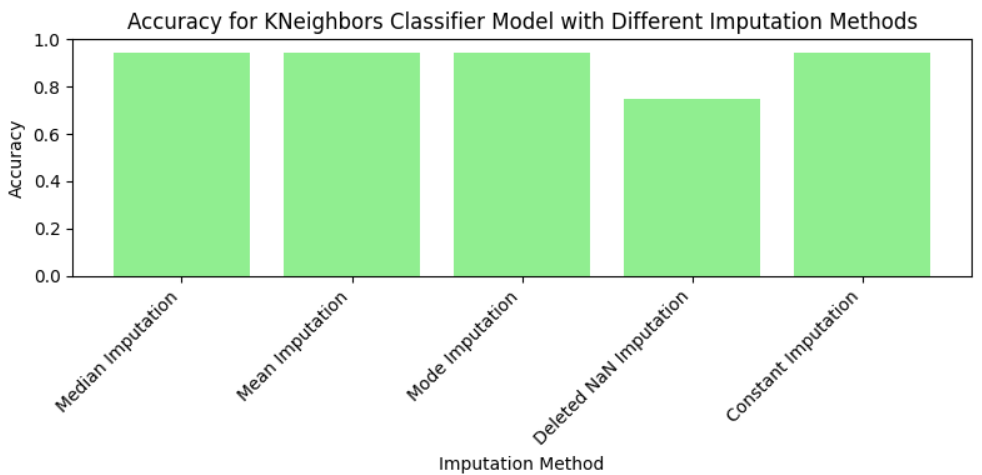
На Рисунках 32 – 36 і Таблицях 9 – 13 порівняно 6 методів оброблення пропущених значень з 5 методами машинного навчання. Найкраща модель машинного навчання для цього датасету є дерево рішень, у цьому наборі найефективніший результат має точність 100%. Найгірший метод машинного навчання для даного датасету є Наївний Баєс, який має найкращу точність 82% з видаленням пропущених даних, а найгіршу 69% з медіаною.

Найкраще поєднання методу і машинного навчання є дерево рішень з видаленням пропущених значень, точність комбінації досягає 100%. Найгірша комбінація машинного навчання і обробки пропущених значень є логістична регресія з медіаною і Наївний Баєс з медіаною, їхня точність 69%.

Метод КНН є ефективним у цьому датасеті, оскільки даний метод швидко виконувався, через те що у наборі даних містилося 1622 рядків.

**4.2 Опис і порівняння результатів використовуючи Covid-19 Dataset**

Порівняння усіх методів імпутації для певного методу машинного навчання.

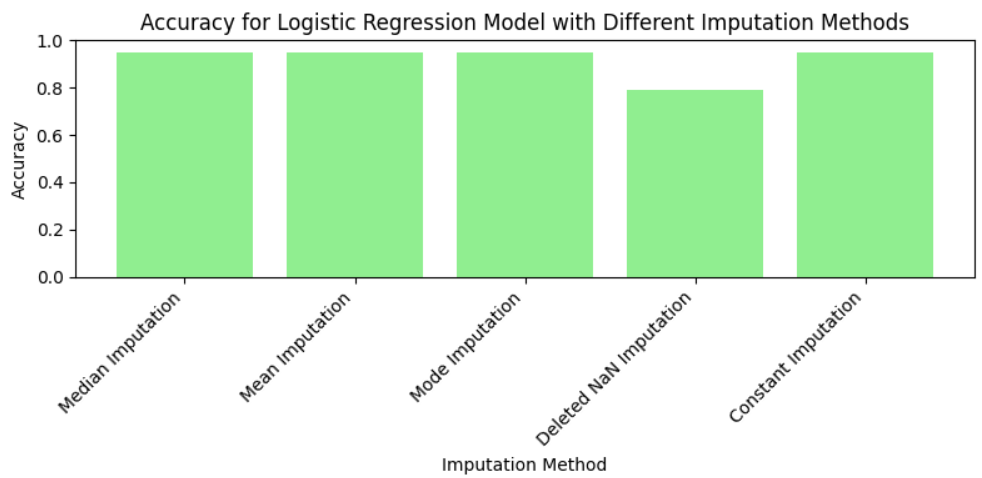


**Рисунок 37** Результати КНН з різними методами імпутації.

|  |  |
| --- | --- |
| Модель машинного навчання КНН | |
| Види імпутації | Точність тренування |
| Медіана | 94.25% |
| Середнє значення | 94.21% |
| Мода | 94.25% |
| Видалення NaN значень | 74.79% |
| Заміна на кностанту | 94.25% |

**Таблиця 14** Результати КНН з 5 методами імпутації.

Рисунок 37 є діаграмою, яка показує точність методу КНН з 5 методами імпутації. Рисунок 37 і Таблиця 14 показують точність даного методу при обробці різними методами імпутації. Метод КНН має доволі точні результати, а саме 4/5 методів імпутації мають точність 95%. Видалення пропущених значень мають точність 75%, що є найменшою точністю у цьому методі. Також даний метод обчислювався найповільніше зі всіх методів обробки, оскільки він створений для малої кількості даних. Повільніше виконувався в 40 разів порівнянні з іншими методами машинного навчання.

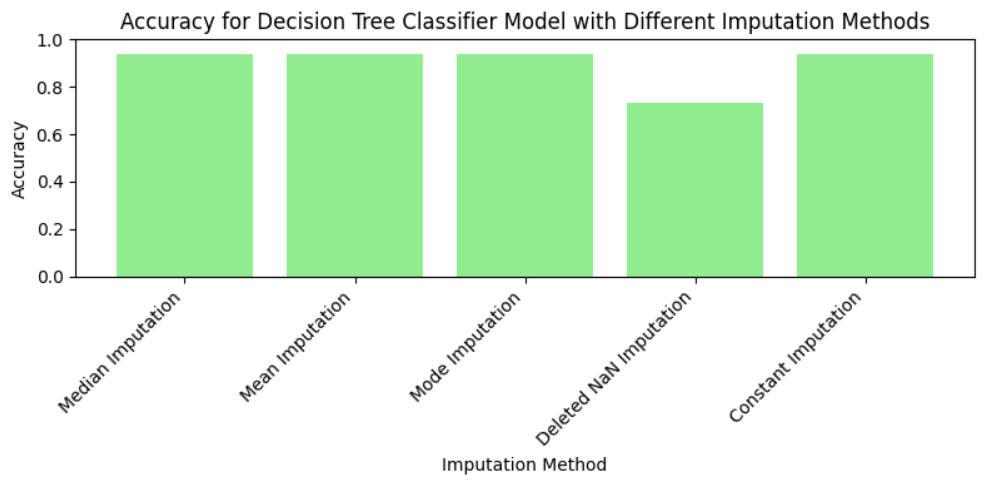


**Рисунок 38** Результати логістичної регресії з різними методами імпутації.

|  |  |
| --- | --- |
| Модель машинного навчання Логістична регресія | |
| Види імпутації | Точність тренування |
| Медіана | 94.82% |
| Середнє значення | 94.84% |
| Мода | 94.83% |
| Видалення NaN значень | 78.81% |
| Заміна на кностанту | 94.82% |

**Таблиця 15** Результати логістична регресія з 5 методами імпутації

Рисунок 38 є діаграмою, яка показує точність результатів логістичної регресії з різними видами обробки пропущених значень. Таблиця 15 показує детальніше інформацію про результати обробки. Найкращі результати є у заміни на константу, медіана, середнє значення, мода – 94% точності. Найгірші показники має метод видалення пропущених значень – 78.8%.

