|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»** |
| **РТУ МИРЭА** |
| Институт кибербезопасности и цифровых технологий |
| *(наименование института, филиала)* |
|  |
| Кафедра информатики |
| *(наименование кафедры)* |

**КУРСОВАЯ РАБОТА**

| по дисциплине | **Основы машинного обучения** |
| --- | --- |
|  | *(наименование дисциплины)* |

| **Тема курсовой работы** | **Реализация алгоритмов решающих деревьев** |
| --- | --- |
|  | |

| **Студент группы** | БФБО-05-22 Нахамкин Константин Дмитриевич |  |  |
| --- | --- | --- | --- |
|  | *(учебная группа, фамилия, имя, отчество студента)* |  | *(подпись студента)* |

| **Руководитель курсовой работы** | зав. каф., к.ф.-м.н., доцент, Шмелева А.Г. |  |  |
| --- | --- | --- | --- |
|  | *(должность, звание, ученая степень, ФИО)* |  | *(подпись руководителя)* |

| **Рецензент** (при наличии) | – |  | – |
| --- | --- | --- | --- |
|  | *(должность, звание, учёная степень)* |  | *(подпись рецензента)* |

Работа представлена к защите « » 2023 г.

Допущен к защите « » 2023 г.

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»** |
| **РТУ МИРЭА** |
| Институт кибербезопасности и цифровых технологий |
| *(наименование института, филиала)* |
| Кафедра информатики |
| *(наименование кафедры)* |

Утверждаю

заведующий кафедрой информатики

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шмелева А.Г.

*Подпись ФИО*

«\_\_\_\_» \_\_\_\_\_\_\_ 20\_\_\_ г.

**ЗАДАНИЕ**

**на выполнение курсовой работы по дисциплине**

**«Основы машинного обучения»**

Студент Нахамкин Константин Дмитриевич Группа БФБО-05-22

**Тема** «Реализация алгоритмов решающих деревьев»

**Исходные данные: Credit Risk Classification Dataset**

**Перечень вопросов, подлежащих разработке, и обязательного графического материала: изучение библиотек NumPy, Pandas, Matplotlib, Sklearn языка программирования Python, методы построения решающих деревьев, решение прикладной задачи анализа датасета, представление результата в виде значений и графиков.**

**Срок предоставления к защите курсового проекта (работы):**  до «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

**Задание на курсовой проект (работу) выдал \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

*Подпись руководителя Ф.И.О. руководителя*

«**01**» **марта**  **2023** г. ***Задание на курсовой проект (работу) получил (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)***

*Подпись обучающегося Ф.И.О. исполнителя*.

**Оглавление**

[**Введение 4**](#_heading=h.ho92oy4j8j1s)

[**1. Теоретические основы алгоритмов реализации решающих деревьев 6**](#_heading=h.30j0zll)

[1.1. Основные определения и формулы метода решающих деревьев 6](#_heading=h.1fob9te)

[1.1.1 Дерево решений и алгоритмы его построения 6](#_heading=h.hcqarwbskuyp)

[1.1.2 Индекс Джини 8](#_heading=h.z9dizzlr2i2g)

[1.1.3 Обрезка дерева(Pruning) 9](#_heading=h.a6re443xiagx)

[1.2. Подробное решение модельной задачи 10](#_heading=h.3znysh7)

[**2. Программная реализация решающих деревьев 14**](#_heading=h.2et92p0)

[2.1. Описание датасета и постановка задачи 14](#_heading=h.tyjcwt)

[2.2. Программное решение 15](#_heading=h.3dy6vkm)

[2.3 Анализ полученных данных. Определение погрешности 16](#_heading=h.kwrr83s3w0xa)

