Лекция 5. Решающие деревья

Определение дерева

Дерево – одна из наиболее широко распространённых структур данных в информатике, эмулирующая древовидную структуру в виде набора связанных узлов. Является связным графом, не содержащим циклы. Большинство источников также добавляют условие на то, что рёбра графа не должны быть ориентированными. В дополнение к этим трём ограничениям, в некоторых источниках указывается, что рёбра графа не должны быть взвешенными (рисунок 1).

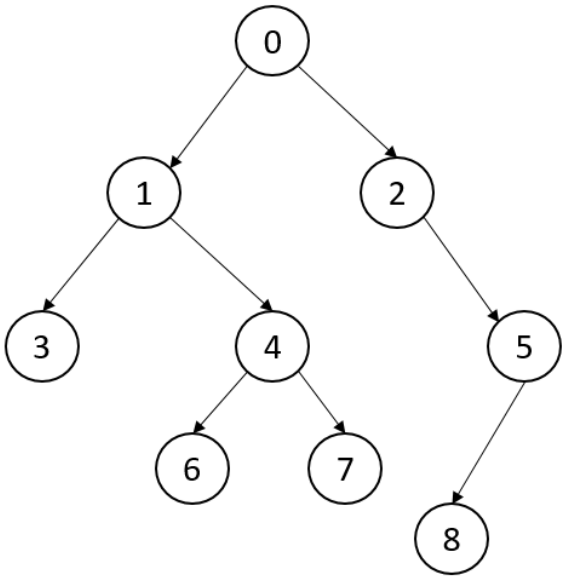

Рисунок 1 – Простой пример дерева

Корневой узел – самый верхний узел дерева (узел 0 на рисунке 1).

Корень – одна из вершин, по желанию наблюдателя.

Лист, листовой или терминальный узел – узел, не имеющий дочерних элементов (узлы 3, 6, 7, 8).

Внутренний узел – любой узел дерева, имеющий потомков, и таким образом, не являющийся листовым узлом (1, 2, 4, 5).

Поддерево – часть древообразной структуры данных, которая может быть представлена в виде отдельного дерева. Любой узел дерева T вместе со всеми его узлами-потомками является поддеревом дерева T. Для любого узла

поддерева либо должен быть путь в корневой узел этого поддерева, либо сам узел должен являться корневым. То есть поддерево связано с корневым узлом целым деревом, а отношения поддерева со всеми прочими узлами определяются через понятие соответствующее поддерево.

Двоичное (бинарное) дерево – это древовидная структура данных, где каждый узел имеет не более двух детей. Этих детей называют левым (Л) и правым (П) потомком или «сыном».

Деревья решений

Деревья решений являются моделями, широко используемыми для решения задач классификации и регрессии. По сути, они задают вопросы и выстраивают иерархию правил «если… то», приводящую к решению.

Эти вопросы похожи на вопросы, которые вы можете спросить в игре «20 Questions». Представьте, вам нужно научиться отличать друг от друга четыре вида животных: медведей, ястребов, пингвинов и дельфинов. Ваша цель состоит в том, чтобы получить правильный ответ, задав несколько вопросов. Вы могли бы начать с вопроса, есть ли у этих видов животных перья, вопроса, который сужает количество возможных видов животных до двух. Если получен ответ «да», вы можете задать еще один вопрос, который может помочь вам различать ястребов и пингвинов. Например, вы могли бы спросить, может ли данный вид животных летать. Если у этого вида животных нет перьев, ваши возможные варианты – дельфины и медведи, и вам нужно задать вопрос, чтобы провести различие между этими двумя видами животных, например, спросить, есть ли плавники у этого вида животных. Эти вопросы можно выразить в виде дерева решений, как это показано на рисунке 2.

