Деревья принятия решений

1 Достоинства и недостатки деревьев решений

Деревья принятия решений (decision tree) являются универсальными алгоритмами машинного обучения, которые могут заниматься задачами классификации и регрессии, включая даже многовыходовые задачи. Они представляют собой очень мощные алгоритмы, способные подгоняться к сложным наборам данных.

Деревья принятия решений также являются фундаментальными компонентами случайных лесов, которые входят в число самых мощных алгоритмов МО, доступных на сегодняшний день.

Деревья решений обучаются на основе данных, чтобы предсказать закон с помощью набора правил принятия решений “если - то - это”. Чем глубже дерево, тем сложнее правила принятия решений и тем точнее модель.

Некоторые преимущества деревьев решений:

- простота понимания и интерпретации. Деревья можно визуализировать;
- не требует особой подготовки данных. Другие методы часто требуют нормализации данных, создания фиктивных переменных и удаления пустых значений. Некоторые комбинации деревьев и алгоритмов поддерживают пропущенные значения;
- стоимость использования дерева (т. е. предсказания данных) логарифмически зависит от количества точек данных, используемых для обучения дерева;
- способно работать как с числовыми, так и с категориальными данными. Однако реализация scikit-learn пока не поддерживает категориальные переменные. Другие методы обычно специализируются на анализе наборов данных, содержащих только один тип переменных. Дополнительную информацию см. в алгоритмы;
- использует модель “белого ящика”. Если данная ситуация наблюдаема в модели, то объяснение состояния легко объясняется с помощью булевой логики. Напротив, в модели “черного ящика” (например, в искусственной нейронной сети) результаты может быть сложнее интерпретировать.

К недостаткам деревьев решений относятся:

- обучающие деревья решений могут создавать слишком сложные деревья, которые плохо обобщают данные. Это называется переобучением.

Чтобы избежать этой проблемы, необходимы такие механизмы, как обрезка, установка минимального количества образцов, необходимых в узле листа, или установка максимальной глубины дерева;

– деревья решений могут быть нестабильными, поскольку небольшие изменения в данных могут привести к созданию совершенно другого дерева. Эта проблема решается путем использования деревьев решений в ансамбле;

– предсказания деревьев решений не являются ни гладкими, ни непрерывными, а представляют собой кусочно-постоянные аппроксимации. Поэтому они плохо подходят для экстраполяции.

– обучающие деревья решений создают смещенные деревья, если некоторые классы доминируют. Поэтому рекомендуется сбалансировать набор данных перед обучением дерева решений.

В задачах машинного обучения чаще всего в вершинах прописываются максимально простые условия. Обычно это сравнение значения одного из признаков x_j с некоторым заданным порогом t :

$$[x_j \leq t].$$

Если решается задача классификации, конечным прогнозом является класс или распределение вероятностей классов. В случае регрессии прогноз в листе является вещественным числом.

В машинном обучении деревья строятся последовательно от корня к листьям. Вначале выбирается корень и критерий, по которому выборка разбивается на две. Затем то же самое делается для каждого из потомков этого корня и так далее до достаточного уровня ветвления. Задача состоит в выборе способа **разбиения каждого из узлов**, то есть в выборе значения порога, с которым будет сравниваться значение одного из признаков в каждом узле.

Разбиение выбирается с точки зрения некоторого заранее заданного функционала качества Q(X,j,t). Находятся наилучшие значения j и t для создания *предсказания* $[x_j<t]$. **Параметры** j **и** t **можно выбирать перебором**: признаков конечное число, а из всех возможных значений порога t можно рассматривать только те, при которых получаются различные разбиения на две подвыборки, таким образом, различных значений параметра t будет столько же, сколько различных значений признака x_j в обучающей выборке.

В каждой вершине производится проверка, не выполнилось ли некоторое условие останова (критерии останова рассмотрим далее), и если оно

выполнилось, разбиение прекращается, и вершина объявляется листом, и он будет содержать прогноз.