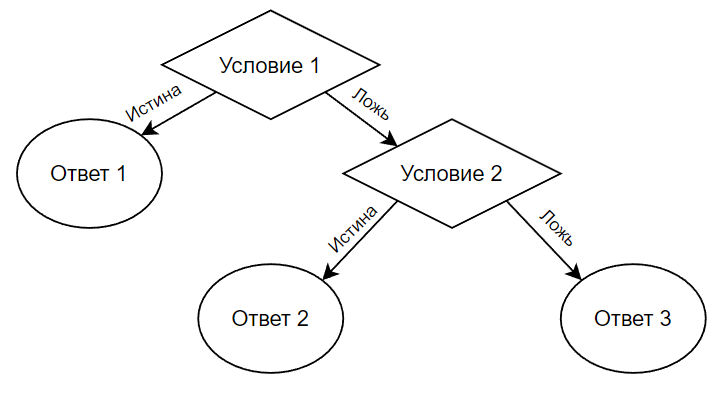
[**Заключение 17**](#_heading=h.1t3h5sf)

[**Список использованных источников 18**](#_heading=h.4d34og8)

[**Приложение 1? 19**](#_heading=h.2s8eyo1)

# Введение

Дисциплина «Основы машинного обучения» направлена на получение навыков создания прикладных программ и реализации алгоритмов на высокоуровневом объектно-ориентированном языке программирования Python. Язык программирования Python является одним из наиболее популярных и востребованных языков, применяемых для решения прикладных задач анализа данных как в крупных компаниях, так и стартапах. Популярность данного языка обеспечивает наличие достаточно большого количества модулей и библиотек, а также бесплатной среды разработки. Деревья решений – популярный и универсальный инструмент интеллектуального анализа данных и предсказательной аналитики, использующийся для решения широкого класса задач. Они представляют из себя графическое представление процесса принятия решений, которое моделирует результаты и предсказывает будущие события. Деревья решений состоят из узлов, представляющих принятие решения, листьев, представляющих окончательные результаты, и веток (ребер графа), соединяющих их друг с другом. Схематическое изображение представлено на рисунке 1.

Рисунок 1 – Пример дерева решений

Деревья решений широко используются в различных областях, включая науку о данных, машинное обучение и искусственный интеллект. Они часто используются при анализе решений для определения наилучшего курса действий с учетом набора возможных результатов и связанных с ними вероятностей. Деревья решений также используются в задачах классификации и регрессии для классификации данных или прогнозирования конкретного результата на основе набора входных переменных.

Построение дерева решений включает в себя процесс, называемый рекурсивным разделением, когда данные разбиваются на более мелкие подмножества на основе наиболее значимого признака. Этот признак выбирается на основе его способности разбивать данные на подмножества, максимально схожие с точки зрения переменной результата. Есть множество алгоритмов выбора признака для следующего ветвления(ID3, C4.5, CART), но в основном все они основаны на какой-либо формуле понижения энтропии.

Целью данной работы является подробное ознакомление с решающими деревьями и алгоритмами их построения, а также их программная реализация на большом датасете.

# 1. Теоретические основы алгоритмов реализации решающих деревьев

## 1.1. Основные определения и формулы метода решающих деревьев

### 1.1.1 Дерево решений и алгоритмы его построения

[Дерево](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)) — это [связный](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84) [ациклический граф](https://ru.wikipedia.org/wiki/%D0%90%D1%86%D0%B8%D0%BA%D0%BB%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B3%D1%80%D0%B0%D1%84). Связность означает наличие маршрута между любой парой вершин, ацикличность — отсутствие циклов. Отсюда, в частности, следует, что число рёбер в дереве на единицу меньше числа вершин, а между любыми парами вершин имеется один и только один путь.