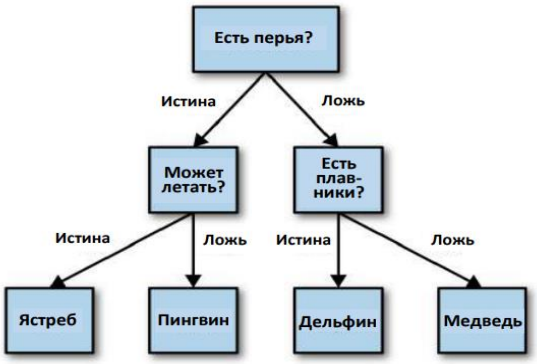

Рисунок 2 – Простой пример дерева принятия решений

Вы скорее всего уже использовали деревья решений для того, чтобы сделать какой-нибудь выбор в своей жизни. Например, нужно *решить*, чем вы займетесь на предстоящих выходных. Итог может зависеть от того, хотите ли вы пойти куда-то с друзьями или провести выходные в одиночестве. В обоих случаях решение зависит также и от погоды. Если она будет солнечной и друзья будут свободны, вы можете сходить поиграть в футбол. Если пойдет дождь, вы пойдете в кино. Если друзья будут заняты, то вы останетесь дома играть в видео игры, даже несмотря на хорошую погоду (рисунок 3).

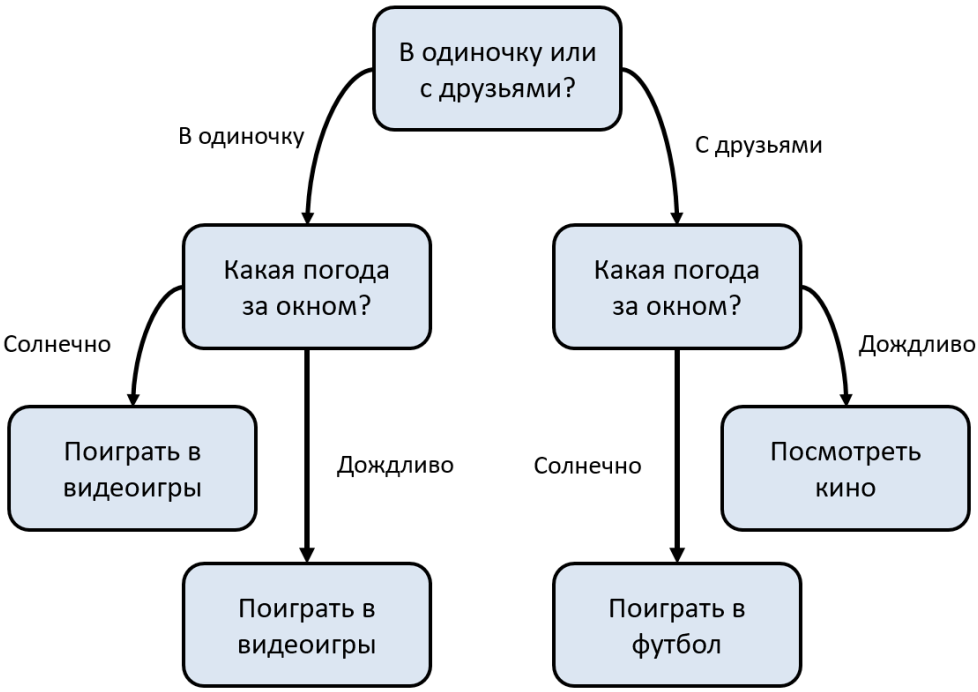

Рисунок 3 –Пример дерева принятия решений из реальной жизни

Этот пример хорошо демонстрирует дерево решений в реальной жизни. Мы построили дерево и смоделировали ряд последовательных, иерархических решений, которые, в конечном итоге, приводят к некоторому результату. Обратите внимание, что мы выбрали максимально общие варианты решения, чтобы дерево было маленьким. Дерево будет просто огромным, если мы установить много возможных вариантов погоды, например: 25 градусов, солнечно; 25 градусов, дождливо; 26 градусов, солнечно; 26 градусов, дождливо; 27 градусов, солнечно; 27 градусов, дождливо и т.д. Конкретная температура не важна. Нам нужно лишь знать, будет ли погода хорошей или нет.

