# МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования   
**«Кубанский государственный университет»**

**(ФГБОУ ВО «КубГУ»)**

**Кафедра вычислительных технологий**

# **Курсовая работа**

# **Исследования методов принятия решений**

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Барулина

(подпись, дата)

Факультет компьютерных технологий и прикладной математики курс 3

Направление 02.03.02 – «Фундаментальная информатика и информационные технологии»

Научный руководитель к.т.н. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.А. Приходько

(подпись, дата)

Нормоконтролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

Краснодар 2016

Оглавление

[МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ 1](#_Toc194254250)

[**Курсовая работа** 1](#_Toc194254251)

[**Исследования методов принятия решений** 1](#_Toc194254252)

[ВВЕДЕНИЕ 3](#_Toc194254253)

[1. ОБЗОР МЕТОДОВ ПРИНЯТИЯ РЕШЕНИЙ 4](#_Toc194254254)

[1.2. Методы интеллектуального анализа данных 4](#_Toc194254255)

[2. ДЕРЕВЬЯ РЕШЕНИЙ 6](#_Toc194254256)

[2.1. Алгоритмы деревьев решений 8](#_Toc194254257)

[2.2. Пример построения дерева решений 16](#_Toc194254258)

[2.3. Преимущества применения деревьев решений 18](#_Toc194254259)

[3. РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ АЛГОРИТМА 19](#_Toc194254260)

[3.1. Обоснование выбора языка программирования 24](#_Toc194254261)

[3.2. Тестирование 26](#_Toc194254262)

[ЗАКЛЮЧЕНИЕ 29](#_Toc194254263)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 30](#_Toc194254264)

# ВВЕДЕНИЕ

Ежедневно мы отправляем множество e-mail сообщений, выкладываем фото в свои профили, а так, же делимся видео с друзьями и коллегами, пользуемся средствами для мониторинга здоровья, шагомеры и прочие гаджеты. Все эти действия и приборы генерируют данные, которые нужно где-то хранить и обрабатывать для извлечения полезной информации. Многие задумывались над решением этой проблемы, одним их них стал Григорий Пятецкий-Шапиро, который в 1989 году ввел термин Data Mining.

Актуальный компьютерный термин Data Mining пока не имеет, устоявшегося перевода на русский язык, поэтому чаще всего переводится, как «извлечение информации» или «добыча данных». Цель Data Mining состоит в выявлении незаметных правил и закономерностей в больших объемах данных. Данный термин обозначает не один, а совокупность огромного числа методов анализа данных. Выбор метода часто зависит от типа имеющихся данных и оттого, какой результат мы хотим получить. Методы Data Mining позволяют решить большое количество различных задач, с которыми сталкивается аналитик по данным.

Целью данной работы является изучение интеллектуальных методов анализа данных, в том числе методов принятия решений, подробный разбор и моделирование некоторых из них.

# 1. ОБЗОР МЕТОДОВ ПРИНЯТИЯ РЕШЕНИЙ

Процесс принятия решений как человеком, так и машиной основывается на интеллектуальном анализе данных. Результатом интеллектуального анализа является выявление в совокупности данных моделей и тенденций, которые помогают принимать решения. Принципы интеллектуального анализа данных известны в течение многих лет, но с появлением *больших данных* они получили еще более широкое распространение.

## 1.2. Методы интеллектуального анализа данных

На рис.1. представлен список ставших уже классическими методов интеллектуального анализа данных (ИАД), а также список средств, которые используются для ИАД.

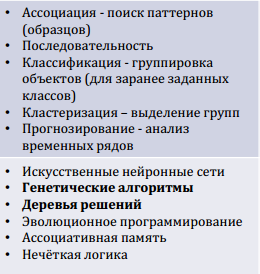


Рис. 1. Методы интеллектуального анализа данных

Рассмотрим подробнее:

1. Задача ассоциации, при поиске ассоциативных правил необходимо найти частные зависимости между объектами или событиями. Найденные зависимости представлены в виде правил и могут быть задействованы в предсказании появления событий.
2. Задача классификации, сводится к определению класса объекта по его характеристикам.
3. Задача регрессии, позволяет определить по известным характеристикам объекта значение нужного параметра.

4) Задача кластеризации ищет независимые группы и их характеристики во множестве имеющихся данных. Данная задача помогает лучше разобраться с данными. Кроме того, при группировке однотипных объектов позволяет сократит их число, следовательно, облегчить упростить анализ.

5) Последовательные шаблоны, служат для установления закономерности между связанными событиями, т.е. если произойдет событие А, то спустя заданное время произойдет событие В.

6) Анализ отклонений, выявляет наиболее нехарактерные шаблоны.

Для решения задач необходимо знать некоторые методы и алгоритмы Data Mining. Рассмотрим наиболее популярные методы:

1. Кибернетические методы (искусственные нейронные сети и др.)

2. Логические методы (Символьные правила, деревья решений, генетические алгоритмы и др.)

3. Методы на основе уравнений

4. Методы кросс-табуляции (Бейсовские сети, агенты)

и др.

