Зміст

[Вступ 3](#_Toc507625391)

[Постановка задачі 4](#_Toc507625392)

[1. Побудова дерев регресії 5](#_Toc507625393)

[1.1 Задача регресії 5](#_Toc507625394)

[1.2 Поняття дерева регресії 5](#_Toc507625395)

[1.3 Алгоритми побудови дерева регресії 8](#_Toc507625396)

[2. Програмне забезпечення побудови дерев регресії 14](#_Toc507625397)

[2.1 Функціональні можливості програми 14](#_Toc507625398)

[2.2 Організація обчислювального процесу 14](#_Toc507625399)

[Висновки 19](#_Toc507625400)

[Список використаної літератури 20](#_Toc507625401)

# Вступ

Регресійний аналіз – це статистичний метод дослідження залежності випадкової величини Y від змінних Xj (j = 1, 2, k), що розглядаються в регресійному аналізі як невипадкові величини незалежно від істинного закону розподілу Xj.

Регресійний аналіз використовують для:

1. визначення ступеня детермінованості варіації критеріальної (залежної) змінної предикторами (незалежними змінними);
2. передбачення значення залежної змінної за допомогою незалежних;
3. визначення вкладу окремих незалежних змінних в варіацію залежної.

Один з підходів до вирішення задачі регресії полягає в побудові дерева регресії. Для побудови дерева регресії створені наступні алгоритми: CART, M5, RETIS, GMT та інші.[6]

В ході виконання роботи для побудови власного дерева був обраний алгоритм CART.

# Постановка задачі

Написати програмне забезпечення, що вирішує задачу регресії на основі алгоритму CART. На вхід подається текстовий файл, що містить у собі дані вигляду

.

Скориставшись алгоритмом CART потрібно побудувати дерево регресії.

Параметри можуть бути як кількісними, так і якісними.

# Побудова дерев регресії

1. **Задача регресії**

Регресійний аналіз – статистичний метод дослідження впливу однієї або декількох незалежних змінних X1, X2, ..., Xp на залежну змінну Y. Незалежні змінні інакше називають регресорами або предикторами, а залежні змінні - критеріальними. Термінологія залежних і незалежних змінних відображає лише математичну залежність змінних, а не причинно-наслідкові зв'язки.

Строго регресійну залежність можна визначити наступним чином. Нехай – випадкові величини із заданим спільним розподілом ймовірностей. Якщо для кожного набору значень визначено умовне математичне очікування

,

то функція називається регресією величини по величинам а її графік – лінією регресії по , або рівнянням регресії.

Залежність від проявляється в зміні середніх значень при зміні значень . Хоча при кожному фіксованому наборі значень величина залишається випадковою величиною з визначеним розподілом.

Регресійний аналіз неможна використовувати для визначення наявності зв'язку між змінними, оскільки наявність такого зв'язку і є передумовою для застосування аналізу.[7]

1. **Поняття дерева регресії**

Дерево регресії є різновидом дерева рішень.

Згідно з найбільш загальним визначенням, дерево прийняття рішень – це засіб підтримки прийняття рішень при прогнозуванні, широко застосовується в статистиці і аналізі даних.[2]

Структура дерева являє собою «листя» і «гілки». На ребрах («гілках») дерева рішення записані атрибути, від яких залежить цільова функція, в «листі» записані значення цільової функції, а в інших вузлах – атрибути, за якими розрізняються випадки. Щоб класифікувати новий випадок, або спрогнозувати значення змінної, треба спуститися по дереву до листа і видати відповідне значення. Подібні дерева рішень широко використовуються в інтелектуальному аналізі даних. Мета полягає в тому, щоб створити модель, яка передбачає значення цільової змінної на основі декількох змінних на вході.[3]

Дерева рішень (рис.1.1) – це спосіб представлення правил в ієрархічній, послідовній структурі, де кожному об'єкту відповідає єдиний вузол, що дає рішення.