В машинном обучении концепция деревьев решений такая же. Нам нужно построить дерево с набором иерархических решений, которые в конце приведут нас к результату, то есть к нашей классификации или прогнозу регрессии. Решения выбираются таким образом, чтобы дерево было максимально маленьким, но при этом сохраняло точность классификации или регрессии.

Деревья решений в машинном обучении

Модели дерева решений строятся в два этапа: индукция и отсечение. Индукция – это то, где мы строим дерево, то есть устанавливаем все границы иерархического решения, основываясь на наших данных. Из-за своего характера обучаемые деревья решений могут быть подвержены значительному переобучению. Отсечение – это процесс удаления ненужной структуры из дерева решений, эффективно упрощая его для понимания и избежания переобучения.

Индукция

Индукция дерева решений проходит 4 главных этапа построения:

1. Начните с обучающего набора данных, в котором должны содержаться признаки переменных и результаты классификации или регрессии.

2. Определите «лучший признак » в наборе данных для их разбиения. О том, как определить этот «лучший признак » поговорим позже.
3. Разбейте данные на подмножества, которые будут содержать возможные значения для лучшего признака. Такое разбиение в основном определяет узел на дереве, то есть каждый узел — это разделенная точка, основанная на определенном признаке из наших данных.

Рекурсивно сгенерируйте новые узлы дерева с помощью подмножества данных, созданных на 3 этапе. Продолжайте разбиение до тех пор, пока не достигните точки, на которой будет находится оптимизированная каким-то способом максимальная точность. Старайтесь минимизировать количество разбиений и узлов.

Первый этап простой. Просто соберите свой набор данных!

На втором этапе выбор признака и определенного разбиения обычно осуществляется с помощью жадного алгоритма для уменьшения функции стоимости. Если подумать, разбиение при построении дерева решений эквивалентно разбиению признакового пространства. Мы будем несколько раз пробовать разные точки разбиения, а в конце выберем ту, которая имеет наименьшую стоимость. Конечно, можно проделать пару умных вещей, таких как разбиение только в диапазоне значений в нашем наборе данных. Это позволит сократить количество вычислений для тестирования точек разбиений, которые являются заведомо бесполезными.

Для дерева регрессии можно использовать простой квадрат ошибки в качестве функции потерь:

$$E = \sum \left(Y - \hat{Y}\right)^2,$$

где Y — это достоверные данные, а $\hat{Y}$ — это прогнозируемое значение. Мы суммируем по всем выборкам в нашем наборе данных, чтобы получить общую ошибку. Для классификации мы используем *функцию коэффициента Джини*:

$$E = \sum \left(p_k(1 - p_k)\right),$$

где p_k — это доля обучающих примеров класса k в определенном узле прогнозирования.

В идеале узел должен иметь значение ошибки, равное нулю, означающее, что каждое разбиение выводит один класс 100% времени. Это именно то, что нам нужно, так как добравшись до конкретно этого узла принятия решения, мы будем знать, каким будет вывод в зависимости от того, на какой стороне границы мы находимся.

Такая концепция единственного класса на узел в наборе данных называется «прирост информации». Посмотрите на пример ниже (рисунок 4).