# 2. ДЕРЕВЬЯ РЕШЕНИЙ

Рассмотрим подробнее логические методы (символьные правила, деревья решений, генетические алгоритмы) они оформляют найденные закономерности, таким образом, что для пользователя имеет почти прозрачный вид. Заметим, что деревья решений могут быть быстро преобразованы в наборы символьных правил путем изменения одного правила по пути от корня дерева до его терминальной вершины. Из всех логических методов реализация дерева решений, является наиболее эффективным и быстрым.

На данный момент аналитики по знаниям уделяют методу деревьев решений большое внимание т.к. он является наиболее удобным для использования на этапе выбора альтернатив. Деревья решений широко применяются для решения практических задач в некоторых областях:

* Банковское дело: оценка кредитоспособности клиента банка при выдаче кредита.
* Промышленность: контроль качества продукции.
* Медицина: диагностика различных заболеваний.
* Молекулярная биология: анализ строения аминокислот.

И это далеко не весь список областей, где применяются деревья решений. Многие сферы только начинают внедрять данный метод в работу. А, так же в областях, где велика цена ошибки, этот метод послужит отличным помощником аналитику или руководителю.

Методы деревьев решений разрешают автоматизацию процесса принятия некоторых видов решений. В процессе развития сферы IT-технологий, а конкретно, прогресс в методах сбора, хранения и обработки информации позволил многим предприятиям собирать большие массивы данных, которые необходимо обработать и принять верное решение.

Объем данных бывает настолько велик, что возможности анализировать потоки данных вручную, доставляет огромную проблему, сводящуюся к почти невозможному принятию верного решения. Традиционная статистика, которая долгое время считалась основным инструментом анализа данных, так же не дает полной картины. Она, как правило, использует средние характеристики выборки, которые часто являются очень усреднёнными значениями. Поэтому методы математической статистики, имеют место быть, но только для проверки заранее сформулированных гипотез. Это проблема и повлияла на спрос на методы автоматического анализа данных. С каждый годом, информации становится больше, а, следовательно, и спрос постоянно растет.

Первые идеи создания деревьев решений были положены еще в XX веке. Однако, это не дало особых результатов, большой толчок был дан книгой Ханта, Мэрина и Стоуна «Experiments in Induction» выпущенной в 1966 году.

Суть метода, а также характеристики и особенности построения дерева решений исследуются в работах многих ученных таких как: М.Мексон, М. Альберт, Ф.Хедоури, М.Эддоуса, Р.Стэнсфилд, А.Кабушкина, и др.

Однако на данный момент в науке еще не сформирована теоретико-методологическая база разработки решений методом дерева решений. Зачастую работы, которые посвящены разработке метода, рассматривают лишь частные случаи и не дают описания в целом.

Для дальнейшего рассмотрения деревьев решений, нам понадобятся следующие термины.

***Варианты*** – разные способы решения поставленной задачи, средство достижения целей.

***Критерий выбора альтернатив*** – определение порядка предпочтения вариантов.

***Решение*** – выбор из нескольких альтернатив или определения порядка предпочтений из нескольких вариантов.

***Числовой признак*** – признак, значения которого могут являться только числа.

***Номинальный признак*** – это признак, значения которого являются перечисляемыми и известными до начала работы с ними

***Допустимое решение*** – решение, удовлетворяющее ограничением, но не всегда являющиеся самым эффективным.

***Эффективное решение*** – решение, являющееся самым оптимальным по нескольким критериям.

***Возможные исход*** – возможные итоги после реализации нескольких альтернатив.

***Объект*** – пример, шаблон, наблюдение.

***Атрибут*** – признак, свойство объекта или класса.

***Метка класса*** – зависимая переменная, основная переменная, признак или свойство, определяющие класс объекта.

***Узел*** – внутренний узел дерева, узел проверки.

***Лист*** – Конечный узел дерева, узел результата.

***Проверка*** – условие в узле.

## 2.1. Алгоритмы деревьев решений

Дерево решений – один из способов представления информации, в виде различных уровней и связей. Зачастую, строится по иерархическому принципу:

* + - главная цель помещается в корень дерева
    - методы достижения этой цели являются ветвями деревьев (первого, второго и последующих уровней).

Основное отличие деревьев решений от других методов заключается в том, что проводимое исследование основывается на логических рассуждениях и вычислениях. Деревья решений – это один из методов построенных на основе правил вывода. Эти системы часто называют системами прямого логического вывода, так как, анализ начинается с фактов, а в результате получаем тот или иной вывод.

Логический порядок построения дерева решений состоит в следующем:

в наиболее простом виде дерево решений – это способ представления правил в иерархической, последовательной структуре. Основа данной структуры являются ответы «Да» или «Нет» на различные вопросы.

Стоит отметить, что бинарные деревья являются самым простым, частным случаем деревьев решений, так как могут иметь только 2 ветви от узла. В остальных деревьях, ответов и, соответственно, ветвей дерева, выходящих из внутреннего узла, может быть больше двух.

В методе построения решений, есть несколько алгоритмов выбора очередного атрибута.