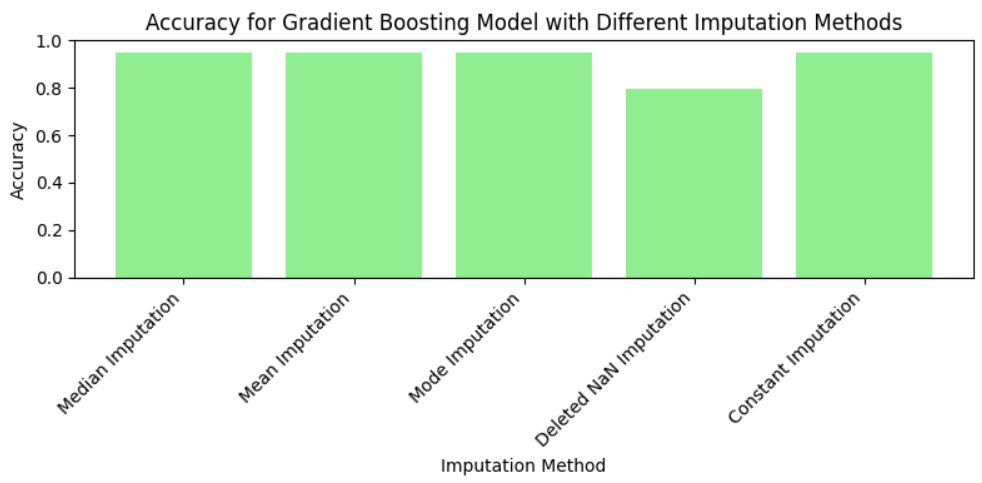


**Рисунок 39** Результати дерева рішень з різними методами імпутації.

|  |  |
| --- | --- |
| Модель машинного навчання Дерево рішень | |
| Види імпутації | Точність тренування |
| Медіана | 93.83% |
| Середнє значення | 93.92% |
| Мода | 93.85% |
| Видалення NaN значень | 73.48% |
| Заміна на кностанту | 93.84% |

**Таблиця 16** Результати дерева рішень з 5 методами імпутації.

Рисунок 39 показує точність кожного методу обробки пропущених значень використовуючи дерево рішень. Таблиця 16 надає точну інформацію щодо результатів точності методу дерева рішень. Найефективнішим методом обробки у даному методі є: медіана, середнє значення, мода, заміна на константу – їх точність досягає 93%. Метод середнього значення є на 0.1% є кращим інші методи. Найгірші показники є у методі видалені NaN значень – 73.48%, що є доволі низьким результатом.

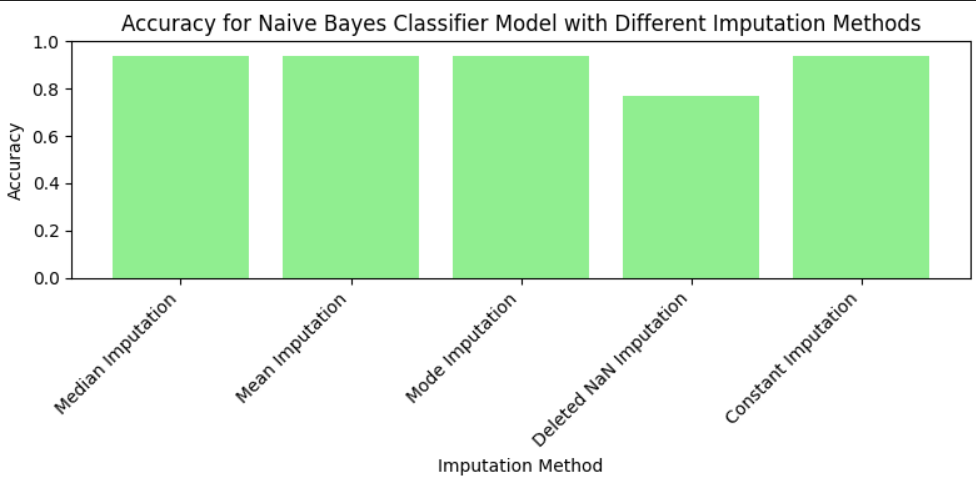


**Рисунок 40** Результати градієнтне прискорення з різними методами імпутації.

|  |  |
| --- | --- |
| Модель машинного навчання Градієнтне прискорення | |
| Види імпутації | Точність тренування |
| Медіана | 95.10% |
| Середнє значення | 95.15% |
| Мода | 95.09% |
| Видалення NaN значень | 79.76% |
| Заміна на кностанту | 95.08% |

**Таблиця 17** Результати градієнтне прискорення з 5 методами імпутації.

Рисунок 40 і Таблиця 17 надають інформацію щодо точності алгоритму машинного навчання, а саме градієнтного прискорення і його точність в залежності від різних методів обробки пропущених значень. Найгірший результат є виконаний за допомогою видалення пропущених значень, його точність досягає 79.76%. Усі інші методи мають точність 95%.



**Рисунок 41** Результати Наївного Баєса з різними методами імпутації.

|  |  |
| --- | --- |
| Модель машинного навчання КНН | |
| Види імпутації | Точність тренування |
| Медіана | 93.84% |
| Середнє значення | 93.88% |
| Мода | 93.84% |
| Видалення NaN значень | 76.89% |
| Заміна на кностанту | 93.84% |

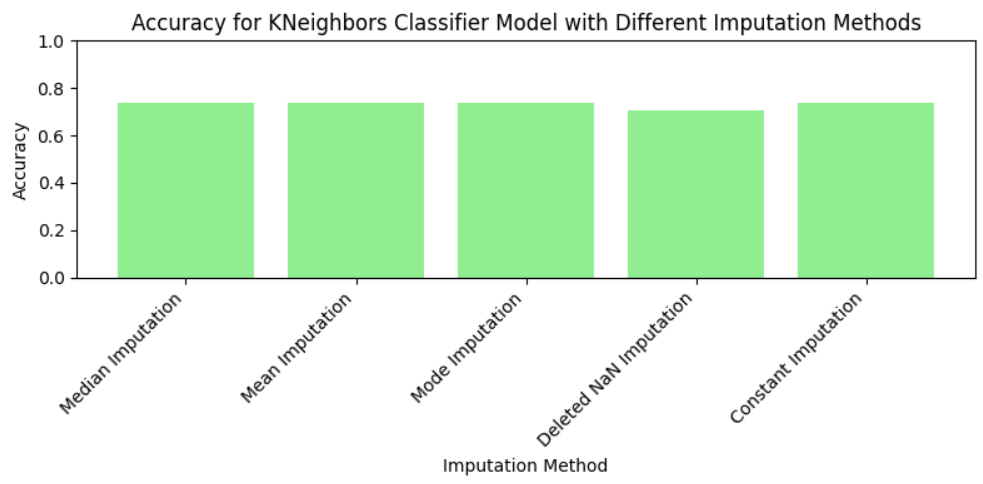
**Таблиця 18** Результати Наївного Баєса з 5 методами імпутації.

Рисунок 41 і Таблиця 18 показують точність е методів імпутації використовуючи метод Наївного Баєса. Найгірші результати також спостерігаються у метода видалення пропущених значень – 76.89%. Інші моделі мають точність 93%. Найкращий метод оброблення пропущених значень у Наївному Баєсі є середнє значення, цей метод кращий на 0.05% від другої з моделі, має точність 93.88%.

У цьому датасеті були проаналізовані Рисунки 37 – 41 і Таблиці 14 – 18. У цьому датасеті всі моделі машинного навчання мають приблизно одинакову точність 93-95%. Найкраща модель машинного навчання для датасету Covid-19 є градієнтне прискорення, його середня точність 95%. Найгірша модель машинного навчання у даного датасету є Наївний Баєс і дерево рішень, які мають точність 93.5%. Найгірший метод оброблення пропущених значень у всіх 5 методів машинного навчання є видалення пропущених значень. Його точність досягає у всіх методах машинного навчання – 73-79%.

**4.3 Опис і порівняння результатів використовуючи Diabetes Health Indicators Dataset**

У цьому датасеті кількість пропущених даних досягає 10%. Порівняння усіх методів імпутації для певного методу машинного навчання.

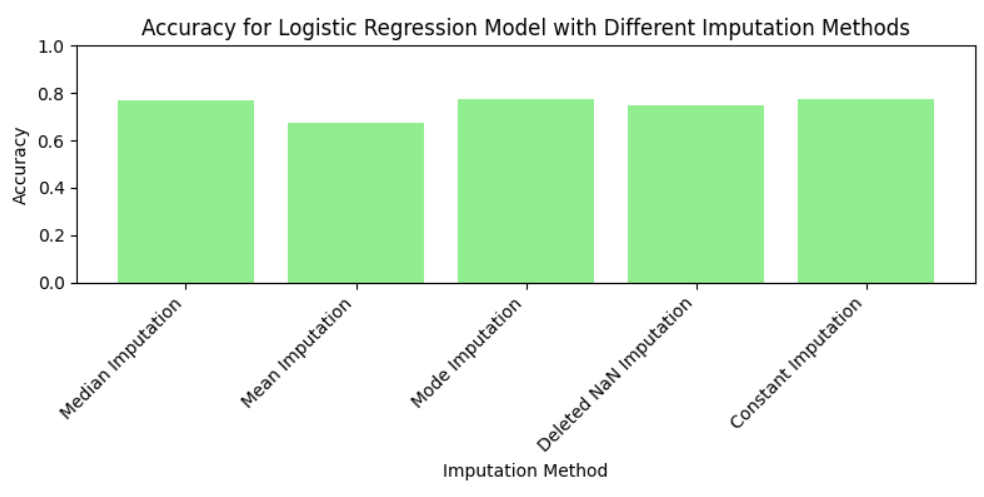


**Рисунок 42** Діаграма точності для методу КНН

|  |  |
| --- | --- |
| Модель машинного навчання КНН | |
| Види імпутації | Точність тренування |
| Медіана | 74.39% |
| Середнє значення | 74.39% |
| Мода | 74.95% |
| Видалення NaN значень | 70.71% |
| Заміна на кностанту | 74.02% |

**Таблиця 19** Результати точності методу КНН з 5-тьма методами імпутації

Рисунок 42 показує діаграму, на якій здійснюється порівняння результатів методів машинного навчання КНН з 6 методами обробки пропущених значень. Таблиця 19 показує точність кожного методу оброблення пропущених значень. Рисунок 42 і Таблиця 19 показують найефективніший метод обробки пропущених значень для методу КНН у цьому датасеті є мода: 74.95%. Найменш ефективним методом обробки пропущених значень є метод видаленням пропущених значень – 70.71%.

