**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 1** Методи бібліотеки sklearn

У таблиці 1 показано, які методи бібліотеки ми будемо імпортувати і використовувати.

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

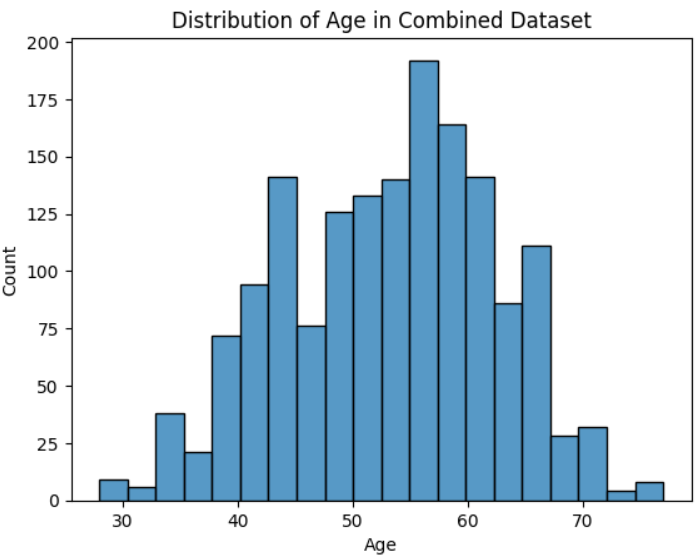
Перед робото з датасетом, перевіримо чи він є збалансованим.

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.

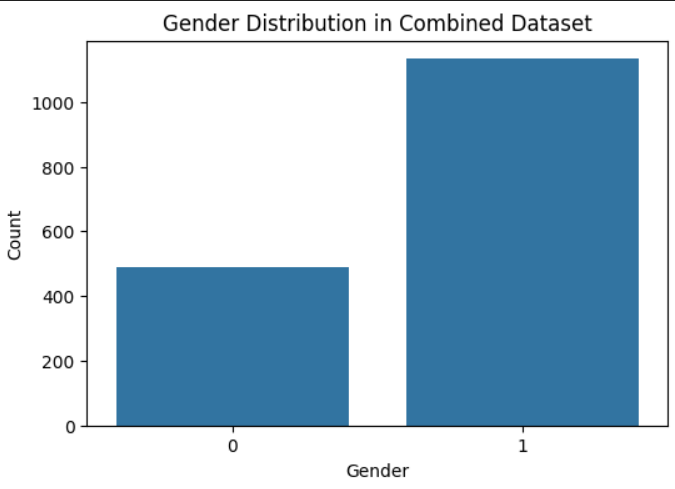
Перевірка датасету за віком:



**Рисунок 1** Діаграма за віком

На рисунку 1 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.

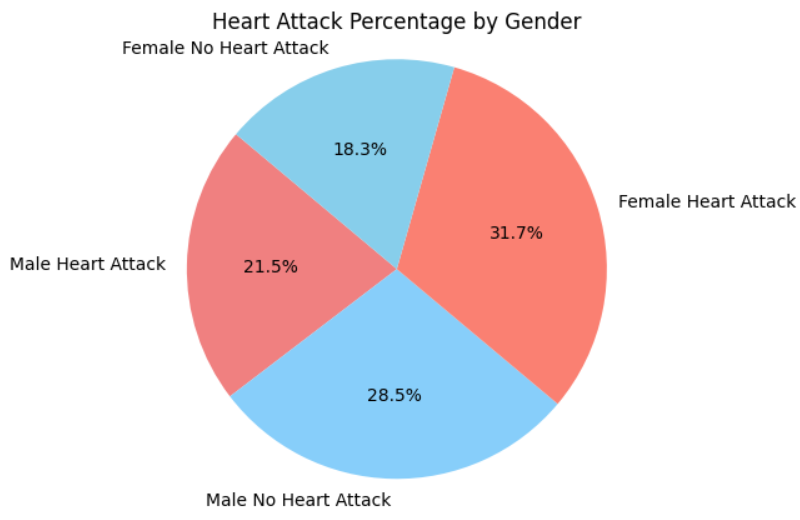
Перевірка датасету за статтю, чоловік чи жінка:



**Рисунок 2** Діаграма за статтею

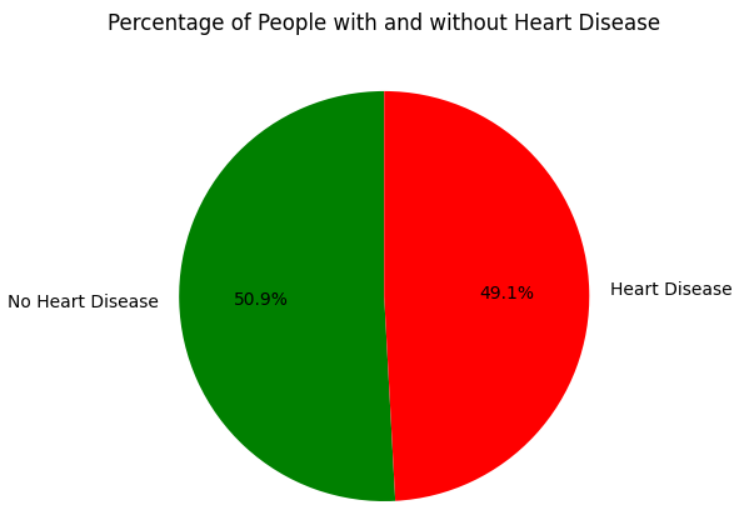
На рисунку 2 показується кількість чоловіків і жінок у датасеті, які мали або не мали інфаркту. Бачимо, що чоловіків в 2 рази більше за жінок.

Перевірка датасету, скільки інфарктів у чоловіків і жінок:

****

**Рисунок 3** Діаграма кількості чоловіків і жінок з і без інфаркту

На рисунку 3 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 4** Кількість людей з і без інфаркту, кругова діаграма

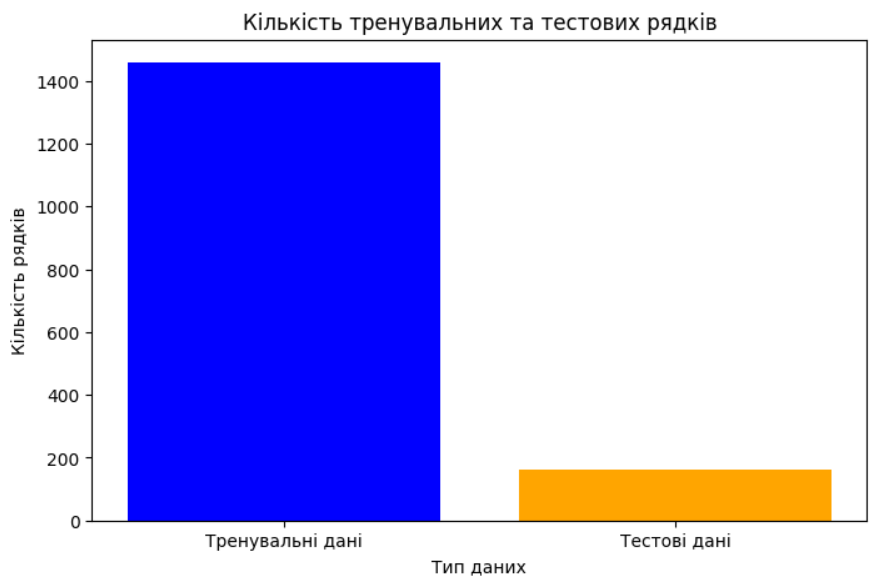
На рисунку 3 і рисунку 4 показано діаграми, які мають співвідношення людей з інфарктом і без інфаркту. У датасеті є більше людей, які не мають інфаркт. Оскільки їхня частка в датасеті 50.9%. Пацієнти які мають інфаркт у датасеті займають 49.1% всього набору даних.