Рассмотрим некоторые из них:

1. Алгоритм ID3 – выбор атрибута происходит на основание увеличения информации, либо на основании индекса Gain.
2. Алгоритм С4.5 (усовершенствованная версия ID3) – выбор атрибута базируется на основании нормализованного прироста информации.
3. Алгоритм CART – алгоритм предназначен для работы с бинарными деревьями, и делит на каждом шаге ответы напополам: по одной ветви выполняются примеры, не удовлетворяющие условию, а по другой удовлетворяющие.
4. Алгоритм CHAID (Хи-квадрат) – алгоритм выполняет многоуровневое разделение при расчете классификации деревья.
5. Алгоритм Mars – расширяет деревья решений для улучшения обработки цифровых данных.

Рассмотрим их подробнее:

* Алгоритм ID3

Возьмем, например, атрибут, который принимает 3 значения: X, Y, Z.

При разбиении множества алгоритм создаст три узла T1(X), T2(Y), T3(Z), в первый из них будет помещены все записи со значение Х, во второй Y, в третий С.

Процедура повторяется рекурсивно до тех пор, пока не останутся только примеры одного класса, после чего они будут объявлены листами и ветвление прекратиться. Самым проблемным этапом является выбор атрибута, по которому производится разбиение. Алгоритм ID3 справляется с этим недугом с помощью увеличение информации или уменьшение энтропии.

Далее, рассмотрим математическую модель алгоритма.

Пусть Т множество текстов, а их количество – мощность данного множества |T|. Множество классов будет обозначаться, как С={C1, C2, … ,Ck}, а множество признаков A={A1, A2, … ,Am}.

По каждому признаку Аi можно разбить множество Т на подмножества Т1,Т2 , … , Тn.

Пусть F(Cj, T) – количество текстов из некоторого множества Т, лежащих в одном классе Cj . Тогда вероятность того, что случайным образом выбранный текст окажется из множество Т и принадлежащим классу Сj .

(1)

Тогда энтропия множества Т имеет вид:

(2)

Условная энтропия множества текстов Т при рассматриваемом признаке Х есть:

(3)

После для каждого из признаков вычисляется объем информации

I(X) = H(T) – H(T|X) (4)

Существует 2 варианта разбиения.

Если Аi – номинальный признак, то количество значение признака Аi, будет соответствовать количеству подмножеств Т.

Если Аi - числовой признак, то множество Т разбивается на два подмножества. При это необходима выбрать порог разбиения, по которому будут сравниваться все значения признака.

v = {v1, v2, … , vn}

Для начала следует отсортировать значения. Тогда значение, лежащие между vi  и vi+1, делит все значения на два множества и в качестве порога можно выбрать среднее значение между vi  и vi+1.

(5)

Следует, можно сделать вывод, что имеется n-1 потенциальное пороговое значение. Так как для номинального признака имеется всего один вариант разбиения, а для числового признака количество вариантов разбиения равно количеству порогов. Если имеется u номинальных признаков и v числовых, то в каждой вершине разбиения можно расписать O способами, где

O = u + v(n-1). (6)

* Алгоритм С4.5

Представляет собой усовершенствованный вариант алгоритма ID3. Отличие появляется в критерии разбиения множества на подмножества.

Критерий разбиения (6) имеет недостаток, он выбирает признаки, которые имеют много значений, так как при разбиении по такому признаку получаются подмножества, содержащие минимальное число текстов. Проблему можно решить при помощи некоторой нормализации.

, (7)

*Где*

Х = {T1, T2, … , Tn} и . (8)

Выражение (10) оценивает потенциальную информацию, полученную при разбиении множества Т на подмножества n.

Тогда критерием разбиения будет

(9)

* Алгоритм CART

CART (Classification And Regression Tree) – переводится как «Дерево Классификации и регрессии». Существует также несколько модифицированных версий IndCART(отличается использованием другого способа обработки пропущенных значений, имеет другие параметры отсечения) и BD-CART(вместо того чтобы использовать обучающий набор данных для определения разбиений, использует его для оценки распределения входных и выходных данных, а затем использует эту оценку для разбиения). В данном алгоритме любой узел дерева имеет двух потомков. На каждом шаге построения дерева атрибут, делит множество примеров на две части – часть, в которой выполняется правило (потомок - right) и часть, в которой правило не выполняется (потомок - left). Для выбора оптимального атрибута необходимо использовать функцию оценки качества разбиения.

Оценочная функция базируется на интуитивной идее уменьшения неопределенности в узле. (Например, рассмотрим пример с двумя классами и узлом, имеющим по 60 примеров одного класс, если найти разбиение делящие данные на две подгруппы 50:5 в одной и 10:55 в другой, то «нечистота уменьшится», но полностью она исчезнет, когда результатом будет 60:0 в одной и 0:60 в другой). В алгоритме CART эта идея реализована в индексе Gini. Если набор данных T содержит данные n классов, тогда индекс Gini определяется, таким образом:

(10)

*где параметр – вероятность класса i в T.*

Существует так же индекс Gini для ситуации, когда набор T разбивается на две части T1 и Т2 с числом примеров в каждом N1 иN2, тогда показатель будет равен:

. (11)

Отметим, что наилучшим считается разбиение, для которого индекс получается минимальным.