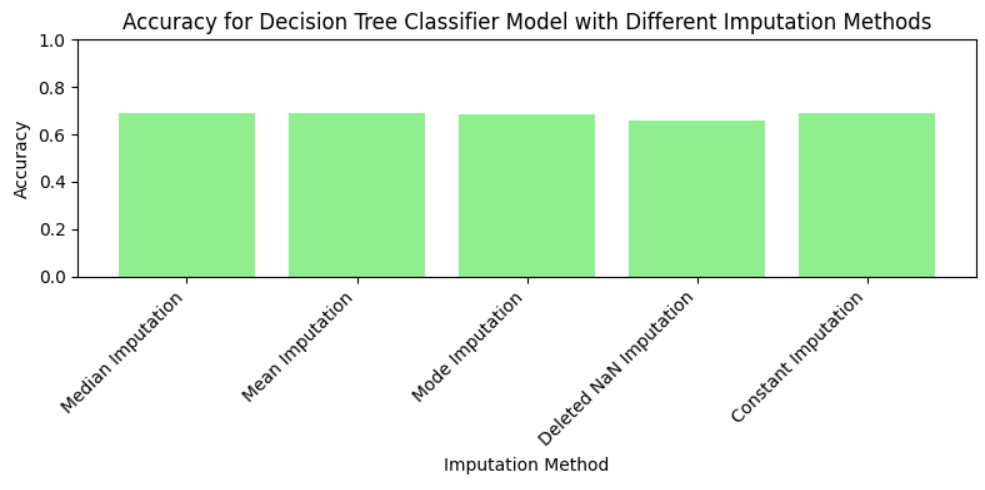
****

**Рисунок 43** Діаграма точності логістичної регресії

|  |  |
| --- | --- |
| Модель машинного навчання Логістична регресія | |
| Види імпутації | Точність тренування |
| Медіана | 78.96% |
| Середнє значення | 67.43% |
| Мода | 78.34% |
| Видалення NaN значень | 74.86% |
| Заміна на кностанту | 77.58% |

**Таблиця 20** Результати точності логістичної регресії з 5-тьма методами імпутації

Рисунок 43 є діаграмою, яка містить порівняння усіх методів оброблення пропущених значення використовуючи логістичну регресію. Рисунок 43 і Таблиця 20 показують, що найкраща точність досягається при методі імпутації: медіана, а саме 78.96% точності, найгірша точність спостерігається у середнього значення – 67.43%.

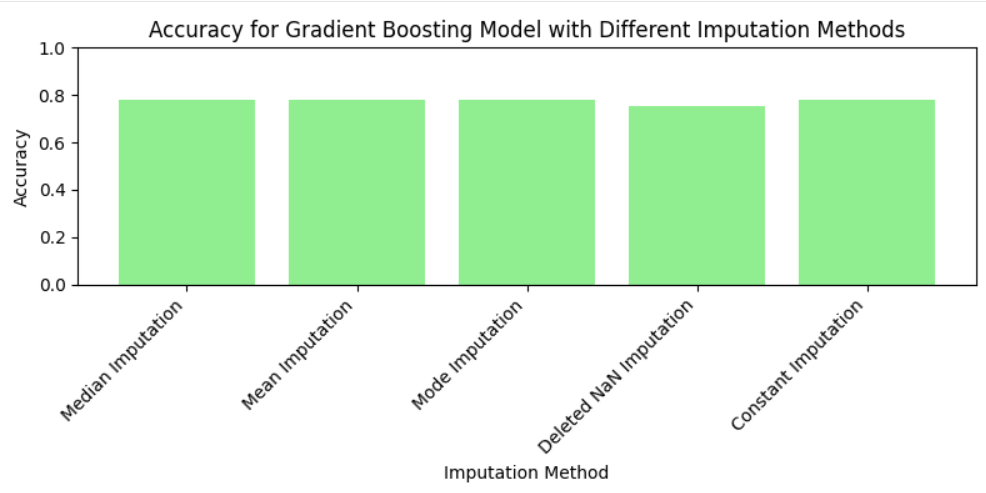
****

**Рисунок 44** Діаграма точності для дерева рішень

|  |  |
| --- | --- |
| Модель машинного навчання Дерево рішень | |
| Види імпутації | Точність тренування |
| Медіана | 69.03% |
| Середнє значення | 69.25% |
| Мода | 68.77% |
| Видалення NaN значень | 65.97% |
| Заміна на кностанту | 68.68% |

**Таблиця 21** Результати точності дерева рішень зі п’ятьма методами імпутації

У рисунку 44 показано діаграми для метода машинного навчання – дерево рішень, яке оброблене 5 видами імпутації. Таблиця 21 демонструє точність результатів оброблення датасету різними методами імпутації. Через те що види імпутації є різними, то набір даних має різні точності, найкраща з яких середнє значення – 69.25%, а найменш ефективним – видалення пропущених значень – 65.97%.

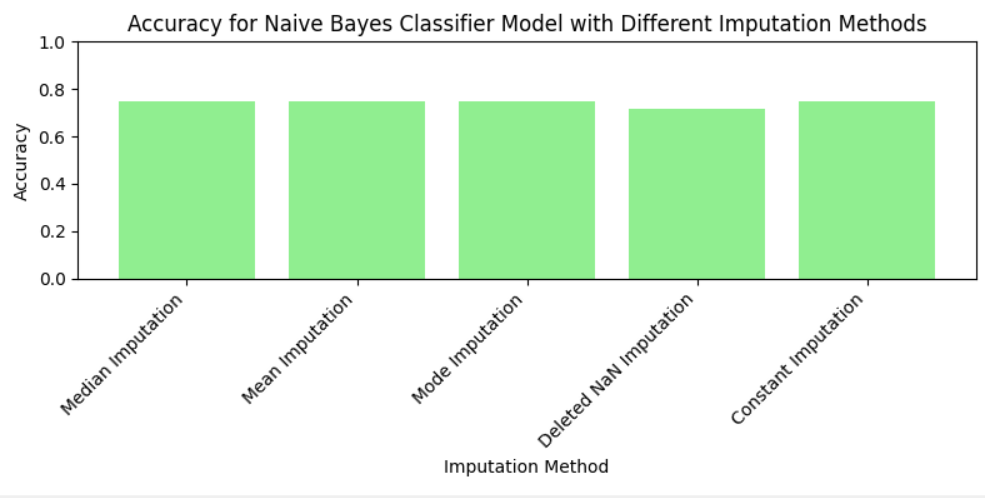
****

**Рисунок 45** Діаграма точності для градієнтного прискорення

|  |  |
| --- | --- |
| Модель машинного навчання Градієнтне прискорення | |
| Види імпутації | Точність тренування |
| Медіана | 77.96% |
| Середнє значення | 78.18% |
| Мода | 77.97% |
| Видалення NaN значень | 75.13% |
| Заміна на кностанту | 78.02% |

**Таблиця 22** Результати моделі машинного навчання – градієнтного прискорення 5 методами імпутації.

Рисунок 45 і Таблиця 22 показують точність моделі градієнтного прискорення з 5 методами обробки пропущених значень. Таблиця 22 містить детальніші результати. Найкращий метод є медіана, його точність є 78.18%, а найгіршим методом являється видаленням пропущених значень – 75.13%.