З Рисунку 4 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 1. Це необхідно для того, щоб дізнатися, які стовпці потрібно обробляти від пропущених значень.

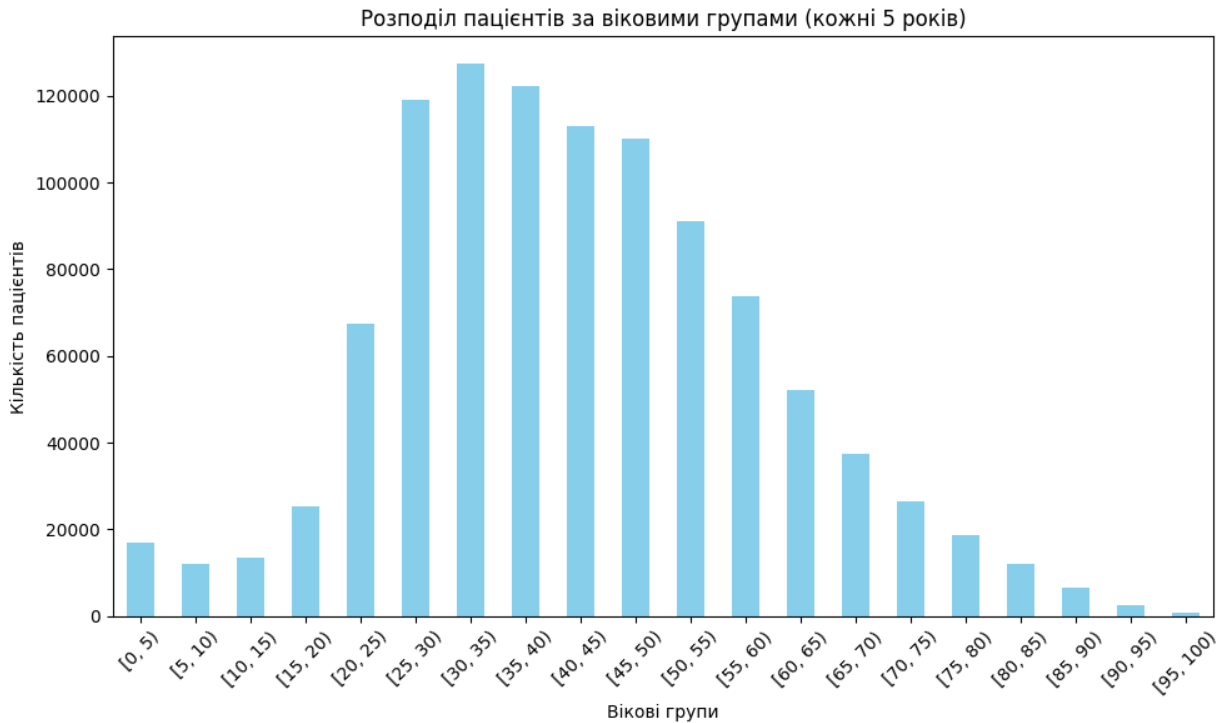


**Рисунок 5** Розділення набору даних на тренувальну і тестову

Рисунок 5 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**

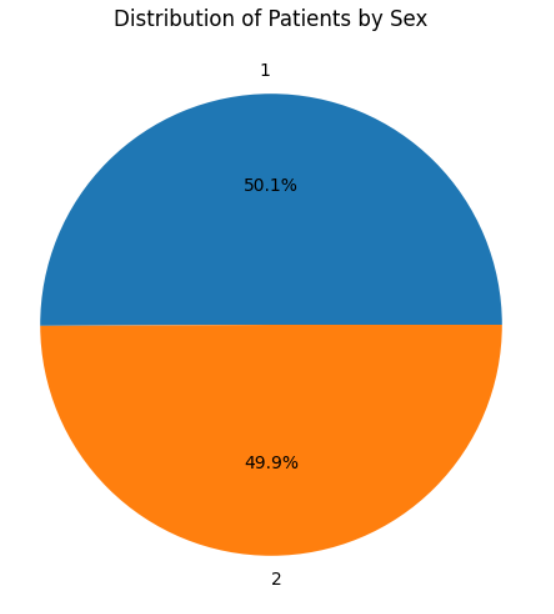
У рисунку 8 показано, що даний датасет є збалансованим. Датасет є збалансованим, оскільки це відображає реальну ситуацію. Також перевіримо кількість людей за віком і статтю:



**Рисунок 6** Відображення статистики за віком

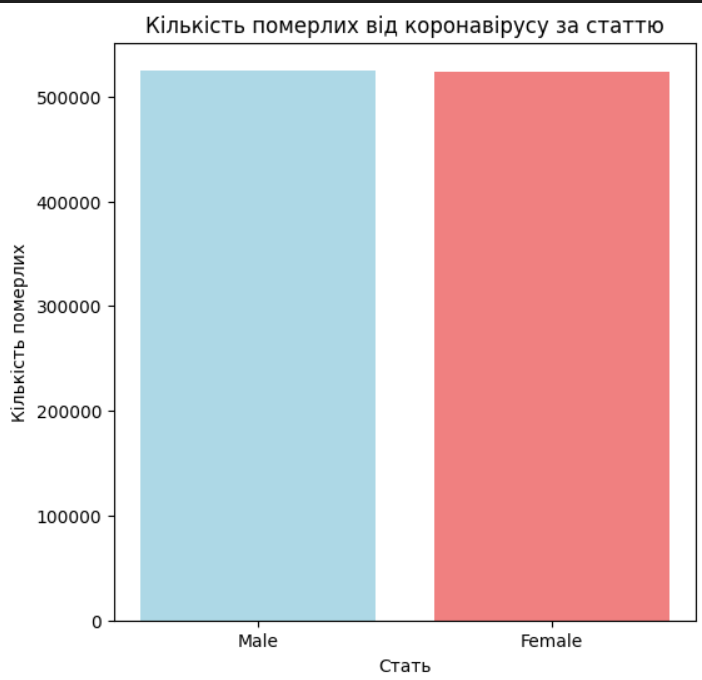
У Рисунку 6 відображається статистика за віком. У датасеті переважає кількість людей від 25 до 50 років. Найбільше захворювань у людей віком 30 – 35 років.

Перевірка датасету за статтю, чоловік чи жінка:



**Рисунок 7** Діаграма за статтею

На рисунку 7 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 8** Кількість чоловіків і жінок померлих від коронавірусу

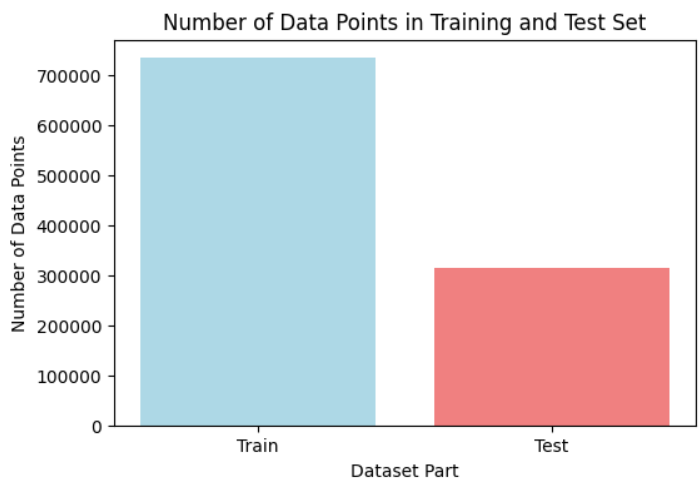
Рисунок 8 показано діаграми, які мають співвідношення чоловіків і жінок померлих від коронавірусу. Померлих чоловіків 50% і померлих жінок 50% серед померлих людей у датасеті.