Правило разбиения в алгоритме CART схоже с алгоритмов ID3. Если переменная числового типа, то в узле формируется правило xi <= c (где с-некоторый порог, который чаще всего формируется, как среднеарифметическое двух соседних значений в множестве). Если переменная категориального типа, то в узле формируется правило xi V(xi), где V(xi) – некоторое непустое подмножество множества значений переменной xi в обучающей выборке. Таким образом, для n значений числового атрибута алгоритм сравнивает n-1 разбиений, а для категориального (2n-1 – 1). На каждом шаге построения алгоритм сравнивает все возможные варианты и выбирает лучший атрибут и наилучшие разбиение.

Механизм отсечения дерева, оригинальное название minimal cost-complexity tree pruning, - этот механизм является значимым отличие алгоритма CART от других. Отсечение решает сразу две значительные проблемы:

* получение дерева оптимального размера;
* получение точной оценки вероятности ошибки.

Обозначим |T| - число листов дерева, R(T) – ошибка классификации дерева. Определим полную стоимость дерева Т так:

, (12)

*где α – некоторый параметр (возможно изменение от 0 до +∞).*

Полная стоимость дерева имеет две составляющие – ошибку классификации дерева и штраф за его сложность. Можно заметить, что с увеличением параметра α будет расти полная стоимость дерева. Поэтому в зависимости от α, менее ветвистое дерево, дает большую классификацию, может стоить меньше, чем дерево более ветвистое, но имеющие меньшую ошибку.

Определим – максимальное по величине дерево, которое предстоит обрезать. Если задать значение α, то будем иметь наименьшее поддерево Т(α), для которого выполняются несколько условий:

* Условие сообщает, о том, что не существует такого поддерева , которое имело бы наименьшую стоимость, чем при заданном значение α;
* Если существует больше одного поддерева, имеющих полную стоимость, то мы выбираем наименьшее дерево.

Несмотря на то, что параметр α имеет бесконечное множество значений, существует конечное множество поддеревьев дерева , поэтому можно создать последовательность поддеревьев:

T1 > T2 > T3 >...> {t1}, (13)

*(где t1 – корневой узел дерева) такую, что – наименьшее поддерево для α ∈ [αk, αk+1).*

Первое дерево в этой последовательности – наименьшее поддерево дерева имеющую такую же ошибку классификации, получим, что . Другими словами, если разбиение идет до тех пор пока в каждом узле не останется только один класс, то .

Выбор финального дерева заключается в выборе лучшего дерева из последовательности деревьев. Стоит отметить, что при отсечении ветвей, дерева использовались только первоначальные данные (если быть точнее, то даже не сами данные, а количество примеров, каждого класса). Зачастую, наиболее эффективным вариантом проверки финального дерева является тестирование на тестовой выборке (качество тестирование в таком случае зависит от объема тестовой информации). Иногда можно наблюдать некоторые ошибки в составление, чтобы уменьшить эту нестабильность, CART использует (1 - ) – правило: выбирается минимальное по размеру дерево с в пределах интервала [min , min +], где - ошибка классификации дерева, а – стандартная ошибка, являющаяся оценкой ошибки реальной:

, (14)

*где - число примеров.*

Алгоритм CART хорошо сочетает в себе результативность построенных моделей, а, так же, высокую скорость их построения. Так же имеет свои уникальные методы обработки пропущенных значений и построения оптимального дерева.

* Алгоритм CHAID (Хи-квадарат)

Чтобы построить дерево решений с помощью данного алгоритма, нужны зависимая (имеет две категории) и независимая (используется для разделения атрибутов на группы) переменные.

CHAID основан на критерии Хи-квадрат, которые позволяет нам определять, связаны ли статистически две переменные. Алгоритм определяет, как лучше сгруппировать категории каждой независимой переменной, так чтобы значение Хи-квадрат было максимальным. Это позволяет определить насколько независимые переменные отличны друг от друга.

Таким образом, получаем дерево, листьями которого являются группы с максимально различными значениями зависимой переменной. По такому дереву легко определить в какой из группы интересующий нас признак максимален.

## 2.2. Пример построения дерева решений

Рассмотрим на простом примере общие принципы построения дерева решений «Выдать ли кредит?».

Внутренние узлы дерева (возраст, доход и образование) эти атрибуты называются атрибутами расщепления или прогнозирующими. Листы или конечные узды дерева, являются метками класса, которые определяют итог «выдать» или «не выдать» кредит.

Каждая ветвь дерева, идущая от внутреннего узла, отмечена предикатом расщепления. Существует особенность предикатов расщепления: каждая запись использует единственный и уникальный путь от корня дерева только к одному узлу-решению. Объединенная информация об атрибутах расщепления в узле называется критерием расщепления.

В начале выбираем атрибут Q (возраст >35) и поместить его в корневой узел. Затем, просматриваем пример и для каждого значения атрибута i (в нашем случае чаще всего встречается «да» и «нет») выбираем только те, для которых Q=i. Далее, рекурсивно строим дерево принятия решений.