**Рисунок 46** Діаграма точності для Наївного Баєса

|  |  |
| --- | --- |
| Модель машинного навчання Наївний Баєс | |
| Види імпутації | Точність тренування |
| Медіана | 74.94% |
| Середнє значення | 74.94% |
| Мода | 74.94% |
| Видалення NaN значень | 74.90% |
| Заміна на кностанту | 71.94% |

**Таблиця 23** Результати Наївного Баєса з різними методами імпутації.

Рисунок 46 і Таблиця 23 показують точність методу машинного навчання, а саме Наївного Баєса з різними методами імпутації. Найгірші показники має метод заміни а константу – 71.94%. Інші методи мають одинакову точність – приблизно 74.94%.

На рисунках 32 – 46 і Таблицях 9 – 23 було порівняно різні методи імпутації з 5 методами машинного навчання. 3 набори даних містили різну кількість даних, їхня кількість суттєво відрізняється. Найгіршим методом для обробки значень у великих наборах даних виявився КНН, цей метод є часо-затратним і зі збільшенням кількості даних, його обчислення сповільнювалося. Цей метод підійшов тільки для першого датасету. Для останнього датасету він не міг обчислитися протягом 10 годин, через довгу класифікацію.

Метод видалення пропущених значень добре показав себе у першому датасеті при наборі даних 1622. Його найкраща точність досягнула 100% з комбінацією дерева рішень. У більших наборах даних, другий і третій датасет, видалення пропущених значень мало найгірші результати при різних моделях машинного навчання.

При великій обробці даних найкраще себе показав метод обробки мода і медіана. Використовуючи 5 методів машинного навчання, результати найкращої ефективності мінялися, то мода, то медіана.

Нажаль немає найкращого методу оброблення пропущених значень у великих обсягах даних для усіх наборів даних. Все залежить від набору даних, чи він є збалансованим, методів машинного навчання. Але з найкращого методу перетворюється в найгірший метод оброблення пропущених значень – видалення пропущених значень. Цей метод надає гірші результати зі збільшенням кількості даних у датасеті.

**ВИСНОВКИ**

У курсовій роботі було детально розглянуто методи оброблення пропущених значень і визначення невизначеності у великих наборах даних. Дослідивши джерела і їх порівнявши, було реалізовано такі моделі машинного навчання: Наївний Баєс, КНН, Логістична регресія, Прискорення градієнту, Дерева рішень з поєднанням методами обробки пропущених значень: Видалення пропущених значень, заміна на константу, медіана, середнє значення, мода. В загальному було побудовано 75 різних комбінацій – методів машинного навчання з методами обробки пропущених значень і поєднанням 3 датасетів. Було детально проаналізовано набори даних у яких прогнозували цільову колонку. Датасети відрізнялися розміром даних, кількістю колонок. На 1 датасеті було випробовувано метод КНН, як заміна пропущених значень. Даний метод стає менш ефективний при збільшені кількості даних, в 3 датасеті, він обчислювався 10 годин, коли інші методи затрачають хвилини. З’ясували що найкраще поєднання для першого набору даних є видалення пропущених значень з деревом рішень. І найгірше показали результати мода і КНН з поєднанням Логістичної регресії і Наївного Баєса. Але у наборах даних №2 і №3, які є в 70 і 600 разів більшими, ніж набір даних №1 – видалення пропущених значень показало гірші результати у всіх методах машинного навчання. Найкраще поєднання – градієнтне прискорення з середнім значенням у другому і третьому наборі даних, його точність 95.15%, 78.18% відповідно.

Порівнявши результати виконання програми з 5 методами машинного навчання і 5 методами обробки пропущених значень, дійшли до висновку, що немає єдиного найкращого методу оброблення пропущених значень у наборі даних. Усі методи можуть дати кращу або гіршу точність в залежності від набору даних. Але не слід використовувати метод видалення пропущених значень через його гіршу точність і метод КНН, через його довге обрахування порівняно зі всіма методами пропущених значень.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Little, Roderick JA, and Donald B. Rubin. "Statistical analysis with missing data." John Wiley & Sons, 2019.
2. Gelman, Andrew, et al. "Bayesian data analysis." CRC Press, 2021.
3. Rubin, Donald B. "Multiple imputation for nonresponse in surveys." John Wiley & Sons, 2004.
4. Van Buuren, Stef. "Flexible imputation of missing data." CRC Press, 2012.
5. Carpenter, James R., and Michael G. Kenward. "Missing data in clinical trials: a practical guide." CRC Press, 2020.
6. Enders, Craig K. "Applied missing data analysis." Guilford Press, 2010.
7. Graham, J. W. (2012). Missing data: Analysis and design. Springer.
8. Numpy: <https://numpy.org/>
9. Scikit-learn: <https://scikit-learn.org/stable/>
10. Logistic regression: <https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression>
11. Decisions tree: <https://scikit-learn.org/stable/modules/tree.html>
12. Naïve Bayes: <https://scikit-learn.org/stable/modules/naive_bayes.html>
13. KNN: <https://scikit-learn.org/stable/modules/neighbors.html#classification>
14. Gradient boosting: <https://scikit-learn.org/stable/modules/ensemble.html#gradient-boosting>
15. Gradient boosting: <https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab>
16. IBM: <https://www.ibm.com/us-en>
17. Logistic regression: <https://towardsdatascience.com/a-visual-understanding-of-logistic-regression-2e6733844397>
18. Decision tree: <https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052>

**ДОДАТКИ**

**Додаток № 1**

%pip install numpy pandas scikit-learn seaborn matplotlib

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

# Створюємо новий стовпець з віковими групами кожні 5 років

df['AGE\_GROUP'] = pd.cut(df['AGE'], bins=range(0, 101, 5), right=False)

# Групуємо дані за віковими групами та підраховуємо кількість

age\_group\_counts = df['AGE\_GROUP'].value\_counts().sort\_index()

# Побудова стовпчастої діаграми

plt.figure(figsize=(10, 6))

age\_group\_counts.plot(kind='bar', color='skyblue')

plt.xlabel('Вікові групи')

plt.ylabel('Кількість пацієнтів')

plt.title('Розподіл пацієнтів за віковими групами (кожні 5 років)')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

# Фільтрація даних за віком та статтею

age\_sex\_df = df[['AGE', 'SEX']]

# Групування за віком та статтею та підрахунок кількості

age\_sex\_counts = age\_sex\_df.groupby(['AGE', 'SEX']).size().unstack()

# Побудова кругової діаграми для виведення за статтю

plt.figure(figsize=(6, 6))

plt.pie(age\_sex\_counts.sum(axis=0), labels=age\_sex\_counts.columns, autopct='%1.1f%%')

plt.title('Distribution of Patients by Sex')

plt.show()

# Фільтруємо дані для померлих чоловіків і жінок

dead\_male\_count = df[(df['SEX'] == 1) & (df['DATE\_DIED'] != '9999-99-99')]['USMER'].count()

dead\_female\_count = df[(df['SEX'] == 2) & (df['DATE\_DIED'] != '9999-99-99')]['USMER'].count()

# Побудова діаграми

plt.figure(figsize=(6, 6))

plt.bar(['Male', 'Female'], [dead\_male\_count, dead\_female\_count], color=['lightblue', 'lightcoral'])

plt.xlabel('Стать')

plt.ylabel('Кількість померлих')

plt.title('Кількість померлих від коронавірусу за статтю')

plt.show()

covid = pd.read\_csv("Dataset/Covid Data.csv")

print(covid)

print(covid.columns)

covid.info()

null\_counts\_1 = covid.isnull().sum()

print(null\_counts\_1)

for column\_name in covid.columns:

    count\_97\_values = (covid[column\_name] == 97).sum()

    print(f"Кількість значень 97 у стовпці '{column\_name}': {count\_97\_values}")

for column\_name in covid.columns:

    count\_99\_values = (covid[column\_name] == 99).sum()

    print(f"Кількість значень 99 у стовпці '{column\_name}': {count\_99\_values}")

covid.DATE\_DIED.value\_counts()

covid.info()

df = pd.read\_csv("Dataset/Covid Data.csv")

df['DATE\_DIED'] = df['DATE\_DIED'].replace(['9999-99-99', '97', '99'], np.nan)

# Заміна неправильних значень у стовпці 'PREGNANT' і 'INTUBED' на NaN

df['PREGNANT'] = df['PREGNANT'].replace([97, 99], np.nan)

df['INTUBED'] = df['INTUBED'].replace([97, 99], np.nan)

df['ICU'] = df['ICU'].replace([97, 99], np.nan)

# Вибірки для обрахунку

columns\_to\_calculate = ['USMER', 'MEDICAL\_UNIT', 'SEX', 'PATIENT\_TYPE', 'INTUBED', 'PNEUMONIA', 'AGE',