[Дерево принятия решений](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B9#:~:text=%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE%20%D0%BF%D1%80%D0%B8%D0%BD%D1%8F%D1%82%D0%B8%D1%8F%20%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B9%20(%D1%82%D0%B0%D0%BA%D0%B6%D0%B5%20%D0%BD%D0%B0%D0%B7%D1%8B%D0%B2%D0%B0%D1%8E%D1%82,%D0%BB%D0%B8%D1%81%D1%82%D1%8C%D1%8F%C2%BB%20%D0%B8%20%C2%AB%D0%B2%D0%B5%D1%82%D0%BA%D0%B8%C2%BB.) (в данной работе будет использоваться именно дерево классификации) — средство поддержки принятия решений, использующееся в [машинном обучении](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), [анализе данных](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и [статистике](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0). Структура дерева представляет собой листья и ветки. На рёбрах(ветках) дерева решения записаны признаки, от которых зависит целевая функция, в листьях записаны значения [целевой функции](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%B5%D0%B2%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F), а в остальных вершинах — признаки, по которым различаются случаи. Чтобы классифицировать новый случай, надо спуститься по дереву от корня до одного из листов и выдать соответствующее значение.

Очень сложно сразу же взять и определить критерии классификации элементов. Чтобы это сделать, используют различные алгоритмы. Например:

* Алгоритм ID3, относительно простой и быстрый алгоритм.
* Алгоритм C4.5 (улучшенная версия ID3).
* Алгоритм CART, более универсальный и .

Все эти алгоритмы имеют свои особенности, но в основе имеют один и тот же алгоритм, который показан на блок-схеме(Рисунок 2).

Выбор алгоритма не всегда прост и основывается на различных критериях: поставленная задача(классификации или регрессии), характеристики желаемого дерева(например глубина и ширина), данные(их размер, наличие в них пропусков, их тип), время, требуемое на построение дерева и т.д. Но для небольших проектов зачастую используется алгоритм, встроенный в используемый инструмент.

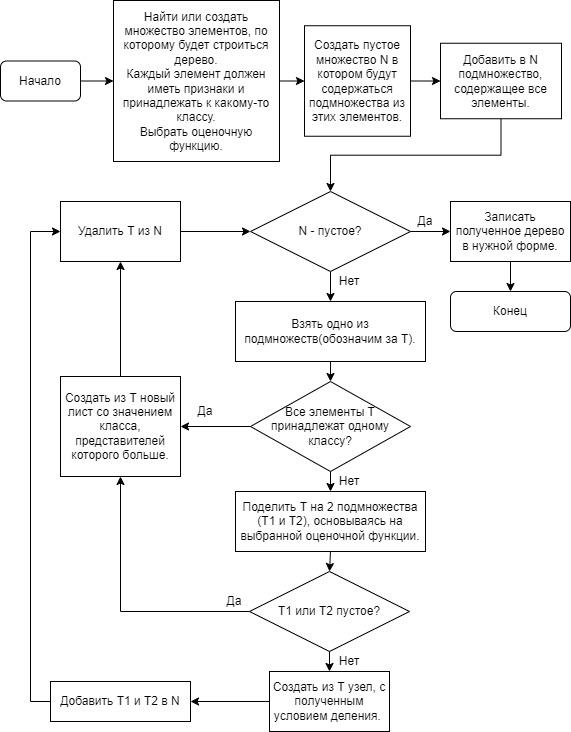


Рисунок 2

Для программного решения задачи будут использоваться методы класса DecisionTreeClassifier библиотеки sklearn, в котором для построения дерева применяется модифицированная версия алгоритма CART(Classification And Regression Tree). Рассмотрим его основные особенности:

* Использование индекса Джини в качестве оценочной функции
* Проведение обрезки готовых деревьев(англ. “Pruning”) — процесс удаления маловажных вершин, для предотвращения переобучения.
* Создание исключительно бинарных деревьев
* Возможность работы с порядковыми и номинальными переменными
* Возможность построения деревьев как классификации, так и регрессии

Рассмотрим самые важные из них.

### 1.1.2 Индекс Джини

Индекс Джини для одного множества элементов T вычисляется следующим образом:

,

где n — это количество классов в Т, — частота появления класса i в Т. Если множество Т мощностью N разбить на две части и с числом элементов в каждом и соответственно, тогда показатель качества разбиения можно найти как:

.

Наилучшим считается то разбиение, для которого минимально. Обозначим и — число экземпляров i-го класса в левом/правом потомке. Тогда качество разбиения оценивается по следующей формуле:

.