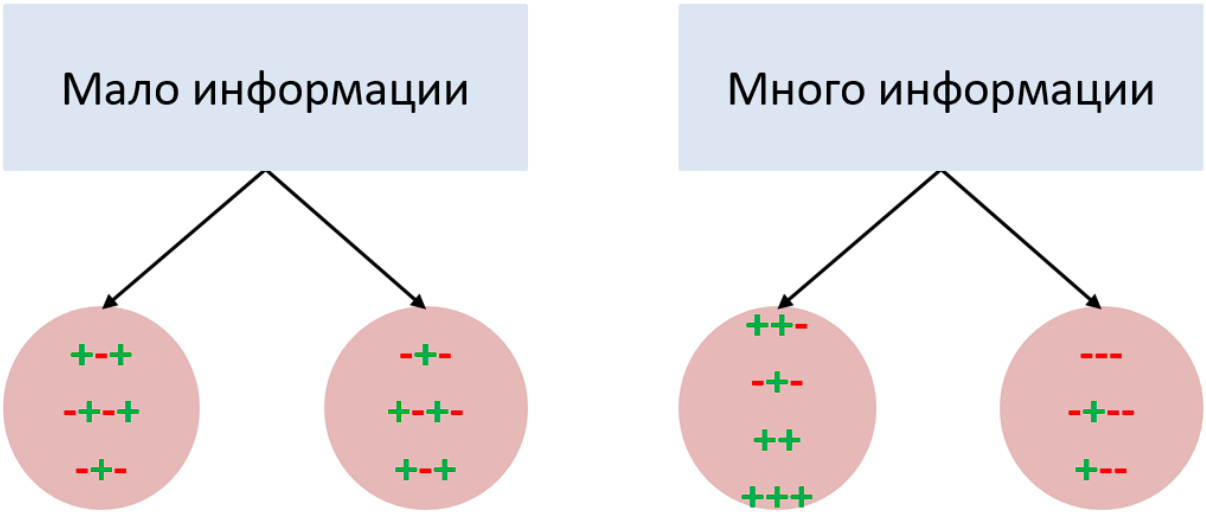

Рисунок 4 – Иллюстрация концепции «прирост информации»

Если бы нам пришлось выбрать разбиение, на котором каждый выход имеет разные классы в зависимости от входных данных, тогда бы мы не *добыли* никакой информации. Мы не узнали бы больше о том, влияет ли конкретный узел, то есть функция, на классификацию данных! С другой стороны, если наше разбиение имеет высокий процент каждого класса для каждого выхода, то мы *добыли* информацию о том, что разбиение таким конкретным образом на эту конкретную переменную дает нам конкретный выход!

Теперь мы могли бы продолжить разбиение до тех пор, пока у нашего дерева не появятся тысячи ветвей… Но это плохая идея! Наше дерево решений было бы огромным, медленным и переобученным для нашего обучающего набора данных. Поэтому мы установим некоторые заранее определенные критерии остановки, чтобы прекратить построение дерева.

Наиболее распространенным методом остановки является использование минимального расчета по количеству обучающих примеров, назначенных для каждой вершины дерева. Если число меньше некоторого минимального значения, то разбиение не считается и узел назначается конечной вершиной дерева. Если все вершины дерева становятся конечными, то обучение прекращается. Чем меньше минимальное число, тем точнее будет разбиение и, соответственно, вы получите больше информации. Но в таком случае минимальное число склонно к переобучению обучающими данными. Слишком большое количество минимальных чисел приведет к остановке обучения слишком рано. Таким образом, минимальное значение обычно устанавливается на основе данных в зависимости от того, сколько примеров ожидается в каждом классе.

Отсечение

Из-за своего характера обучающие деревья решений могут быть подвержены значительному переобучению. Выбор правильного значения для минимального количества примеров на узел может оказаться сложной задачей. Во многих случаях можно было бы просто пойти по безопасному пути и сделать этот минимум очень маленьким. Но в таком случае, у нас было бы огромное количество разбиений и, соответственно, сложное дерево. Дело в том, что многие из получившихся разбиений окажутся лишними и не помогут увеличить точность модели.

Отсечение ветвей дерева — это метод, сокращающий количество разбиений с помощью удаления, т.е. отсечения, ненужных разбиений дерева. Отсечение обобщает границы решений, эффективно уменьшая сложность

дерева. Сложность дерева решений определяется количеством разбиений. Рассмотрим метод «снизу вверх», для этого, необходимо:

1. Построить полное дерево (чтобы все листья содержали примеры одного класса).