                        'PREGNANT', 'DIABETES', 'COPD', 'ASTHMA', 'INMSUPR', 'HIPERTENSION', 'OTHER\_DISEASE',

                        'CARDIOVASCULAR', 'OBESITY', 'RENAL\_CHRONIC', 'TOBACCO', 'CLASIFFICATION\_FINAL', 'ICU']

# Функція для обрахунку статистики

def calculate\_statistics(column\_name):

    mean\_value = df[column\_name].mean()

    mode\_value = df[column\_name].mode().values[0] if not df[column\_name].mode().empty else np.nan

    median\_value = df[column\_name].median()

    return mean\_value, mode\_value, median\_value

# Створення DataFrame для виведення результатів

statistics\_df = pd.DataFrame(columns=['mean', 'mode', '50%'])

# Додавання результатів обрахунків до DataFrame

for column in columns\_to\_calculate:

    mean\_val, mode\_val, median\_val = calculate\_statistics(column)

    statistics\_df.loc[column] = [mean\_val, mode\_val, median\_val]

# Виведення результатів у вигляді таблички

print("Summary Statistics:")

print(statistics\_df)

covid['dead'] = covid['DATE\_DIED'].apply(lambda x: 0 if x == '9999-99-99' else 1)

covid.drop('DATE\_DIED', axis=1, inplace=True)

print(covid.columns)

counts = covid['dead'].value\_counts()

print(counts)

from sklearn.impute import KNNImputer

covid.replace({99: np.nan, 97: np.nan}, inplace=True)

covid\_median = covid.fillna(covid.median())

covid\_mode = covid.apply(lambda x: x.fillna(x.mode().iloc[0]))

covid\_mean = covid.fillna(covid.mean())

covid\_dropped = covid.dropna()

covid\_constant = covid.fillna(2)

# Розмір тренувальної та тестової частини

train\_size = X\_train\_median.shape[0]

test\_size = X\_test\_median.shape[0]

# Побудова діаграми

plt.figure(figsize=(6, 4))

plt.bar(['Train', 'Test'], [train\_size, test\_size], color=['lightblue', 'lightcoral'])

plt.xlabel('Dataset Part')

plt.ylabel('Number of Data Points')

plt.title('Number of Data Points in Training and Test Set')

plt.show()

from sklearn.pipeline import Pipeline

from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, mean\_absolute\_error, mean\_squared\_error

models = [

    ('Logistic Regression', LogisticRegression(max\_iter=1000, solver='liblinear', random\_state=42)),

    ('KNeighbors Classifier', KNeighborsClassifier()),

    ('Decision Tree Classifier', DecisionTreeClassifier(random\_state=42)),

    ('Gradient Boosting', GradientBoostingClassifier(random\_state=42)),

    ('Naye base Classifier', GaussianNB())

]

datasets = [

    ('Median Imputation', X\_train\_median, X\_test\_median, y\_train\_median, y\_test\_median),

    ('Mean Imputation', X\_train\_mean, X\_test\_mean, y\_train\_mean, y\_test\_mean),

    ('Mode Imputation', X\_train\_mode, X\_test\_mode, y\_train\_mode, y\_test\_mode),

    ('Deleted NaN Imputation', X\_train\_deleted\_nan, X\_test\_deleted\_nan, y\_train\_deleted\_nan, y\_test\_deleted\_nan),

    ('Constant Imputation', X\_train\_constant, X\_test\_constant, y\_train\_constant, y\_test\_constant)

]

knn\_classifier = KNeighborsClassifier()

logistic\_regression = LogisticRegression(max\_iter=1000, solver='liblinear', random\_state=42)

decision\_tree = DecisionTreeClassifier(random\_state=42)

gradient\_boosting = GradientBoostingClassifier(random\_state=42)

naive\_bayes = GaussianNB()

# List of models and their corresponding estimator objects

models = [

    ('KNeighbors Classifier', knn\_classifier),

    ('Logistic Regression', logistic\_regression),

    ('Decision Tree Classifier', decision\_tree),

    ('Gradient Boosting', gradient\_boosting),

    ('Naive Bayes Classifier', naive\_bayes)

]

# Code for imputation and splitting data goes here

# Define a function to plot accuracy for each model and imputation method

def plot\_accuracy\_for\_ml\_method(model\_name, model, X\_train, X\_test, y\_train, y\_test, imputation\_datasets):

    dataset\_names = []

    accuracy\_values = []

    for dataset\_name, X\_train\_imp, X\_test\_imp, y\_train\_imp, y\_test\_imp in imputation\_datasets:

        pipeline = Pipeline([('imputation', 'passthrough'), ('model', model)])

        pipeline.fit(X\_train\_imp, y\_train\_imp)

        y\_pred = pipeline.predict(X\_test\_imp)

        accuracy = accuracy\_score(y\_test\_imp, y\_pred)

        dataset\_names.append(dataset\_name)

        accuracy\_values.append(accuracy)

        print(f"Model: {model\_name}")

        print("Dataset Used: ", dataset\_name)

        print("Test Accuracy: ", accuracy)

    plt.figure(figsize=(8, 4))

    plt.bar(dataset\_names, accuracy\_values, color='lightgreen')

    plt.xlabel('Imputation Method')

    plt.ylabel('Accuracy')

    plt.title(f'Accuracy for {model\_name} Model with Different Imputation Methods')

    plt.xticks(rotation=45, ha='right')

    plt.ylim(0, 1)

    plt.tight\_layout()

    plt.show()

# Call the function for each model

for model\_name, model in models:

    plot\_accuracy\_for\_ml\_method(model\_name, model, X\_train, X\_test, y\_train, y\_test, datasets)

best\_model = None

best\_accuracy = 0.0

for dataset\_name, X\_train, X\_test, y\_train, y\_test in datasets:

    print(f"Dataset: {dataset\_name}")

    print("---------------------")

    for model\_name, model in models:

        pipeline = Pipeline([('model', model)])

        scores = cross\_val\_score(pipeline, X\_train, y\_train, cv=5)

        pipeline.fit(X\_train, y\_train)

        y\_pred = pipeline.predict(X\_test)

        accuracy = accuracy\_score(y\_test, y\_pred)

        print(f"Model: {model\_name}")

        print("Dataset Used: ", dataset\_name)

        print("Test Accuracy: ", accuracy)

        print()

        if accuracy > best\_accuracy:

            best\_accuracy = accuracy

            best\_model = (dataset\_name, model\_name, pipeline)

print("Best Model: ", best\_model)

**Додаток № 2**

%pip install numpy, pandas, matplotlib, sklearn

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

plt.switch\_backend('TkAgg')

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import mean\_squared\_error

from sklearn.tree import plot\_tree

diabet\_dataset = pd.read\_csv('Dataset/diabetes\_binary\_5050split\_health\_indicators\_BRFSS2015.csv')

diabet\_dataset.info()

nan\_count = int(diabet\_dataset.shape[0] \* 0.1)

# Вибір випадкових індексів для заповнення NaN

nan\_indices = np.random.choice(diabet\_dataset.index, nan\_count, replace=False)

# Заповнення вибраних рядків усіма стовпцями NaN

diabet\_dataset.loc[nan\_indices, :] = np.nan

# Вивід результату

print("Dataset with missing values (10% NaN and 90% not null):")

print("Dataset with missing values:")

diabet\_dataset.info()

# Розбиття віку на інтервали до 20 років по 5 років

age\_intervals = [(i, i + 5) for i in range(0, 20, 5)]

age\_group\_counts = []

# Підрахунок кількості людей для кожного інтервалу

for interval in age\_intervals:

    count = diabet\_dataset[(diabet\_dataset['Age'] >= interval[0]) & (diabet\_dataset['Age'] < interval[1])].shape[0]

    age\_group\_counts.append(count)

# Побудова графіка кількості людей по інтервалам до 20 років

plt.figure(figsize=(8, 5))

plt.bar([f"{i[0]}-{i[1]}" for i in age\_intervals], age\_group\_counts, color='skyblue')

plt.xlabel('Age Intervals')

plt.ylabel('Number of People')

plt.title('Number of People by Age Intervals (0-20 years)')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

sex\_counts = diabet\_dataset['Sex'].value\_counts()

# Побудова стовпчатої діаграми

plt.figure(figsize=(6, 4))

sex\_counts.plot(kind='bar', color=['lightcoral', 'skyblue'])

plt.xlabel('Sex (1 - Female, 0 - Male)')

plt.ylabel('Number of People')

plt.title('Distribution of People by Sex')

plt.xticks(rotation=0)

plt.tight\_layout()

plt.show()