З Рисунку 8 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.

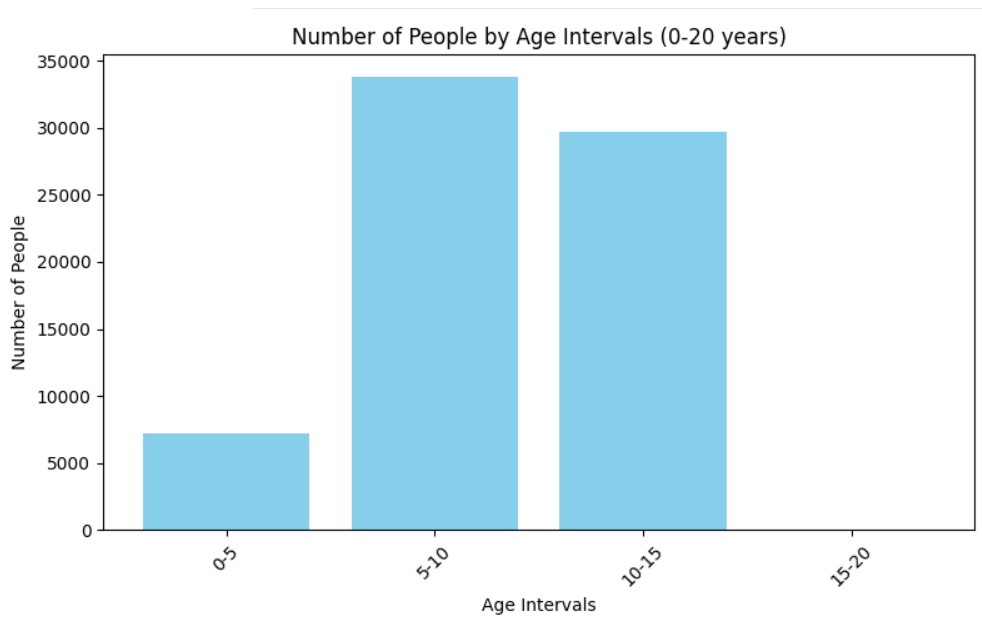


**Рисунок 9** Розділення набору даних на тренувальну і тестову

Рисунок 9 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**

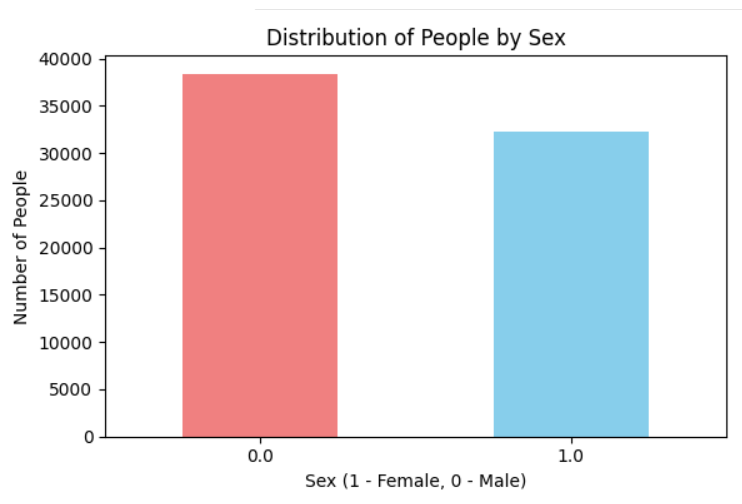
У рисунку 11 показано, що даний датасет є збалансованим. Датасет є збалансованим, оскільки він містить 50% людей з діабетом і 50% людей без діабету. Розберемо такі колонки, як вік і стать:



**Рисунок 6** Відображення статистики за віком

У Рисунку 6 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 7** Діаграма за статтею