Під правилом розуміється логічна конструкція, представлена у вигляді "якщо ... то ...".[2]

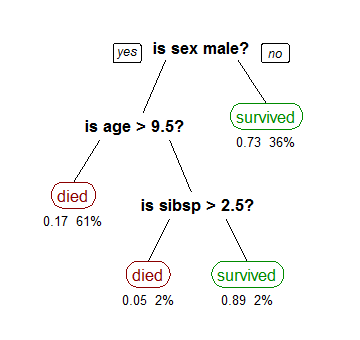


рис.1.1. Дерево рішень

Область застосування дерева рішень в даний час широка, але всі завдання, які вирішуються цим апаратом можуть бути об'єднані в наступні три класи:

* **Опис даних**: Дерева рішень дозволяють зберігати інформацію про дані в компактній формі, замість них можна зберігати дерево рішень, яке містить точний опис об'єктів.
* **Класифікація**: Дерева рішень відмінно справляються з завданнями класифікації, тобто віднесенням об'єктів до одного з заздалегідь відомих класів. Цільова змінна повинна мати дискретні значення.
* **Регресія**: Якщо цільова змінна має безперервні значення, дерева рішень дозволяють встановити залежність цільової змінної від незалежних (вхідних) змінних. Наприклад, до цього класу належать задачі чисельного прогнозування (передбачення значень цільової змінної).[2]

Дерева регресії є одним з важливих класів регресійних моделей, що дозволяють здійснити поділ вхідного простору на сегменти з наступною побудовою для кожного з них власної (локальної) моделі і представити кусково-задану функцію регресії в інтуїтивно зрозумілій і наочній формі. В такому дереві внутрішні вузли містять правила поділу простору пояснюючих змінних X; дуги – умови переходу по ним; а «листя» – локальні регресійні моделі.[6]

**Переваги дерев регресії:**

* простота інтерпретації та наочність;
* можливість роботи як з категоріями, так і з кількісними значеннями;
* можливість роботи з пропусками в даних (порожніми значеннями атрибутів). Причому дерева регресії можна використовувати для заповнення пропусків найбільш імовірним значенням;
* хороша продуктивність в процесі визначення результату по вже побудованому дереву (так як алгоритм пошуку в дереві дуже ефективний навіть для великих наборів даних).

**Недоліки дерев регресії:**

* нестабільність процесу. Нерідко невеликі зміни в наборі даних можуть призводити до побудови абсолютно іншого дерева. Це пов'язано з ієрархічністю дерева. Зміни в вузлі на верхньому рівні ведуть до змін у всьому дереві нижче.
* складність контролю розміру дерева. Розмір дерева є критичним фактором, що визначає якість виконання завдання. При використанні простих критеріїв зупинки дерева часто стають або дуже короткими, або дуже великими.
* неадекватність поділу на піддерева в складних випадках. У найпростіших деревах рішень розбиття в вузлах відбувається за значенням одного атрибута. В даному випадку кожен атрибут - це вісь координат зі своїми значеннями. І дерево «нарізає» весь простір на «паралелепіпеди», всередині яких і групуються точки набору даних, які відповідають тому чи іншому класу. Іноді такий поділ не може точно описати складні області.
* критерій приросту інформації характеризується схильністю надавати перевагу атрибутам, що мають велике число різних значень. У граничному випадку у кожного рядка може бути своє значення атрибута. І тоді приріст буде максимальним.

Більш розвинуті алгоритми побудови дерев дозволяють вирішувати зазначені проблеми.[5]

## Алгоритми побудови дерева регресії

Більшість сучасних алгоритмів побудови дерев регресії є жадібними. Такі алгоритми здійснюють побудову дерев зверху вниз шляхом рекурсивного поділу навчальних даних і коротко можуть бути описані в такий спосіб:

1. вибрати кращий розподіл (зазвичай вибір розподілу, який забезпечує екстремум деякого критерію);

2. розділити вихідні дані на підмножини;

3. рекурсивно застосувати цю процедуру до кожної з виділених підмножин.

Жадібні алгоритми мають низьку трудомісткість, добре масштабуються, але мають ряд недоліків:

а) дерево регресії будується поступово без повернення до раніше прийнятих рішень;

б) на кожному кроці роботи алгоритму приймається локально оптимальне рішення, тобто рішення, що дає максимальний ефект на поточному кроці, без урахування його впливу на все рішення в цілому. Тому вони призводять, як правило, до неоптимальному поділу даних.[6]

**Алгоритм С4.5**

**Що він робить?** Алгоритм C4.5 будує класифікатор в формі дерева рішень. Щоб зробити це, йому потрібно передати набір вже класифікованих даних.