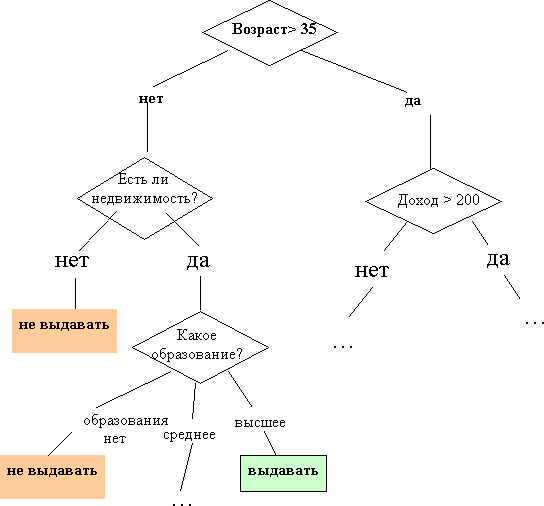


Рис. 3 Дерево решений "Выдавать ли кредит?" [1]

Основная проблема, чаще всего, кроется в первом шаге – по какому принципу выбирается каждый следующий атрибут Q. Для этого сформировали несколько частных алгоритмов принятия решений.

## 2.3. Преимущества применения деревьев решений

Метод деревьев решений является одним из важных инструментов в работе многих специалистов, занимающихся анализом данных, благодаря значительному количеству достоинств:

* + - Прост для восприятия. Результат построения легко понимается пользователем. Наглядно показывает, почему тот или иной объект отнесен именно к этому классу;
    - Алгоритм построения не требует выбора входных атрибутов. Для построения применяются все атрибуты, алгоритм сам вычисляет наиболее значимые на их основе и строится дерево решений;
    - Гибкость. Деревья решений позволяют работать с непрерывными и символьными целевыми признаками, что позволяет деревьям решений применятся в самых различных задачах;
    - организация правил в сферах, где эксперту трудно формализовать свои знания;
    - интуитивно понятная классификационная модель;
    - достаточно высокая точно прогноз;
    - Для построения требуется небольшой объем информации, поэтому он занимает мало места в памяти.

# 3. РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ АЛГОРИТМА

Рассмотрим пример построения дерева принятия решений.

Предположим, что нас интересует, выиграет ли команда «ФКТиПМ» свой следующий матч. Результат зачастую зависит от ряда параметров, возьмем несколько основных, так как:

* выше или ниже в турнирной таблице находится соперник;
* пропускают ли матч, основные(лидеры) игроки команды;
* погодные условия;
* дома ли проводится матч.

Допустим у нас есть некоторая статистика, по итогам прошлых игр, из которых были получены данные указанные в табл. 1.

Таблица 1. Как играет «ФКТиПМ»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Соперник | Место пр. матча | Погодные условия | Лидеры | Победа |
| Выше | Дома | Отрицательные | На месте | Нет |
| Выше | Дома | Положительные | На месте | Да |
| Выше | Дома | Положительные | Пропускают | Да |
| Выше | В гостях | Отрицательные | На месте | Нет |
| Ниже | Дома | Положительные | Пропускают | Да |
| Ниже | В гостях | Положительные | Пропускают | Нет |
| Ниже | Дома | Отрицательные | Пропускают | Да |
| Ниже | В гостях | Положительные | На месте | ? |

Зная, эти факторы попробуем составить дерево принятия решений. В узлах, не являющихся листьями, находятся атрибуты, по которым различают случаи. По ребрам будем спускаться чтобы классифицировать имеющиеся случаи. Далее просто записываем атрибуты в порядке, указанном в табл. 1. Тогда у нас получается дерево, изображенное на рис. 4.

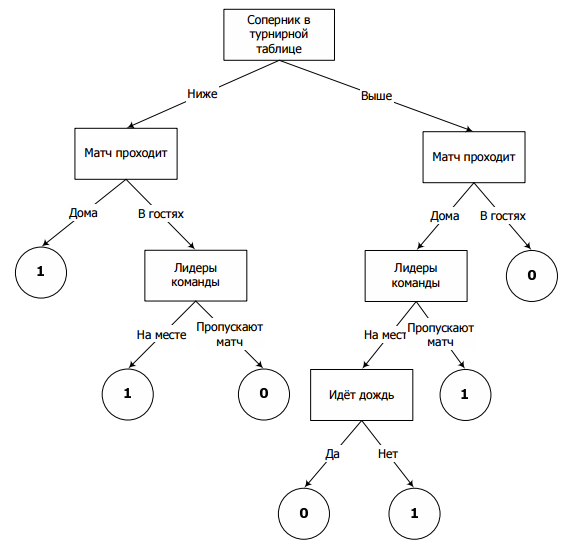


Рис. 4. Вариант дерева принятия решений

Построенное дерево, конечно, дает правильный результат, но не идеально т.к. его глубина равна четырем. Чтобы это исправить можно взять в качестве корня другой атрибут, например, поместим туда вопрос о погоде. В случае если, погода положительна, то следующим атрибутом будет положение соперника в турнирной таблице, а в случае, отрицательной погоды, будет смотреть на атрибут проходит ли матч дома. В этом случае его глубина будет равна двум.