2. Определить два показателя: относительную точность модели и абсолютную ошибку.

3. Удалить из дерева листья и узлы, отсечение которых не приведёт к значимому уменьшению точности модели или увеличению ошибки.

Простой, но очень эффективный метод отсечения происходит снизу вверх через узлы, оценивая необходимость удаления определенного узла. Если узел не влияет на результат, то он отсекается (рисунок 5).

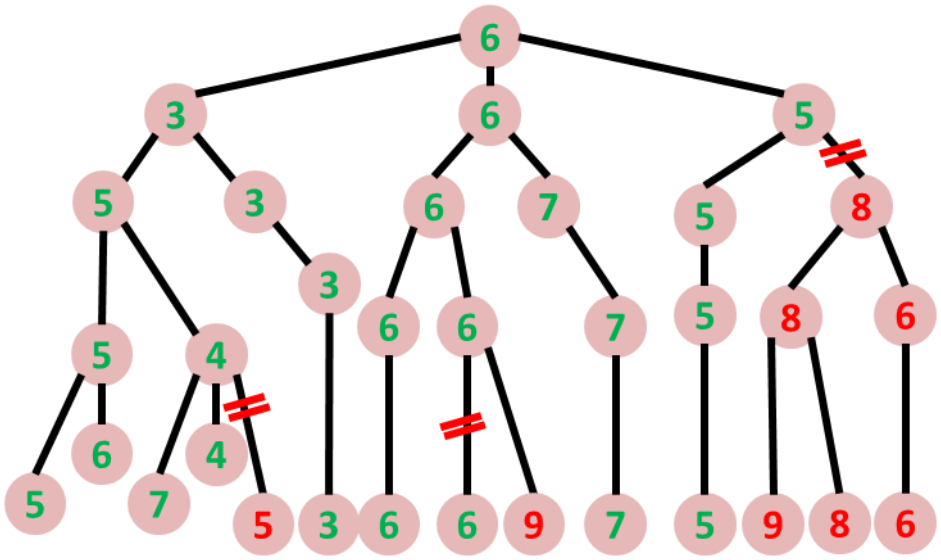

Рисунок 5 – Пример отсечения в дереве

Ансамбли деревьев

Ансамбли (ensembles) – это методы, которые сочетают в себе множество моделей машинного обучения, чтобы в итоге получить более мощную модель. Существует много моделей машинного обучения, которые принадлежат к этой категории, но есть две ансамблевых модели, которые доказали свою эффективность на самых различных наборах данных для задач классификации и регрессии, обе используют деревья решений в качестве строительных

блоков: случайный лес деревьев решений и градиентный бустинг деревьев решений.

При построении решающего дерева нам нужно вопросом делить исходные данные на две части. Пусть нужно разделить данные A на A_1 и A_2. Это нужно сделать наиболее информативно, т.е. чтобы энтропия $H(A_1) + H(A_2)$ была минимальной. Энтропию можно считать по Шеннону, но в задачах числовой регрессии удобнее использовать понятие дисперсии.

Общий вид алгоритма выглядит следующим образом:

1. Взять весь набор входных данных.
2. Вычислить энтропию целевой переменной.
3. Рассчитать прирост информации по всем атрибутам.
4. Выбрать атрибут с наибольшим объемом информации в качестве корневого узла.
5. Повторить ту же процедуру для каждой ветви, пока узел решения каждой ветви не будет завершён.

Рассмотрим пример построения дерева решений для задачи регресси. Пусть заданы начальные данные, представленные в таблице 1.

Таблица 1 – Начальные данные

X	1	2	9	4	5
Y	1	3	3	10	10

Пусть нужно разделить данные A на A_1 и A_2. Это нужно сделать наиболее информативно, т.е. чтобы энтропия $H(A_1) + H(A_2)$ была минимальной. Энтропию можно считать по Шеннону, но в задачах числовой регрессии удобнее использовать понятие дисперсии.