Класифікатор - це інструмент, який застосовується в data mining, який використовує класифіковані дані і на їх підставі намагається передбачити, до якого класу варто віднести нові дані.

**Суть алгоритму**. Використовуючи набір атрибутів об’єкту і відповідний клас, C4.5 будує дерево рішень, здатне передбачити клас для нових об’єктів на підставі їх атрибутів.

**Чи потребує цей метод навчання або він самонавчальний?** Цей метод вимагає навчання, тут тренувальний набір даних розмічається класами.

Відмінності C4.5 від інших систем, що використовують дерева рішень:

1. C4.5 використовує ентропію інформації, при створенні дерева рішень.
2. хоча інші системи також проріджують гілки дерева рішень, C4.5 використовує однопрохідне проріджування, щоб уникнути перенавчання. Відсікання гілок поліпшує модель.
3. C4.5 може працювати з дискретними і безперервними значеннями.
4. пропущені дані обробляються своїми власними способами.

Представлений вище алгоритм застосовується для побудови дерев класифікації. Проте Квінлан вдосконалив його для побудови дерев регресії. Вдосконалений алгоритм має назву М5.

**Алгоритм М5**

M5 будує деревовидні моделі. Звичайні дерева регресії мають числові значення у своїх листах, проте дерева, побудовані за m5, можуть мати багатовимірні лінійні моделі; таким чином ця модель дерев аналогічна кусково-лінійним функціям. М5 ефективно вчиться і може вирішувати завдання з дуже великою розмірністю (до сотень атрибутів, ця здатність встановлює m5 (і CART) окремо від інших алгоритмів, чиї обчислювальні вимоги ростуть дуже швидко з розмірністю). Перевага m5 над CART полягає в тому, що моделі дерев, побудовані першим алгоритмом, як правило, набагато менше за дерева регресії побудовані останнім, і виявляються більш точними в досліджуваних завданнях.[8]

**Алгоритм CART**

**Що він робить?** CART (classification and regression trees) - це абревіатура, що позначає методи класифікації і регресії з використанням дерева рішень. Це методика навчання, заснована на деревах рішень, яка повертає класифікаційні або регресивні дерева. Як було у випадку з C4.5, CART - це класифікатор.

**Дерево класифікації виглядає так само як дерево рішень?** Дерево класифікацій - це підвид дерева рішень. Результатом роботи дерева класифікацій є клас.

**Що таке дерево регресії?** Дерево класифікацій на виході має клас, а дерево регресії числову або безперервну величину. Дерева класифікацій виводять класи, дерева регресії – числа.

**Чи потребує цей метод навчання або він самонавчальний?** CART вимагає навчання, оскільки для побудови дерева класифікацій та дерева регресії необхідний розмічений набір даних.

**Відмінності алгоритмів CART та С4.5**

|  |  |
| --- | --- |
| **С4.5** | **CART** |
| Використовує ентропію до сегменту даних в процесі створення дерева рішень. | Тут використовується невизначеність Джині. |
| Використовує однопрохідний метод проріджування, щоб зменшити перенавчання. | Використовує механізм відсікання дерева при проріджуванні. Починаючи з низу дерева, CART оцінює помилку класифікації в вузлі і поза вузлом. Якщо похибка перевищує граничну, то гілка відкидається |
| Вузли дерева рішень можуть мати дві або більше гілок. | Вузли рішення мають дві гілки |
| На основі ймовірностей розподіляє відсутні значення між «дітьми». | Використовує сурогатні змінні, щоб передати відсутні дані «дітям».[4] |

Для побудови дерева регресії був обраний алгоритм CART.

Цей алгоритм призначений для вирішення завдань класифікації і регресії. Існує також кілька модифікованих версій – алгоритми IndCART і DB-CART. Алгоритм IndCART, є частиною пакета Ind і відрізняється від CART використанням іншого способу обробки пропущених значень, не здійснює регресійну частина алгоритму CART і має інші параметри відсікання. Алгоритм DB-CART базується на наступній ідеї: замість того щоб використовувати навчальний набір даних для визначення розбиття, використовуємо його для оцінки розподілу вхідних і вихідних значень і потім використовуємо цю оцінку, щоб визначити розбиття. Стверджується, що ця ідея дає значне зменшення помилки класифікації, в порівнянні зі стандартними методами побудови дерева.