# Виведення кількості

print("Кількість Жінок (Sex = 0):", sex\_counts[1])

print("Кількість Чоловіків (Sex = 1):", sex\_counts[0])

diabet\_dataset['Diabetes\_binary'] = diabet\_dataset['Diabetes\_binary'].astype('category')

# Побудова стовпчастої діаграми для відображення кількості людей з діабетом

plt.figure(figsize=(8, 6))

diabet\_dataset['Diabetes\_binary'].value\_counts().plot(kind='bar', color='skyblue')

plt.xlabel('Diabetes Status')

plt.ylabel('Count')

plt.title('Number of People with Diabetes')

plt.xticks(rotation=0)

plt.tight\_layout()

plt.show()

# Побудова кругової діаграми для відображення кількості людей з діабетом

plt.figure(figsize=(6, 6))

diabet\_dataset['Diabetes\_binary'].value\_counts().plot(kind='pie', autopct='%1.1f%%', colors=['lightblue', 'lightgreen', 'lightcoral'])

plt.title('Percentage of People with Diabetes')

plt.ylabel('')

plt.tight\_layout()

plt.show()

numeric\_columns = diabet\_dataset.select\_dtypes(include=np.number).columns.tolist()

# Функція для обрахунку медіани, середнього значення і моди

def calculate\_statistics(column\_name):

    mean\_value = diabet\_dataset[column\_name].mean()

    median\_value = diabet\_dataset[column\_name].median()

    mode\_value = diabet\_dataset[column\_name].mode().values[0] if not diabet\_dataset[column\_name].mode().empty else np.nan

    return mean\_value, median\_value, mode\_value

# Створення DataFrame для збереження результатів обрахунків

statistics\_df = pd.DataFrame(columns=['mean', 'median', 'mode'])

# Обрахунок статистики для кожного числового стовпця

for column in numeric\_columns:

    mean\_val, median\_val, mode\_val = calculate\_statistics(column)

    statistics\_df.loc[column] = [mean\_val, median\_val, mode\_val]

# Виведення результатів

print("Summary Statistics:")

print(statistics\_df)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(diabet\_dataset[['Diabetes\_binary']], diabet\_dataset['Sex'], test\_size=0.3, random\_state=42)

# Підрахунок кількості пацієнтів з діабетом у тренувальному та тестовому наборах

train\_diabetes\_counts = X\_train[X\_train['Diabetes\_binary'] == 1].shape[0]

test\_diabetes\_counts = X\_test[X\_test['Diabetes\_binary'] == 1].shape[0]

# Побудова діаграми

fig, ax = plt.subplots(figsize=(8, 5))

ax.bar(['Train Diabetes'], [train\_diabetes\_counts], color='skyblue', label='Training Set')

ax.bar(['Test Diabetes'], [test\_diabetes\_counts], color='lightcoral', label='Testing Set')

ax.set\_xlabel('Diabetes\_binary')

ax.set\_ylabel('Number of Patients')

ax.set\_title('Distribution of Diabetes\_binary (1) in Training and Testing Sets')

ax.legend()

plt.show()

from sklearn.preprocessing import LabelEncoder

# Заміна 99 та 97 на np.nan

diabet\_dataset.replace({99: np.nan, 97: np.nan}, inplace=True)

# Конвертуємо всі категоріальні стовпці в числові

label\_encoders = {}

for column in diabet\_dataset.select\_dtypes(include=['category', 'object']).columns:

    le = LabelEncoder()

    diabet\_dataset[column] = le.fit\_transform(diabet\_dataset[column].astype(str))

    label\_encoders[column] = le

# Заповнення відсутніх значень медіаною

diabet\_dataset\_median = diabet\_dataset.fillna(diabet\_dataset.median())

# Заповнення відсутніх значень модою

diabet\_dataset\_mode = diabet\_dataset.apply(lambda x: x.fillna(x.mode().iloc[0]))

# Заповнення відсутніх значень середнім значенням

diabet\_dataset\_mean = diabet\_dataset.fillna(diabet\_dataset.mean())

# Видалення рядків з відсутніми значеннями

diabet\_dataset\_dropped = diabet\_dataset.dropna()

# Заповнення відсутніх значень константою 2

diabet\_dataset\_constant = diabet\_dataset.fillna(2)

diabet\_dataset\_knn = diabet\_dataset.copy()

# Поділ даних на ознаки (X) і цільовий стовпець (y) для різних варіантів заповнення

X\_median = diabet\_dataset\_median.drop('Diabetes\_binary', axis=1)

y\_median = diabet\_dataset\_median['Diabetes\_binary']

X\_mean = diabet\_dataset\_mean.drop('Diabetes\_binary', axis=1)

y\_mean = diabet\_dataset\_mean['Diabetes\_binary']

X\_mode = diabet\_dataset\_mode.drop('Diabetes\_binary', axis=1)

y\_mode = diabet\_dataset\_mode['Diabetes\_binary']

X\_deleted\_nan = diabet\_dataset\_dropped.drop('Diabetes\_binary', axis=1)

y\_deleted\_nan = diabet\_dataset\_dropped['Diabetes\_binary']

X\_constant = diabet\_dataset\_constant.drop('Diabetes\_binary', axis=1)

y\_constant = diabet\_dataset\_constant['Diabetes\_binary']

# Поділ даних на навчальні і тестові набори

X\_train\_median, X\_test\_median, y\_train\_median, y\_test\_median = train\_test\_split(X\_median, y\_median, test\_size=0.3, random\_state=42)

X\_train\_mean, X\_test\_mean, y\_train\_mean, y\_test\_mean = train\_test\_split(X\_mean, y\_mean, test\_size=0.3, random\_state=42)

X\_train\_mode, X\_test\_mode, y\_train\_mode, y\_test\_mode = train\_test\_split(X\_mode, y\_mode, test\_size=0.3, random\_state=42)

X\_train\_deleted\_nan, X\_test\_deleted\_nan, y\_train\_deleted\_nan, y\_test\_deleted\_nan = train\_test\_split(X\_deleted\_nan, y\_deleted\_nan, test\_size=0.3, random\_state=42)

X\_train\_constant, X\_test\_constant, y\_train\_constant, y\_test\_constant = train\_test\_split(X\_constant, y\_constant, test\_size=0.3, random\_state=42)

# Якщо ви маєте ще один набір даних для KNN, наприклад, knn\_dataset, то змініть також його назву

# knn\_dataset = ...

X\_knn = diabet\_dataset\_knn.drop('Diabetes\_binary', axis=1)

y\_knn = diabet\_dataset\_knn['Diabetes\_binary']

X\_train\_knn, X\_test\_knn, y\_train\_knn, y\_test\_knn = train\_test\_split(X\_knn, y\_knn, test\_size=0.3, random\_state=42)

from sklearn.pipeline import Pipeline

from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, mean\_absolute\_error, mean\_squared\_error

models = [

    ('Logistic Regression', LogisticRegression(max\_iter=1000, solver='liblinear', random\_state=42)),

    ('KNeighbors Classifier', KNeighborsClassifier()),

    ('Decision Tree Classifier', DecisionTreeClassifier(random\_state=42)),

    ('Gradient Boosting', GradientBoostingClassifier(random\_state=42)),

    ('Naye base Classifier', GaussianNB())

]

datasets = [

    ('Median Imputation', X\_train\_median, X\_test\_median, y\_train\_median, y\_test\_median),

    ('Mean Imputation', X\_train\_mean, X\_test\_mean, y\_train\_mean, y\_test\_mean),

    ('Mode Imputation', X\_train\_mode, X\_test\_mode, y\_train\_mode, y\_test\_mode),

    ('Deleted NaN Imputation', X\_train\_deleted\_nan, X\_test\_deleted\_nan, y\_train\_deleted\_nan, y\_test\_deleted\_nan),

    ('Constant Imputation', X\_train\_constant, X\_test\_constant, y\_train\_constant, y\_test\_constant)

]

knn\_classifier = KNeighborsClassifier()

logistic\_regression = LogisticRegression(max\_iter=1000, solver='liblinear', random\_state=42)

decision\_tree = DecisionTreeClassifier(random\_state=42)

gradient\_boosting = GradientBoostingClassifier(random\_state=42)

naive\_bayes = GaussianNB()

# List of models and their corresponding estimator objects

models = [

    ('Naive Bayes Classifier', naive\_bayes)

]