Для упрощения вычислений формулу можно упростить:

.

Так как при минимизации константы не играют роли, можно сократить N:

.

Или же:

.

Таким образом, лучшим будет разбиение с максимальным или минимальным .

### 1.1.3 Обрезка дерева(Pruning)

Алгоритмы обрезки делятся на два основных типа: работающие во время и после построения дерева. Алгоритмы ,использующиеся во время построения, обычно ограничивают какой-либо параметр, не давая дереву расти. Например максимальную глубину или минимальное количество элементов в листе. Алгоритмы, применяющиеся на уже готовом дереве, как правило дают оценку каждому узлу и удаляют все, не соответствующие заданным критериям.

Здесь будет описан алгоритм “[Minimal Cost-Complexity Pruning](https://scikit-learn.org/stable/modules/tree.html#bre)”, который используется для оптимизации деревьев решения в библиотеке sklearn.

Для каждого дерева T задается критерий стоимости-сложности (cost-complexity measure), зависящий от параметра (по-умолчанию ).

,

где — взвешенный индекс Джини (может использоваться и другая формула, например ошибка классификации), — количество листьев в дереве. Из этой формулы можно вывести критерий стоимости-сложности отдельного узла(t):

.

Если — дерево, полученное сразу после деления узла t, то, при , , но при определенном значении они будут равны. Обозначим это значение как . Его можно найти приравняв предыдущие уравнения:

.

Так как алгоритм CART создает только бинарные деревья, то

.

Это и есть оценочный параметр в данном алгоритме. Все узлы, которых меньше параметра ccp\_alpha, который задается при создании дерева, будут удалены.

Этот метод обрезки деревьев является относительно времязатратным, но в то же время эффективным.

## 1.2. Подробное решение модельной задачи

Рассмотрим алгоритм построения дерева решений на небольшом датасете. Он был сгенерирован при помощи чат-бота ChatGTP. В нем содержится 10 элементов, у каждого из которых есть 3 параметра. Первые два из них числовые, а третий — категориальный. Каждый элемент также принадлежит к одному из двух классов, указанном в столбце Label (Рисунок 3).

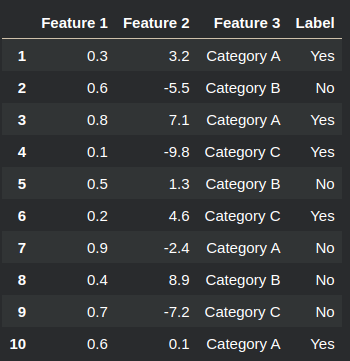
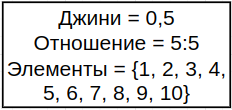


Рисунок 3

Для начала можно найти индекс Джини начального состояния (корневой вершины), чтобы убедиться, что с ростом дерева он уменьшится.

.

Наше дерево на данном этапе выглядит следующим образом:

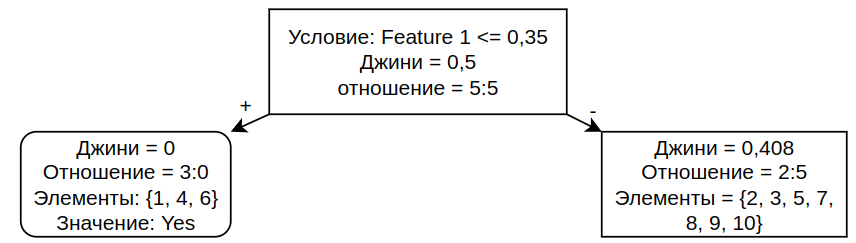


Методом перебора находим условие, минимизирующее . В данном случае это . Элементы с индексами 1, 4, 6 уйдут в левую вершину, а остальные в правую.

.

В ходе вычислений так же были получены индексы Джини новых вершин: 0 и 0,285 для левой и правой соответственно. Так как в левой вершине содержатся представители только одного класса, она становится листом со значением “Yes”, а последующие деления будут касаться только правого поддерева.