Побудова дерева регресії багато в чому схожа з деревом класифікації. Спочатку ми будуємо дерево максимального розміру, потім обрізаємо дерево до оптимального розміру.

Основна перевага дерев у порівнянні з іншими методами регресії – можливість працювати з багатовимірними завданнями і завданнями, в яких присутня залежність вихідної змінної від змінної або змінних категоріального типу.

Основна ідея – розбиття всього простору на прямокутники, необов'язкового однакового розміру, в яких вихідна змінна вважається постійною. Варто зауважити, що існує сильна залежність між обсягом навчальної вибірки і помилкою відповіді дерева.

**Структура дерева**

Оскільки дерево, що будує алгоритм CART – бінарне, кожен вузол дерева рішень має двох нащадків. На кожному кроці побудови дерева правило, що формується в вузлі, ділить задану множину прикладів (навчальну вибірку) на дві частини – частину, в якій виконується правило (нащадок - right) і частину, в якій правило не виконується (нащадок - left). Для вибору оптимального правила використовується функція оцінки якості розбиття.

**Побудова дерева**

Процес побудови дерева відбувається послідовно. На першому кроці ми отримуємо регресійну оцінку просто як константу по всьому простору прикладів. Константу вважаємо середнім арифметичним вихідної змінної в навчальній вибірці. Отже, якщо ми позначимо все значення вихідної змінної як , , ..., тоді регресійна оцінка виходить:

,

де R – простір навчальних прикладів, n – число прикладів, – індикаторна функція простору – фактично, набір правил, що описують потрапляння змінної в простір.

Розглядаємо простір R як прямокутник. На другому кроці ми ділимо простір на дві частини. Вибирається деяка змінна і якщо змінна числового типу, тоді визначаємо:

якщо категоріального типу з можливими значеннями , , ..., , тоді вибирається деяка підмножина і визначаємо:

Яким чином вибирається найкраще розбиття? В якості оцінки тут служить сума квадратів різниць:

Обирається розбиття з мінімальною сумою квадратів різниць.

Продовжуємо розбиття до тих пір, поки в кожному підпросторі не залишиться мале число прикладів або сума квадратів різниць не стане менше деякого порога.

**Правило розбиття**

Вектор предикторних змінних, що подається на вхід дерева може містити як числові (порядкові) так і категоріальні змінні. У будь-якому випадку в кожному вузлі розбиття йде тільки по одній змінній. Якщо змінна числового типу, то в вузлі формується правило виду , де с – деякий поріг, який найчастіше вибирається як середнє арифметичне двох сусідніх упорядкованих значень змінної навчальної вибірки. Якщо змінна категоріального типу, то в вузлі формується правило , V(xi), де V() – деяка не порожня підмножина множини значень змінної в навчальній вибірці. На кожному кроці побудови дерева алгоритм послідовно порівнює всі можливі розбиття для всіх атрибутів і вибирає найкращий атрибут і найкраще розбиття для нього.[9]

# Програмне забезпечення побудови дерев регресії

## Функціональні можливості програми

Під час переддипломної практики був створений консольний додаток для побудови дерев регресії.

За допомогою консольного додатку можна будувати дерева регресії за вибіркою, що зберігається у текстовому файлі формату .txt. У файлі елементи вибірки зберігаються наступним чином:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Y** | **X1** | **IsQ** | **…** | **…** | **XN** | **IsQ** |
|  |  | {0,1} | … | … |  | {0,1} |
| … | … | … | … | … | … | … |
|  |  | {0,1} | … | … |  | {0,1} |

У першому стовпці знаходяться значення залежної змінної Y. У наступних двох стовпцях інформація про незалежну змінну, у першому значення, у другому інформація про те, чи є ця змінна якісною (1), чи ні (0). Кількість залежних змінних необмежена.

Таким чином програма вміє працювати з багатомірними наборами даних, а також з кількісними та якісними параметрами.

Будує програма бінарні дерева використовуючи алгоритм CART.

## Організація обчислювального процесу

Для побудови дерева регресії, за допомогою алгоритму CART були створені наступні класи (рис.2.1).