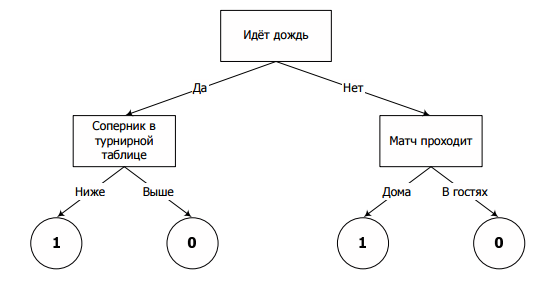


Рис 5. Оптимальное дерево принятия решений

Теперь нужно обучить дерево, так, чтобы оно вбирало оптимальное решение. Это требование формализуется посредствам энтропии.

Определим оптимальный атрибут, вычислим исходную энтропию

(15)

Теперь определим приросты информации для различных атрибутов:

Из данных вычислений отчетливо видно, что мы выбрали не слишком удачный атрибут для корня дерева.

Теперь сведем все то, о чем рассуждали в единый рекурсивный алгоритм для построения дерева принятия решений.

Для реализации возьмем алгоритм ID3 на языке Python.

Рассмотрим псевдокод алгоритма:

ID3 (A,S,Ϭ) , где А – множество элементов, S – свойство, которому должны удовлетворять некоторые элементы множества А, Ϭ – множество, в котором хранится прирост информации, каждого атрибута.

1. Создать корень дерева
2. Если условие S выполняется на всех элементах A, поставить в корень метку 1 и выйти.
3. Если S не выполняется ни на одном элементе А, поставить в корень метку 0, выйти.
4. Если Q= Ø, то:
5. если S выполняется на большей части или половине множества А, поставить в корень метку 1 и выйти
6. если S выполняется на части, которая меньше половины A, поставить в корень метку 0 и выйти.
7. Выбрать Q ∈ Ϭ, для которого Gain(A, Q) максимален.
8. Метку Q поставить в корень.
9. Для каждого значения q атрибут Q:
10. добавить нового потомка корня и пометить соответствующее исходящее ребро меткой q;
11. если в А нет примеров, для которых Q принимает значение q, то пометить этого потомка в зависимости от того, на какой части А выполняется S (возврат к пункту 4);
12. иначе реализовать ID3(A, S, Ϭ \ {Q} ) и добавляем его результат, как поддерево с корнем в этом потомке.

У данного алгоритма реализации, есть несколько значительных преимуществ таких как:

* обработке ситуации, когда одному и тому же набору атрибутов соответствуют несколько случаев с разными исходами (за решение данной проблемы отвечает пункт 4);
* обработке ситуации, когда у атрибута может встретиться несколько вариантов, к тому же, может так произойти, что какой-то из этих вариантов не реализуется, в таком случае мы заполняем соответствующий лист в зависимости от того, каких исходов было больше в его предке (за это отвечает пункт 7b).

Программа обрабатывает только бинарные атрибуты. Рассмотрим формат входного файла:

* число атрибутов n;
* строки формата, количество строк равно n ([Название атрибута], [Положительное значение], Отрицательное значения]);
* название основного атрибута;
* m – число тестовых примеров;
* строки формата ([атрибут1 = значение1], …, [атрибутn  = атрибутn]).

## 3.1. Обоснование выбора языка программирования

Перед началом выполнения поставленной задачи были проанализированы зарубежные и отечественные источники на тему: «Популярность языков анализа данных». В соответствии с веб-сайтом американской компании по бизнес-аналитике, где проводились голосования по выбору лучшего программного обеспечения для анализа данных в 2012, 2013, 2014 годах. Топ-3 голосов составили языки R, SAS, Python.

Таблица 2. Сравнительная характеристика средств анализа данных [8]

|  |  |
| --- | --- |
| **What programming/statistics languages you used for an analytics / data mining / data science work in 2014?** | |
| **Language used** | http://www.kdnuggets.com/polls/graph/g0.gif % voters in 2014 (719 total)  http://www.kdnuggets.com/polls/graph/g4.gif % voters in 2013 (713 total)  http://www.kdnuggets.com/polls/graph/g8.gif % voters in 2012 (579 total) |
| R (352 voters in 2014) | http://www.kdnuggets.com/polls/graph/g0.gif 49.0% http://www.kdnuggets.com/polls/graph/g4.gif 60.9% http://www.kdnuggets.com/polls/graph/g8.gif 52.5% |
| SAS (262) | http://www.kdnuggets.com/polls/graph/g0.gif 36.4% http://www.kdnuggets.com/polls/graph/g4.gif 20.8% http://www.kdnuggets.com/polls/graph/g8.gif 19.7% |
| Python (252) | http://www.kdnuggets.com/polls/graph/g0.gif 35.0% http://www.kdnuggets.com/polls/graph/g4.gif 38.8% http://www.kdnuggets.com/polls/graph/g8.gif 36.1% |
| SQL (220) | http://www.kdnuggets.com/polls/graph/g0.gif 30.6% http://www.kdnuggets.com/polls/graph/g4.gif 36.6% http://www.kdnuggets.com/polls/graph/g8.gif 32.1% |
| Java (89) | http://www.kdnuggets.com/polls/graph/g0.gif 12.4% http://www.kdnuggets.com/polls/graph/g4.gif 16.5% http://www.kdnuggets.com/polls/graph/g8.gif 21.2% |