Пусть мы имеем набор чисел: A=(1, 3, 9, 10, 10).

Среднее значение:

E[A] = (1+3+9+10+10)/5 = 6.6.

Дисперсия:

$D(A)=[(1-6.6)^2+(3-6.6)^2+(9-6.6)^2+(10-6.6)^2+(10-6.6)^2]/5 = 14.64$.

Для построения дерева нам нужно задавать вопросы, которые будут делить выборку на две части. Вопросы могут быть только к переменной x: например, “x < 4?” и т.д.

Вопрос: “x < 4?” делит выборку на два множества $A_1 = (1, 3, 9)$ и $A_2 = (10, 10)$. Для этих множеств имеем:

$D(A_1)= [(1–4.33)^2+ (3–4.33)^2+ (9–4.33)^2]/3=11.55$

$D(A_2) = [(10-10)^2+(10-10)^2]/2 = 0.$

В сумме: 11.55.

Вопрос: “x < 3?” делит: A1=(1, 3) и A2=(9, 10, 10).

$D(A_1)=[(1-2)^2+(3-2)^2]/2 = 1;$

$D(A_2)=[(9-9.66)^2+(10-9.66)^2+(10-9.66)^2]/3 = 0.22;$

В сумме: 1.22 – это значительно лучше

Далее, необходимо повторить процедуру аналогично для оставшихся двух узлов: (x=2) и (x=4). Итоговый граф представлен на рисунке 6.

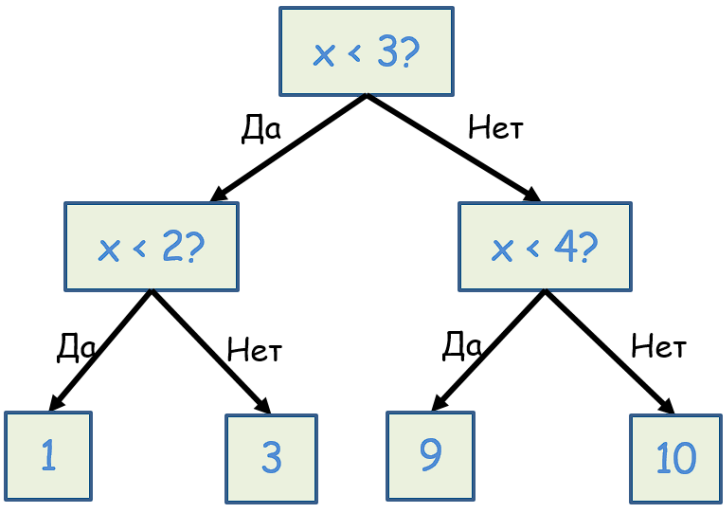

Рисунок 6 – Результирующее дерево

Случайный лес

Как мы только что отметили, основным недостатком деревьев решений является их склонность к переобучению. Случайный лес является одним из способов решения этой проблемы. По сути случайный лес – это набор деревьев решений, где каждое дерево немного отличается от остальных. Идея случайного леса заключается в том, что каждое дерево может довольно

хорошо прогнозировать, но скорее всего переобучается на части данных. Если мы построим много деревьев, которые хорошо работают и переобучаются с разной степенью, мы можем уменьшить переобучение путем усреднения их результатов. Уменьшение переобучения при сохранении прогнозной силы деревьев можно проиллюстрировать с помощью строгой математики.

Для реализации вышеизложенной стратегии нам нужно построить большое количество деревьев решений. Каждое дерево должно на приемлемом уровне прогнозировать целевую переменную и должно отличаться от других деревьев. Случайные леса получили свое название из-за того, что в процесс построения деревьев была внесена случайность, призванная обеспечить уникальность каждого дерева. Существует две техники, позволяющие получить рандомизированные деревья в рамках случайного леса: сначала выбираем точки данных (наблюдения), которые будут использоваться для построения дерева, а затем отбираем признаки в каждом разбиении.