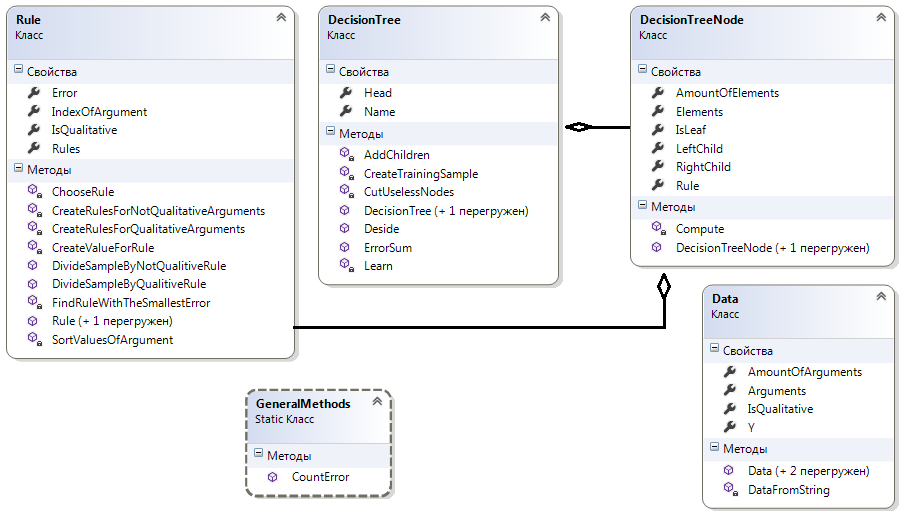


Рис.2.1 Діаграма класів

* DecisionTree.

Клас який зберігає в собі дерево (посилання на «голову»). Реалізує навчання і отримання прогнозованого рішення.

Містить у собі методи:

Публічні:

* public void Deside() – використовується на вже збудованому дереві аби спрогнозувати рішення на основі заданих параметрів.

Приватні:

* private Data[] CreateTrainingSample(List<string>) – цей метод викликається при ініціалізації об’єкта класу DecisionTree. Відповідає за створення навчальної вибірки спеціального формату. Отримуючи елементи у вигляді текстових рядків, повертає масив об’єктів Data.
* private void Learn() – цей метод викликається при ініціалізації об’єкта класу DecisionTree. Відповідає за побудову дерева за алгоритмом CART. Здійснює виклик методів AddChildren та ErrorSum.
* private void AddChildren(DecisionTreeNode) – цей метод викликається під час побудови дерева. Відповідає створення нащадків у заданому вузлі.
* private double ErrorSum() – цей метод викликається під час побудови дерева. Відповідає за підрахунок сумарної помилки у листах дерева. Для цього використовує алгоритм проходу дерева у ширину.
* DecisionTreeNode. Клас, який представляє вузол дерева, зберігає посилання на правий та лівий вузол. Зберігає інформацію про кількість елементів у вузлі, чи є даний вузол «листом» та яке правило розподілу має цей вузол.

Допоміжні класи:

* Data.

Цей клас зберігає навчальні дані у спеціальному вигляді. Екземпляр цього класу зберігає інформацію про кількість параметрів, результуюче значення, параметри та якого вони виду, кількісні чи якісні.

Містить у собі приватний метод private void DataFromString(string str), який використовується для опрацювання вхідного текстового рядка і вилучення з нього всієї необхідної інформації для створення об’єкта.

* Rule.

Цей клас зберігає правило, яке створюється у вузлі і визначає наступний розподіл дерева. Екземпляр цього класу зберігає інформацію про помилку, яку дає цей розподіл, аргумент, який був задіяний у розподілі та інформацію про те кількісний цей аргумент чи якісний, а також саме правило. Також даний клас реалізує алгоритм визначення, яке правило найкраще розіб’є вибірку елементів, що знаходяться у вузлі і використовує цей алгоритм при ініціалізації екземпляра класу.

Містить у собі методи:

Публічні:

* public void DivideSampleByQualitiveRule(out Data[], out Data[], Data[], List<double>, int) – цей метод задіяний в реалізації алгоритму пошуку оптимального правила для вузла дерева, а також використовується при створені нащадків у вузлі для розподілу множини елементів на дві підмножини. Працює з якісними параметрами.
* public void DivideSampleByNotQualitiveRule(out Data[], out Data[], Data[], List<double>, int) – цей метод задіяний в реалізації алгоритму пошуку оптимального правила для вузла дерева, а також використовується при створені нащадків у вузлі для розподілу множини елементів на дві підмножини. Працює з кількісними параметрами.