# Code for imputation and splitting data goes here

# Define a function to plot accuracy for each model and imputation m

def plot\_accuracy\_for\_ml\_method(model\_name, model, X\_train, X\_test, y\_train, y\_test, imputation\_datasets):

    dataset\_names = []

    accuracy\_values = []

    print(f"Model: {model\_name}")

    for dataset\_name, X\_train\_imp, X\_test\_imp, y\_train\_imp, y\_test\_imp in imputation\_datasets:

        pipeline = Pipeline([('imputation', 'passthrough'), ('model', model)])

        pipeline.fit(X\_train\_imp, y\_train\_imp)

        y\_pred = pipeline.predict(X\_test\_imp)

        accuracy = accuracy\_score(y\_test\_imp, y\_pred)

        dataset\_names.append(dataset\_name)

        accuracy\_values.append(accuracy)

        print("Dataset Used: ", dataset\_name)

        print("Test Accuracy: ", accuracy)

    plt.figure(figsize=(8, 4))

    plt.bar(dataset\_names, accuracy\_values, color='lightgreen')

    plt.xlabel('Imputation Method')

    plt.ylabel('Accuracy')

    plt.title(f'Accuracy for {model\_name} Model with Different Imputation Methods')

    plt.xticks(rotation=45, ha='right')

    plt.ylim(0, 1)

    plt.tight\_layout()

    plt.show()

# Call the function for each model

for model\_name, model in models:

    plot\_accuracy\_for\_ml\_method(model\_name, model, X\_train, X\_test, y\_train, y\_test, datasets)

best\_model = None

best\_accuracy = 0.0

for dataset\_name, X\_train, X\_test, y\_train, y\_test in datasets:

    print(f"Dataset: {dataset\_name}")

    print("---------------------")

    for model\_name, model in models:

        pipeline = Pipeline([('model', model)])

        scores = cross\_val\_score(pipeline, X\_train, y\_train, cv=5)

        pipeline.fit(X\_train, y\_train)

        y\_pred = pipeline.predict(X\_test)

        accuracy = accuracy\_score(y\_test, y\_pred)

        print(f"Model: {model\_name}")

        print("Dataset Used: ", dataset\_name)

        print("Test Accuracy: ", accuracy)

        print()

        if accuracy > best\_accuracy:

            best\_accuracy = accuracy

            best\_model = (dataset\_name, model\_name, pipeline)

print("Best Model: ", best\_model)

**Додаток №3**

%pip install numpy pandas scikit-learn seaborn matplotlib

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

heart\_disease\_dataset\_1 = pd.read\_csv("Dataset/data\_1.csv")

print(heart\_disease\_dataset\_1)

print(heart\_disease\_dataset\_1.columns)

heart\_disease\_dataset\_2 = pd.read\_csv("Dataset/data\_2.csv")

print(heart\_disease\_dataset\_2)

print(heart\_disease\_dataset\_2.columns)

heart\_disease\_dataset\_3 = pd.read\_csv("Dataset/data\_3.csv")

print(heart\_disease\_dataset\_3)

print(heart\_disease\_dataset\_3.columns)

null\_counts\_1 = heart\_disease\_dataset\_1.isnull().sum()

print(null\_counts\_1)

heart\_disease\_dataset\_2.replace('?', np.nan, inplace=True)

null\_counts\_2 = heart\_disease\_dataset\_2.isnull().sum()

print(null\_counts\_2)

null\_counts\_3 = heart\_disease\_dataset\_3.isnull().sum()

print(null\_counts\_3)

heart\_disease\_dataset\_1.columns = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target']

heart\_disease\_dataset\_2.columns = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target']

heart\_disease\_dataset\_3.columns = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target']

heart\_disease\_dataset\_3['target'] = heart\_disease\_dataset\_3['target'].apply(lambda x: 1 if x > 0.61 else 0)

print(heart\_disease\_dataset\_3)

combined\_dataset = pd.concat([heart\_disease\_dataset\_1, heart\_disease\_dataset\_2, heart\_disease\_dataset\_3])

combined\_dataset.shape

combined\_dataset.info()

target\_counts = combined\_dataset['target'].value\_counts()

plt.figure(figsize=(6, 4))

plt.bar(target\_counts.index.astype(str), target\_counts.values, color=['green', 'red'])

plt.xlabel('Target')

plt.ylabel('Count')

plt.title('Number of People with and without Heart Disease')

plt.xticks([0, 1], ['No Heart Disease', 'Heart Disease'])

plt.tight\_layout()

plt.show()

plt.figure(figsize=(6, 6))

plt.pie(target\_counts, labels=['No Heart Disease', 'Heart Disease'], colors=['green', 'red'], autopct='%1.1f%%', startangle=90)

plt.title('Percentage of People with and without Heart Disease')

plt.tight\_layout()

plt.show()

# Виведення середнього значення, моди та медіани кожної колонки

summary\_stats = combined\_dataset.describe(include='all').T

modes = combined\_dataset.mode().iloc[0]

summary\_stats['mode'] = modes

summary\_stats = summary\_stats[['mean', 'mode', '50%']]  # Вибір стовпців для виведення

print("Summary Statistics:")

print(summary\_stats)

# Побудова діаграм для кожної колонки

for column in combined\_dataset.columns:

    plt.figure(figsize=(6, 4))

    combined\_dataset[column].hist()

    plt.title(f'Histogram for {column}')

    plt.xlabel(column)

    plt.ylabel('Frequency')

    plt.grid(axis='y')

    plt.tight\_layout()

    plt.show()

nan\_counts = combined\_dataset.isnull().sum()

print(nan\_counts)

combined\_dataset['age'].min(), combined\_dataset['age'].max()

sns.histplot(combined\_dataset['age'], bins=20)

plt.xlabel('Age')

plt.ylabel('Count')

plt.title('Distribution of Age in Combined Dataset')

plt.show()

age\_column = combined\_dataset['age']

median\_age = age\_column.median()

mean\_age = age\_column.mean()

mode\_age = age\_column.mode()[0]

print(f"Медіана віку: {median\_age}")

print(f"Середнє значення віку: {mean\_age}")

print(f"Мода віку: {mode\_age}")

stats\_df = pd.DataFrame({'Stat': ['Median', 'Mean', 'Mode'],

                         'Value': [median\_age, mean\_age, mode\_age]})

plt.figure(figsize=(15, 6))

sns.barplot(x='Stat', y='Value', data=stats\_df)

plt.title('Statistics by Age')

plt.xlabel('Statistic')

plt.ylabel('Value')

plt.show()

gender\_counts = combined\_dataset['sex'].value\_counts()

print(gender\_counts)

plt.figure(figsize=(6, 4))

sns.barplot(x=gender\_counts.index, y=gender\_counts.values)

plt.title('Gender Distribution in Combined Dataset')

plt.xlabel('Gender')

plt.ylabel('Count')

plt.show()

# Групування за статтю та інфарктом

gender\_target\_groups = combined\_dataset.groupby(['sex', 'target']).size()

# Отримання кількості чоловіків та жінок з інфарктом та без нього

male\_heart\_attack = gender\_target\_groups[1, 1] if (1, 1) in gender\_target\_groups.index else 0

male\_no\_heart\_attack = gender\_target\_groups[1, 0] if (1, 0) in gender\_target\_groups.index else 0

female\_heart\_attack = gender\_target\_groups[0, 1] if (0, 1) in gender\_target\_groups.index else 0

female\_no\_heart\_attack = gender\_target\_groups[0, 0] if (0, 0) in gender\_target\_groups.index else 0

# Обчислення відсоткового співвідношення

total\_male = male\_heart\_attack + male\_no\_heart\_attack

total\_female = female\_heart\_attack + female\_no\_heart\_attack

male\_heart\_attack\_percent = (male\_heart\_attack / total\_male) \* 100

male\_no\_heart\_attack\_percent = (male\_no\_heart\_attack / total\_male) \* 100

female\_heart\_attack\_percent = (female\_heart\_attack / total\_female) \* 100

female\_no\_heart\_attack\_percent = (female\_no\_heart\_attack / total\_female) \* 100