В настоящее время случайные леса регрессии и классификации являются одним из наиболее широко используемых методов машинного обучения. Они обладают высокой прогнозной силой, часто дают хорошее качество модели без утомительной настройки параметров и не требуют масштабирования данных.

По сути, случайные леса обладают всеми преимуществами деревьев решений, хотя и не лишены некоторых их недостатков. Одна из причин, в силу которой деревья решений еще используются до сих пор, – это компактное представление процесса принятия решений. Детальная интерпретация десятков или сотен деревьев невозможна в принципе, и, как правило, деревья в случайном лесе получаются более глубокими по сравнению с одиночными деревьями решений (из-за использования подмножеств признаков). Поэтому, если вам нужно в сжатом виде визуализировать процесс принятия решений для неспециалистов, одиночное дерево решений может быть оптимальным выбором. Несмотря на то, что построение случайных лесов на больших

наборах данных может занимать определенное время, его можно легко распараллелить между несколькими ядрами процессора в компьютере (рисунок 7).

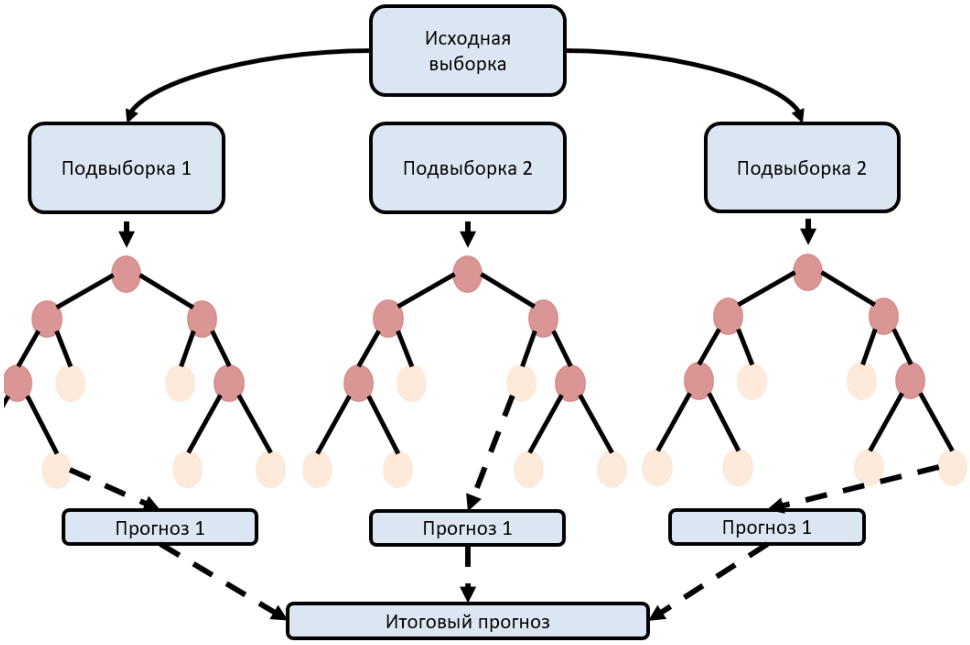

Рисунок 7 –Схема работы алгоритма «случайный лес»

Контрольные вопросы по теме:

1. Что представляет собой дерево решений?
2. Из каких объектов состоит дерево решений?
3. В чем отличие узла от листа?
4. Для каких задач Data Mining может использоваться дерево решений?
5. Какой вид правил используется в деревьях решений?
6. Всегда ли дерево, распознавшее все обучающие примеры, является наилучшим?
7. Какие существуют способы упрощения деревьев решений?
8. Почему узлы и листья, содержащие всего несколько примеров, имеет смысл отсекать?
9. Дерево решений как линейный классификатор.
10. В чем суть использования «случайного леса»?