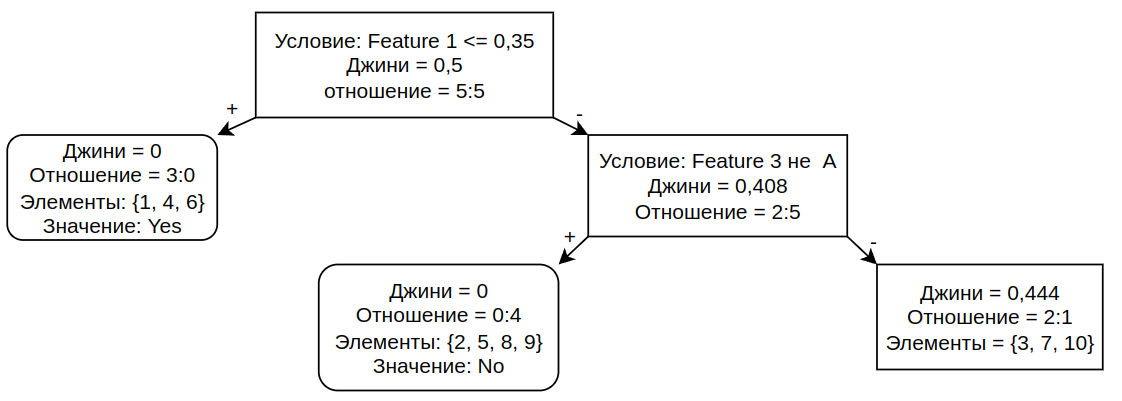
После этого деления дерево выросло:



Следующим условием будет .

.

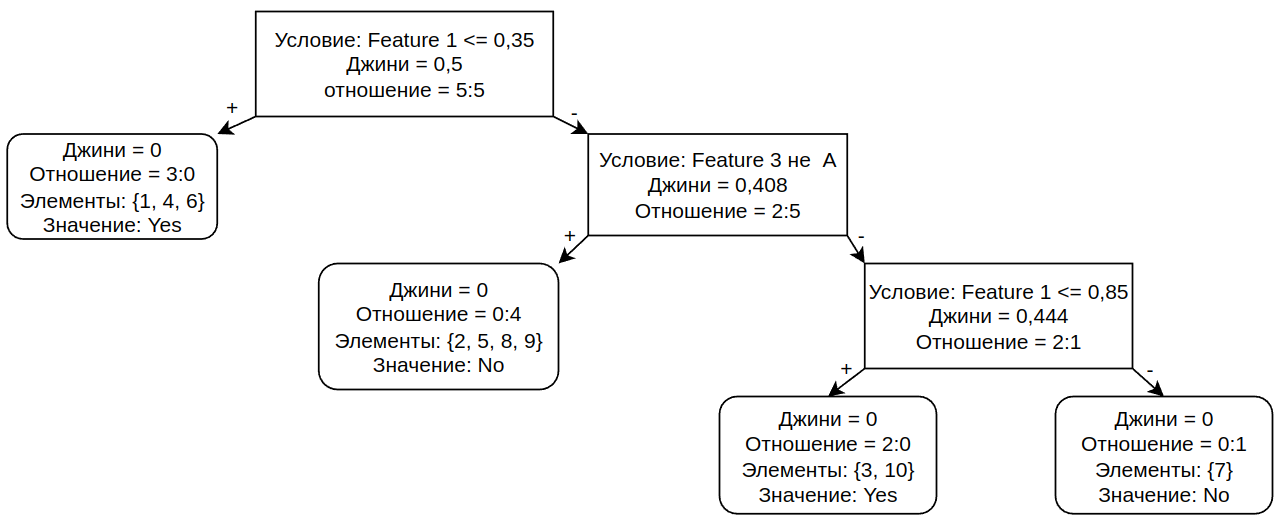
В ходе деления так же образуется новый лист и узел.



Последним условием деления будет . Оно делит узел на два, в каждом из которых индекс Джини равен 0.

