**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.

Було проаналізовано різні підходи для обробки пропущених значень та визначення невизначеності у великих наборах даних, було розглянено різні математичні моделі для передбачення NaN значень, такі як: лінійна регресія, дерево рішень, KNN, MICE, deep learning, логістична регресія, LGBM. 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. **Diabetes Health Indicators Dataset**

Даний датасет знаходиться на платформі Kaggle у відкритому доступі. Цей датасет складається 3 датасетів з опитування. Давайте його розберемо:

1. **diabetes\_012\_health\_indicators\_BRFSS2015** – це набір даних із 253 680 відповідей на опитування CDC BRFSS2015. Цільова колонка – має 3 класи: де 0 означає відсутність діабету або лише під час вагітності, 1 – переддіабет, 2 – діабет. Даний датасет містить 21 змінну функції, що робить його дизбалансованим:

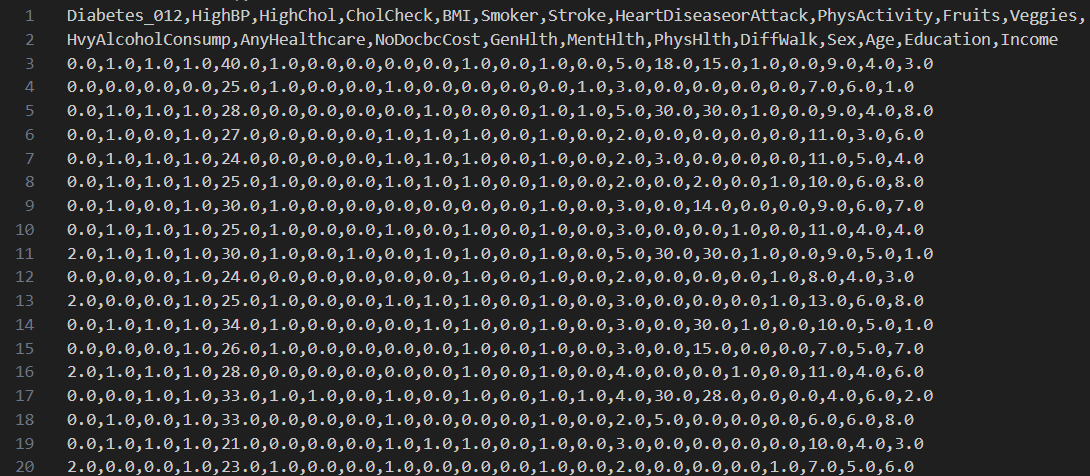


Рисунок 2.1.1.1

**Основні характеристики датасету:**

1. **Кількість рядків:** 253,680 записів анкетних відповідей.
2. **Кількість стовпців:** датасет має 21 ознаку (feature) для кожної анкетної відповіді.
3. **Цільова змінна:**

* Датасет має дві версії цільової змінної: Diabetes\_012 та Diabetes\_binary.
* Diabetes\_012 має 3 класи: 0 (відсутність діабету або лише під час вагітності), 1 (предіабет) та 2 (діабет).
* Diabetes\_binary має 2 класи: 0 (відсутність діабету) та 1 (предіабет або діабет).
* В деяких випадках спостерігається дисбаланс класів.

1. **Характеристики:**

* Ознаки включають питання, що стосуються ризикових факторів для діабету, хронічних захворювань та поведінкових чинників.
* Деякі ознаки можуть бути визначені на основі відповідей учасників.

1. **Джерело даних:** датасет зібраний з анкет BRFSS2015, які збираються щорічно CDC та містять інформацію про поведінкові ризики, хронічні захворювання та використання профілактичних послуг в області здоров'я в США.
   * 1. **Heart Disease Dataset**

Даний набір даних знаходиться на платформі Kaggle у відкритому доступі. Цей набір даних є багатовимірним, що означає, що він включає різноманітні математичні або статистичні змінні, що дозволяє проводити багатовимірний числовий аналіз даних. Він складається з 14 атрибутів, які надають важливу інформацію для аналізу захворювання на серце. Також він містить велику кількість даних, а саме 919 унікальних рядків.

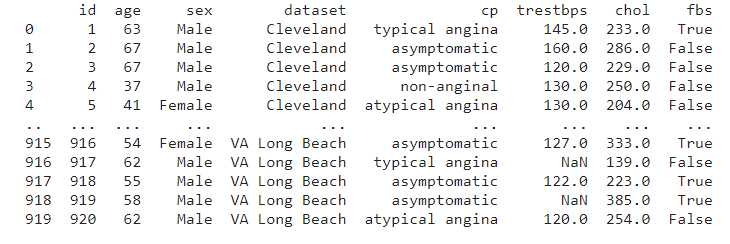


Рисунок 2.1.2

**2.2 Основні алгоритми**

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).

**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.

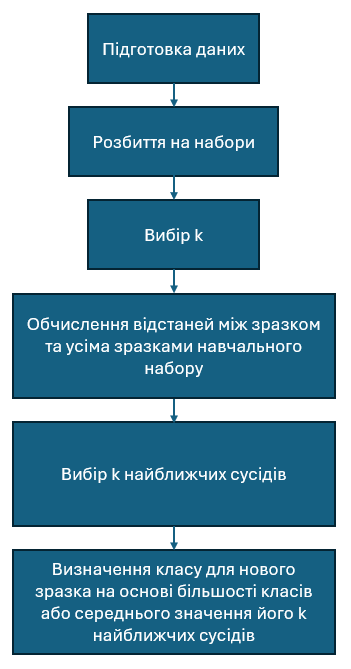


Рисунок 2.2.1.1

**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.

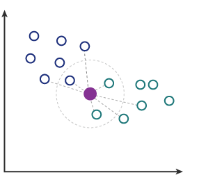


Рисунок 2.2.1.2

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.

**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.

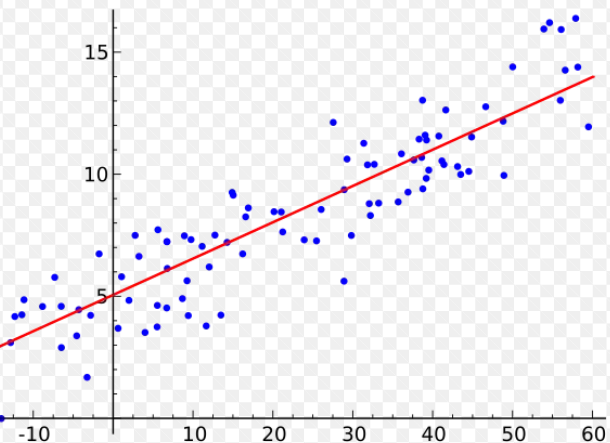


Рисунок 2.2.2

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.

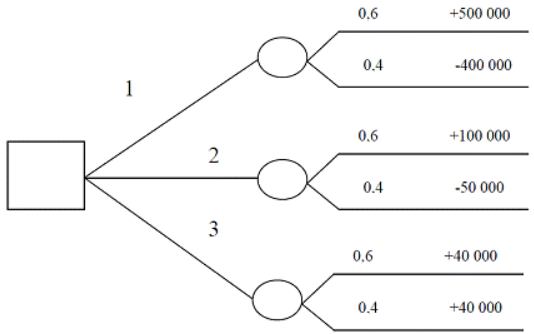


Рисунок 2.2.3

**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.

**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.

**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 Основні методи 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 |