# Побудова графіку

labels = ['Male Heart Attack', 'Male No Heart Attack', 'Female Heart Attack', 'Female No Heart Attack']

sizes = [male\_heart\_attack\_percent, male\_no\_heart\_attack\_percent, female\_heart\_attack\_percent, female\_no\_heart\_attack\_percent]

colors = ['lightcoral', 'lightskyblue', 'salmon', 'skyblue']

plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)

plt.axis('equal')

plt.title('Heart Attack Percentage by Gender')

plt.show()

cp\_counts = combined\_dataset['cp'].value\_counts()

print(cp\_counts)

plt.figure(figsize=(6, 4))

sns.barplot(x=cp\_counts.index, y=cp\_counts.values)

plt.title('Chest Pain Type Distribution')

plt.xlabel('Chest Pain Type')

plt.ylabel('Count')

plt.show()

combined\_dataset['trestbps'].describe()

combined\_dataset = combined\_dataset.dropna(subset=['trestbps'])

print(combined\_dataset)

combined\_dataset['trestbps'] = combined\_dataset['trestbps'].astype(float)

median\_trestbps = combined\_dataset['trestbps'].median()

mean\_trestbps = combined\_dataset['trestbps'].mean()

mode\_trestbps = combined\_dataset['trestbps'].mode()[0]

combined\_dataset.info()

combined\_dataset = combined\_dataset.dropna(subset=['restecg'])

print(combined\_dataset)

combined\_dataset['restecg'] = combined\_dataset['restecg'].astype(float)

median\_restecg = combined\_dataset['restecg'].median()

mean\_restecg = combined\_dataset['restecg'].mean()

mode\_restecg = combined\_dataset['restecg'].mode()[0]

combined\_dataset.info()

combined\_dataset['chol'] = pd.to\_numeric(combined\_dataset['chol'], errors='coerce').astype(float)

combined\_dataset['fbs'] = pd.to\_numeric(combined\_dataset['fbs'], errors='coerce').astype(float)

combined\_dataset['slope'] = pd.to\_numeric(combined\_dataset['slope'], errors='coerce').astype(float)

combined\_dataset['ca'] = pd.to\_numeric(combined\_dataset['ca'], errors='coerce').astype(float)

combined\_dataset['thal'] = pd.to\_numeric(combined\_dataset['thal'], errors='coerce').astype(float)

# Виведення оновленої інформації про датасет

print(combined\_dataset.info())

# Список стовпців, для яких потрібно виконати імпутацію

columns\_to\_impute = ['chol', 'fbs', 'slope', 'ca', 'thal']

# Копія початкового датасету для кожного методу імпутації

median\_dataset = combined\_dataset.copy()

mean\_dataset = combined\_dataset.copy()

mode\_dataset = combined\_dataset.copy()

constant\_dataset = combined\_dataset.copy()

deleted\_nan\_dataset = combined\_dataset.copy()

knn\_dataset = combined\_dataset.copy()

# Цикл для кожного методу імпутації

for column in columns\_to\_impute:

    # Медіана

    median\_value = combined\_dataset[column].median()

    median\_dataset[column] = median\_dataset[column].fillna(median\_value)

    # Середнє значення

    mean\_value = combined\_dataset[column].mean()

    mean\_dataset[column] = mean\_dataset[column].fillna(mean\_value)

    # Мода

    mode\_value = combined\_dataset[column].mode()[0]

    mode\_dataset[column] = mode\_dataset[column].fillna(mode\_value)

    # Видалення NaN значень

    deleted\_nan\_dataset = deleted\_nan\_dataset.dropna(subset=[column])

    # Заміна на константу

    constant\_value = 0  # Наприклад, вибір деякої константи

    constant\_dataset[column] = constant\_dataset[column].fillna(constant\_value)

knn\_imputer = KNNImputer(n\_neighbors=5)

knn\_dataset[columns\_to\_impute] = knn\_imputer.fit\_transform(knn\_dataset[columns\_to\_impute])

# Виведення інформації про оновлені датасети

print("Median Dataset:")

print(median\_dataset.info())

print("\nMean Dataset:")

print(mean\_dataset.info())

print("\nMode Dataset:")

print(mode\_dataset.info())

print("\nDeleted NaN Dataset:")

print(deleted\_nan\_dataset.info())

print("\nConstant Dataset:")

print(constant\_dataset.info())

print("\nKNN Imputed Dataset:")

print(knn\_dataset.info())

X\_median = median\_dataset.drop('target', axis=1)

y\_median = median\_dataset['target']

X\_train\_median, X\_test\_median, y\_train\_median, y\_test\_median = train\_test\_split(X\_median, y\_median, test\_size=0.1, random\_state=42)

X\_mean = mean\_dataset.drop('target', axis=1)

y\_mean = mean\_dataset['target']

X\_train\_mean, X\_test\_mean, y\_train\_mean, y\_test\_mean = train\_test\_split(X\_mean, y\_mean, test\_size=0.1, random\_state=42)

X\_mode = mode\_dataset.drop('target', axis=1)

y\_mode = mode\_dataset['target']

X\_train\_mode, X\_test\_mode, y\_train\_mode, y\_test\_mode = train\_test\_split(X\_mode, y\_mode, test\_size=0.1, random\_state=42)

X\_deleted\_nan = deleted\_nan\_dataset.drop('target', axis=1)

y\_deleted\_nan = deleted\_nan\_dataset['target']

X\_train\_deleted\_nan, X\_test\_deleted\_nan, y\_train\_deleted\_nan, y\_test\_deleted\_nan = train\_test\_split(X\_deleted\_nan, y\_deleted\_nan, test\_size=0.1, random\_state=42)

X\_constant = constant\_dataset.drop('target', axis=1)

y\_constant = constant\_dataset['target']

X\_train\_constant, X\_test\_constant, y\_train\_constant, y\_test\_constant = train\_test\_split(X\_constant, y\_constant, test\_size=0.1, random\_state=42)

X\_knn = knn\_dataset.drop('target', axis=1)

y\_knn = knn\_dataset['target']

X\_train\_knn, X\_test\_knn, y\_train\_knn, y\_test\_knn = train\_test\_split(X\_knn, y\_knn, test\_size=0.1, random\_state=42)

train\_rows = len(X\_train\_median)

test\_rows = len(X\_test\_median)

# Виводимо інформацію про кількість тренувальних та тестових рядків

print(f"Кількість тренувальних рядків: {train\_rows}")

print(f"Кількість тестових рядків: {test\_rows}")

# Побудова діаграми

plt.figure(figsize=(8, 5))

plt.bar(['Тренувальні дані', 'Тестові дані'], [train\_rows, test\_rows], color=['blue', 'orange'])

plt.xlabel('Тип даних')

plt.ylabel('Кількість рядків')

plt.title('Кількість тренувальних та тестових рядків')

plt.show()

from sklearn.pipeline import Pipeline

from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, mean\_absolute\_error, mean\_squared\_error

models = [

    ('KNeighbors Classifier', KNeighborsClassifier()),

    ('Logistic Regression', LogisticRegression(max\_iter=1000, solver='liblinear', random\_state=42)),

    ('Decision Tree Classifier', DecisionTreeClassifier(random\_state=42)),

    ('Gradient Boosting', GradientBoostingClassifier(random\_state=42)),

    ('Naye base Classifier', GaussianNB())

]

datasets = [

    ('Median Imputation', X\_train\_median, X\_test\_median, y\_train\_median, y\_test\_median),

    ('Mean Imputation', X\_train\_mean, X\_test\_mean, y\_train\_mean, y\_test\_mean),

    ('Mode Imputation', X\_train\_mode, X\_test\_mode, y\_train\_mode, y\_test\_mode),

    ('Deleted NaN Imputation', X\_train\_deleted\_nan, X\_test\_deleted\_nan, y\_train\_deleted\_nan, y\_test\_deleted\_nan),

    ('Constant Imputation', X\_train\_constant, X\_test\_constant, y\_train\_constant, y\_test\_constant),

      ('KNN Imputation', X\_train\_knn, X\_test\_knn, y\_train\_knn, y\_test\_knn)

]

# Define a function to plot accuracy for each model

def plot\_accuracy\_for\_ml\_method(model\_name, model, imputation\_datasets):

    dataset\_names = []

    accuracy\_values = []

    for dataset\_name, X\_train\_imp, X\_test\_imp, y\_train\_imp, y\_test\_imp in imputation\_datasets:

        pipeline = Pipeline([('imputation', 'passthrough'), ('model', model)])

        pipeline.fit(X\_train\_imp, y\_train\_imp)

        y\_pred = pipeline.predict(X\_test\_imp)

        accuracy = accuracy\_score(y\_test\_imp, y\_pred)

        dataset\_names.append(dataset\_name)

        accuracy\_values.append(accuracy)

        print(f"Model: {model\_name}")

        print("Dataset Used: ", dataset\_name)

        print("Test Accuracy: ", accuracy)

    plt.figure(figsize=(8, 4))

    plt.bar(dataset\_names, accuracy\_values, color='lightgreen')

    plt.xlabel('Imputation Method')

    plt.ylabel('Accuracy')

    plt.title(f'Accuracy for {model\_name} Model with Different Imputation Methods')

    plt.xticks(rotation=45, ha='right')

    plt.ylim(0, 1)

    plt.tight\_layout()

    plt.show()

# Call the function for each model

for model\_name, model in models:

    plot\_accuracy\_for\_ml\_method(model\_name, model, datasets)