Окончание табл. 2

|  |  |
| --- | --- |
| Unix shell/awk/sed (63) | http://www.kdnuggets.com/polls/graph/g0.gif 8.8% http://www.kdnuggets.com/polls/graph/g4.gif 11.1% http://www.kdnuggets.com/polls/graph/g8.gif 14.7% |
| Pig Latin/ Hive/ other Hadoop-based languages (61) | http://www.kdnuggets.com/polls/graph/g0.gif 8.5% http://www.kdnuggets.com/polls/graph/g4.gif 8.0% http://www.kdnuggets.com/polls/graph/g8.gif 6.7% |
| SPSS (58) | http://www.kdnuggets.com/polls/graph/g0.gif 8.1% not asked not asked |
| MATLAB (45) | http://www.kdnuggets.com/polls/graph/g0.gif 6.3% http://www.kdnuggets.com/polls/graph/g4.gif 12.5% http://www.kdnuggets.com/polls/graph/g8.gif 13.1% |
| Scala (28) | http://www.kdnuggets.com/polls/graph/g0.gif 3.9% http://www.kdnuggets.com/polls/graph/g4.gif 2.2% http://www.kdnuggets.com/polls/graph/g8.gif 2.4% |
| C/C++ (26) | http://www.kdnuggets.com/polls/graph/g0.gif 3.6% http://www.kdnuggets.com/polls/graph/g4.gif 9.3% http://www.kdnuggets.com/polls/graph/g8.gif 14.3% |

Для моделирования дерева решений автором работы был выбран именно Python 3, так как в данном языке есть огромный выбор уже встроенных функций, который помогают при написании кода для поставленной задачи, а также потому что язык Python хорошо зарекомендовал себя при решении задач ИАД.

Язык программирования Python 3 – это хороший инструмент для создания программ разнообразного значения. У него достаточно много преимуществ, таких как:

* Интерпретируемый язык программирования;
* Значительная поддержка модульности, что позволяет написать свой модуль;
* Поддержка объектно-ориентированного программирования, а реализация является одной из самых понятных;
* Отсутствие утечек памяти;
* Понятный и лаконичный синтаксис;
* И др.

## 3.2. Тестирование

Запишем в файл input.txt информацию, содержащуюся в табл. 1.

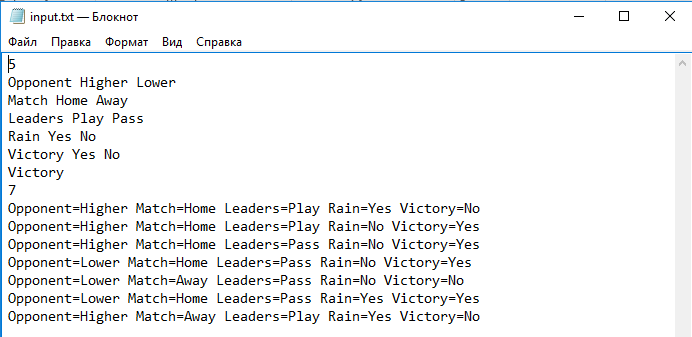


Рис. 6 Данные подающиеся на вход программы

На вход программы подается файл, который считывается функцией applyID3. Вторым аргументом служит имя файла, куда программа записывает результат своей работы. Результатом является дерево, уровни которого отделяются табуляциями.

def applyID3(infname,outfname):

bigarr = ParseAttributes(infname)

attrnum,attrnames,attr,tests,num=bigarr[0],bigarr[1],bigarr[2],bigarr[3],  
 bigarr[4]

f = open(outfname,'w')

usedattr=[]

for i in xrange(attrnum): usedattr.append(i==num)  
 ID3(tests,attrnum-1,f,0,usedattr,attrnames,attr)

Функция entropy(tests,num), находит исходную энтропию, с помощью которой в дальнейшим вычисляется простота информации для различных атрибутов.

def entropy(tests,num):

import math

def log2(x): return math.log(x)/math.log(2)

neg = float(len(filter(lambda x:(x[num]==0),tests))) //кол-во отрицательных вариантов

tot = float(len(tests)) //кол-во различных вариантов

if ((neg==tot) or (neg==0)): return 0

return -(neg/tot)\*log2(neg/tot)-((tot-neg)/tot)\*log2(tot-neg)

Далее функция gain(tests,attrnum,num) вычисляет прирост информации каждого атрибута.

Далее с помощью основной функции под названием ID3 на каждом шаге просчитывается исходная энтропия, прирост информации, каждого атрибута, после выбирается атрибут с самым большим приростом на данном шаге.