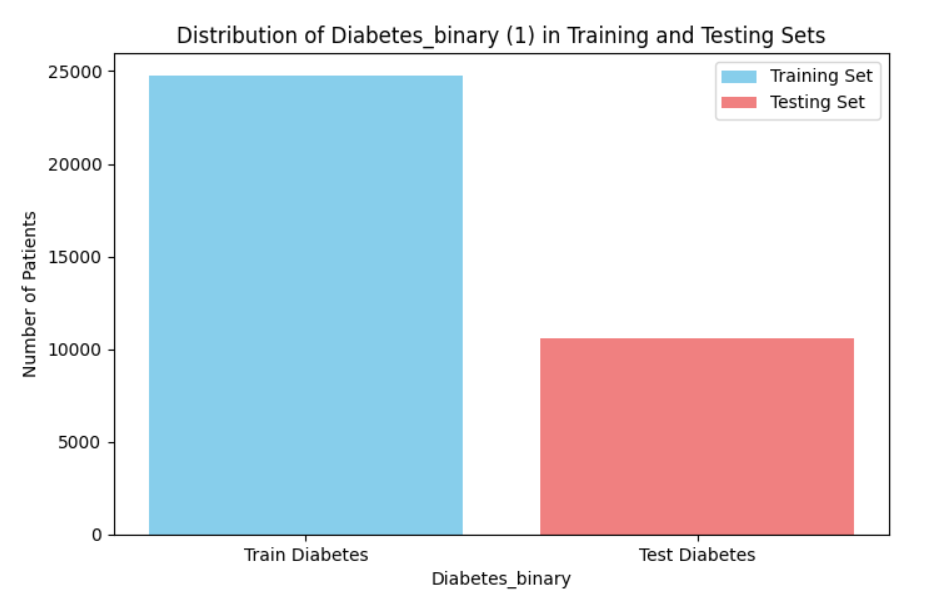
На рисунку 7 відображається діаграма – кількість чоловіків і жінок у датасеті, які мають і не маю діабет. Бачимо, що чоловіків така сама кількість як і жінок. Частка чоловіків 32306, а жінок 38386, тобто переважають жінки.

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 9** Розділення набору даних на тренувальну і тестову

Рисунок 9 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

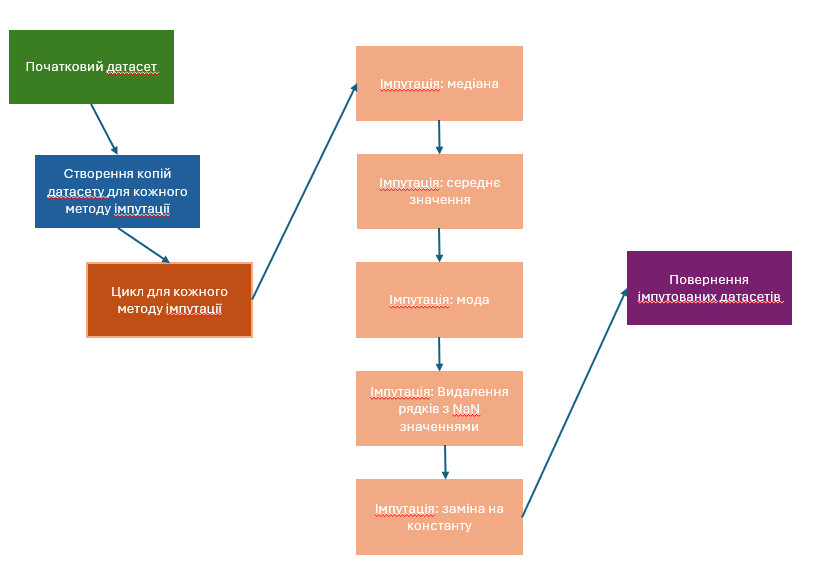
    deleted\_nan\_dataset = deleted\_nan\_dataset.dropna(subset=[column])

    # 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 5** Кроки обробки пропущених значень

У Рисунку 5 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 6** Псевдо код для навчання різними машинними моделями

Рисунок 6 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.