Приватні:

* private Rule ChooseRule(Data[] elements) – «головний» метод у класі. Реалізує вибір оптимального правила для вузла дерева на основі заданої множини елементів. Викликається при ініціалізації. Викликає методи CreateRulesForQualitativeArguments і CreateRulesForNotQualitativeArguments залежно від якісних та кількісних параметрів відповідно.
* private List<Rule> CreateRulesForQualitativeArguments(Data[], int) – на основі заданої множини елементів створює множину усіх можливих правил заданих визначеним якісним аргументом. Використовує методи GeneralMethods.CountError та DivideSampleByQualitiveRule.
* private List<Rule> CreateRulesForNotQualitativeArguments(Data[] elements, int indexOfArgument) – на основі заданої множини елементів створює множину усіх можливих правил заданих визначеним кількісним аргументом. Використовує методи GeneralMethods.CountError, CreateValueForRule, SortValuesOfArgument та DivideSampleByNotQualitiveRule.
* private double CreateValueForRule(double[], int) – цей метод використовується тільки для кількісних аргументів, створює значення, яке можна розмістити у вузлі дерева в якості правила.
* private Rule FindRuleWithTheSmallestError(List<Rule> rules) – використовується, коли всі можливі правила вже сформовані. Серед них обирає правило, яке дає найменшу похибку при розподілі.
* Статичний клас CountError.

Цей клас зберігає в собі спільні допоміжні методи. Наприклад метод для підрахунку помилки у «листі» дерева

public static double CountError(Data[] elements), який використовується як класом DecisionTree, так і класом Rule.

# Висновки

В ході виконання роботи були вивчені такі поняття як регресія, дерева рішень та дерева регресії.

Була створена консольна програма, що будує дерева регресії, використовуючи алгоритм CART. Вона працює з багатомірними наборами даних, опрацьовує як кількісні, так і якісні параметри.

В подальшому планується вдосконалити програму наступним чином:

* вдосконалити побудову дерева етапом «відрізання» малозначущих «гілок»;
* реалізувати інший алгоритм побудови дерева регресії і порівняти результати;
* реалізувати графічний інтерфейс;
* реалізувати логіку для обробки об’єктів з пропущеними параметрами.

# Список використаної літератури

1. Основные задачи регрессионного анализа. Особенности классической линейной модели множественной регрессии.[Електроний ресурс] – Режим доступу: <http://metr-ekon.ru/index.php?request=full&id=273>
2. Шахиди Акобир, Деревья решений — общие принципы работы. [Електроний ресурс] – Режим доступу: <https://basegroup.ru/community/articles/description>
3. Дерево принятия решений. [Електроний ресурс] – Режим доступу: <https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D0%BF%D1%80%D0%B8%D0%BD%D1%8F%D1%82%D0%B8%D1%8F_%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B9>
4. Топ-10 data mining-алгоритмов простым язиком. [Електроний ресурс] – Режим доступу: <https://habrahabr.ru/company/iticapital/blog/262155/>
5. Использование деревьев решений в задачах прогнозной аналитики. [Електроний ресурс] – Режим доступу: <http://www.prognoz.ru/blog/platform/decision-tree-in-predictive-analytics/>
6. Г.А. Мельников, В.В. Губарев. Метод построения деревьев регрессии на основе муравьиных алгоритмов. [Електроний ресурс] – Режим доступу: <http://old.tusur.ru/filearchive/reports-magazine/2014-34-4/11.pdf>
7. Регресійний аналіз. [Електроний ресурс] – Режим доступу: <https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7>
8. MACHINE LEARNING ALGORITHMS. M5. [Електроний ресурс] – Режим доступу: <http://trymachinelearning.com/machine-learning-algorithms/decision-tree/m5/>
9. Иван Андреев. Деревья решений — CART математический аппарат. Часть 1. [Електроний ресурс] – Режим доступу: <https://basegroup.ru/community/articles/math-cart-part1>