Я не очень разобралась с этим куском программы

def ID3(tests,num,f,tabnum,usedattr,attrnames,attr):

def findgains(x):

if usedattr[x]: return 0 return gain(tests,x,num)

if (len(tests)==0): f.write('\t'\*tabnum+'1')

return

if len(filter(lambda x:(x[num]==0),tests))>len(filter(lambda x:(x[num]==1),tests)):

majority = '0'

else: majority = '1'

gains = map(findgains,xrange(len(tests[0])))

maxgain = gains.index(max(gains))

if (gains[maxgain]==0):

f.write('\t'\*tabnum+majority+'\n')

return

arrpos=filter(lambda x:(x[maxgain]==1),tests)

arrneg=filter(lambda x:(x[maxgain]==0),tests)

newusedattr=usedattr

newusedattr[maxgain]=True

f.write('\t'\*tabnum+attrnames[maxgain]+'='+attr[attrnames[maxgain]][1]+'\n')

if (len(arrpos)==0): f.write('\t'\*(tabnum+1)+majority+'\n')

else:

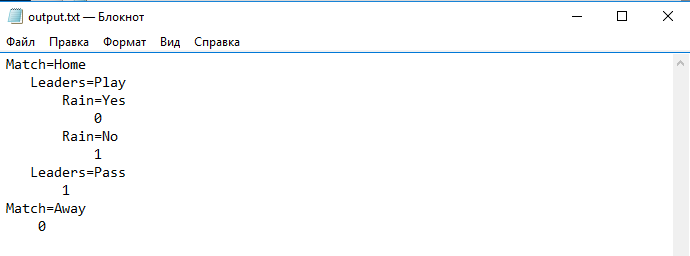
ID3(arrpos,num,f,tabnum+1,newusedattr,attrnames,attr)

f.write('\t'\*tabnum+attrnames[maxgain]+'='+attr[attrnames[maxgain]][2]+'\n')

if (len(arrneg)==0): f.write('\t'\*(tabnum+1)+majority+'\n')

else: ID3(arrneg,num,f,tabnum+1,newusedattr,attrnames,attr)

После мы записываем результаты с помощью уже знакомой функции applyID3 в файл output.txt.



## ЗАКЛЮЧЕНИЕ

Дерево принятия решений – распространенное средство интеллектуального анализа данных, используемое принятия решений на основе имеющихся альтернатив. Оно позволяет представить поставленную задачу схематично и сравнить альтернативы визуально. Этот метод можно использовать в применении к сложным ситуациям, когда достаточно много альтернатив и трудно сделать выбор. Методы деревьев решений находят свое применение в робототехнике и во многих других задачах нахождения решений в пространстве состояний.

Качество работы рассмотренного метода деревьев решений зависит как от выбора алгоритма, так и от набора исследуемых данных. На вход алгоритму можно подавать любые входные данные, он сам выбирает наиболее подходящие из них для решения проблемы. В сравнении, например, с нейронными сетями (выбор количества входных данных влияет на время работы), это значительно облегчает пользователю работу.

В ходе исследования курсового проекта я узнала, что такое деревья принятия решений, для чего они нужны, проанализировала существующие алгоритмы и выполнила реализацию одного из методов на языке Python.

В перспективе планируется выполнить моделирование нескольких различных алгоритмов, сравнить результаты их работы, проанализировать эффективность их применения в интеллектуальных системах принятия решения.

## СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. J Ross Quinlan. C4.5: Programs for Machine learning. // Morgan Kaufman Publishers, 1993.
2. Левитин А.В. Глава 10. Ограничения мощи алгоритмов: Деревья принятия решений // [Вильямс](https://www.wikidata.org/wiki/Q21694521), 2006. — С. 409—417. — 576 с.
3. L.Breiman, J.H. Friedman, R.A. Olshen, C.T. Stone Classification and Regression Trees // Belmont, 1984.
4. Деревья решений - общие принципы работы [Электронный ресурс] Статья, URL: <https://basegroup.ru/comunity//description> [Дата обращения: 5 ноября 2016].
5. Методы построения деревьев решений в задачах классификации в Data Mining [Электронный ресурс] Cтатья, URL: <http://ami.nstu.ru/~vms/lecture/data_mining/trees.htm> [Дата обращения: 15 ноября 2016].
6. Технологии принятия решений [Электронный ресурс] Cтатья, URL: <http://citforum.ru/consulting/BI/resolution/#структуры> [Дата обращения: 15 ноября 2016].
7. Описание алгоритма CART [Электронный ресурс] Cтатья, URL: <http://iamdrunk.ru/teach/src/278_andreev.pdf> [Дата обращения: 15 ноября 2016].
8. Four main languages for Analytics, Data Mining, Data Science. [Электронный ресурс] Статья, URL:http://[kdnuggets.com/2014/08/four-main-languages-analytics-data-mining-data-science.html](http://www.kdnuggets.com/2014/08/four-main-languages-analytics-data-mining-data-science.html) [Дата обращения: 1 декабря 2015].