0

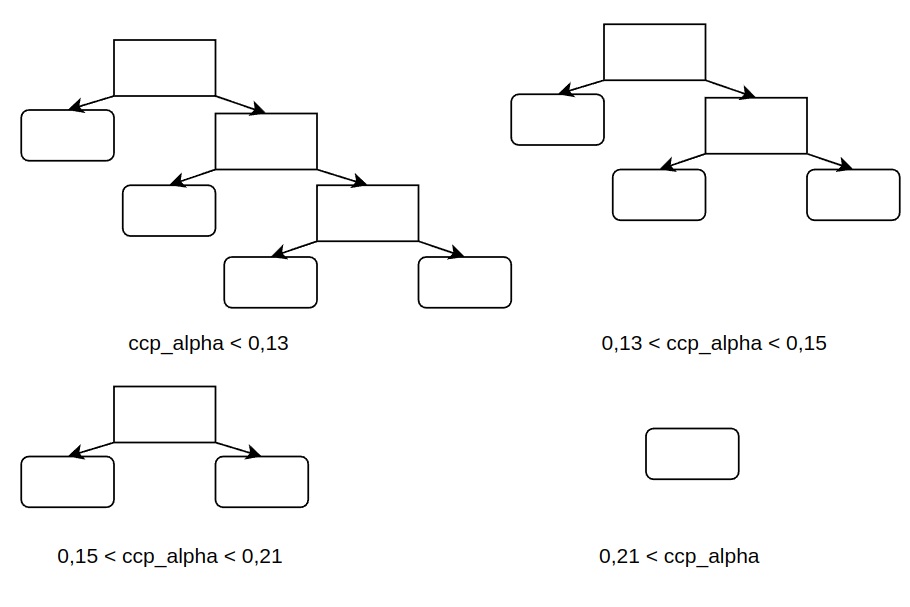
Тогда окончательный вид дерева будет следующим:



Итоговым значением индекса Джини для дерева будет сумма произведений индексов Джини каждого листа на отношение количества элементов в нем к общему количеству элементов в дереве. Так как в каждом листе элементы принадлежат одному и тому же классу, это значение будет равно 0, что меньше начального .

Теперь рассчитаем значение для каждого узла снизу вверх и посмотрим какой вид будет принимать дерево при различных ccp\_alpha:

1. ,
2. ,
3. .



Если бы этот датасет был разделен на обучающую и тестовую выборки, то можно было бы найти оптимальное значение ccp\_alpha, подставляя его из каждого интервала и оценивая ошибку на тестовых значениях. Этот процесс будет представлен в программном решении задачи, так как он не представляет большой пользы для маленьких выборок и выглядит более понятно на больших.

# 2. Программная реализация решающих деревьев

## 2.1. Описание датасета и постановка задачи

pass

## 2.2. Программное решение

pass

## 2.3 Анализ полученных данных. Определение погрешности

pass

# Заключение

pass

# Список использованных источников

1. <https://online-edu.mirea.ru/pluginfile.php?file=%2F1044700%2Fmod_resource%2Fcontent%2F0%2F%D0%A0%D0%B0%D0%B1%D0%BE%D1%87%D0%B0%D1%8F%20%D1%82%D0%B5%D1%82%D1%80%D0%B0%D0%B4%D1%8C%202_%D0%9E%D0%9C%D0%9E_2_%D1%81%D0%B5%D0%BC%D0%B5%D1%81%D1%82%D1%80.pdf>
2. <https://online-edu.mirea.ru/pluginfile.php?file=%2F1047978%2Fmod_resource%2Fcontent%2F0%2F%D0%9B%D0%B5%D0%BA%D1%86%D0%B8%D1%8F%202%20%D0%9F%D0%BE%D1%81%D1%82%D1%80%D0%BE%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D1%8C%D0%B5%D0%B2.pdf>
3. <https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B9>
4. <https://towardsdatascience.com/build-better-decision-trees-with-pruning-8f467e73b107>

# Приложение 1?

Листинг программы?