В задаче *классификации* это будет класс, к которому относится большая часть объектов из выборки в листе X_m

$$B=\arg c\in C\max P(c|Xm)$$

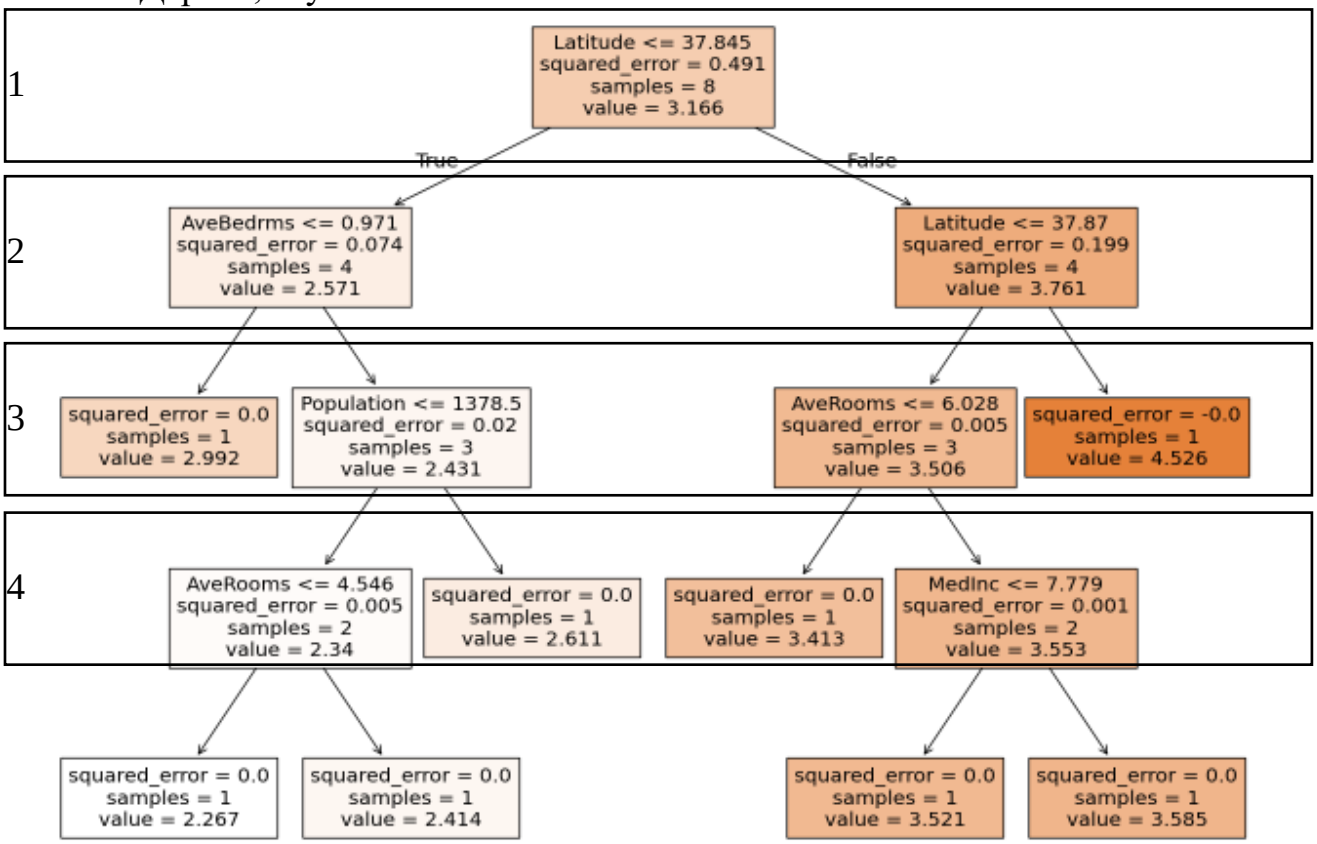
где C — множество классов,

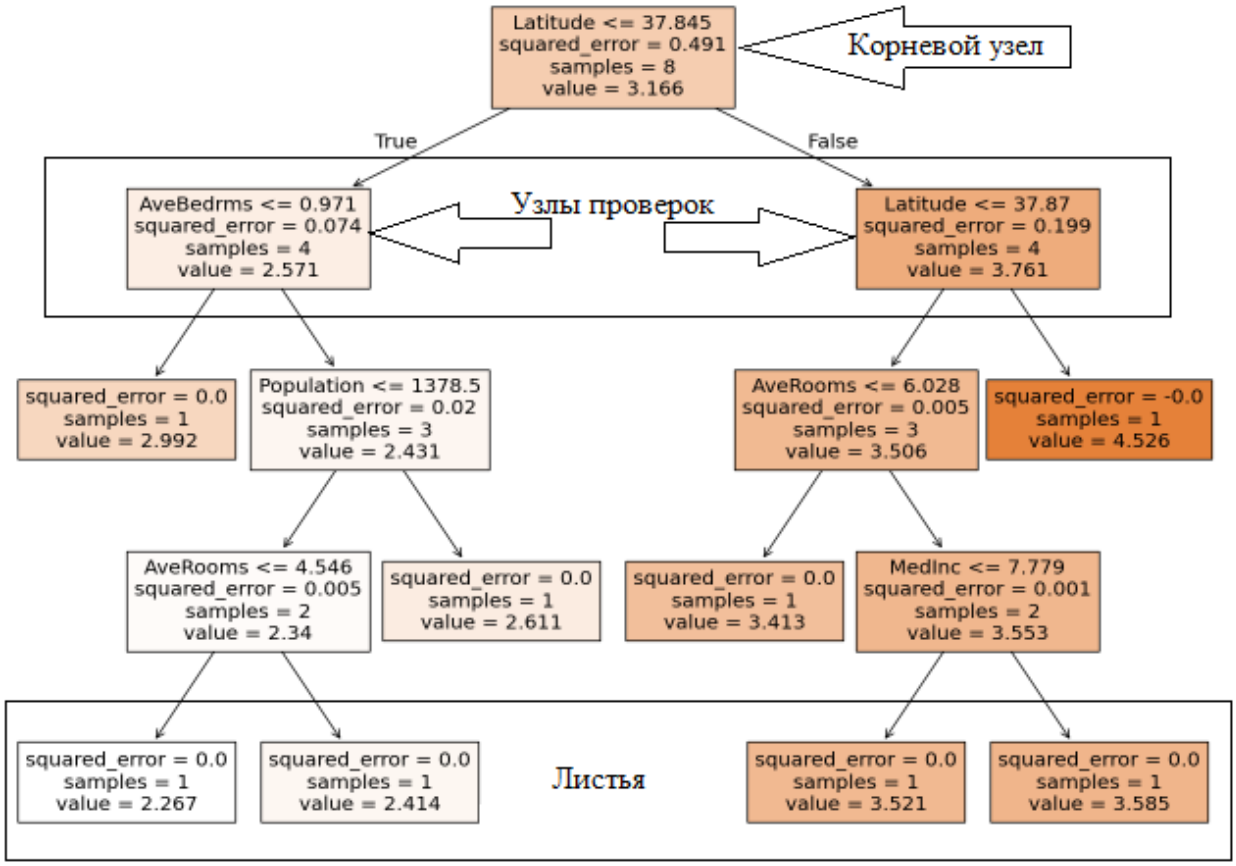
$P(c|X_m)$ — условная вероятность того, что объект принадлежит классу c, при условии, что он относится к выборке Xm.

В случае *регрессии* можно в качестве ответа возвращать средний по выборке в листе

$$a_m = \frac{1}{|X_m|}\sum_{i\in X_m} y_i.$$

Дерево, глубиной 4.





2 Критерии информативности и прирост информации

За функционал качества при работе с деревом решений принимается функционал вида

$$Q(X_m, j, t) = H(X_m) - \frac{|X_l|}{|X_m|} H(X_l) - \frac{|X_r|}{|X_m|} H(X_r)$$

также его называют **приростом информации** (information gain).

Где X_m – множество объектов, попавших в вершину на данном шаге,

$|X_l|$ и $|X_r|$ – множества, попадающие в левое и правое поддерево, после разбиения

$H(\mathrm{X})$ – *критерий информативности верхнего узла*. Он оценивает качество распределения объектов в подмножестве и тем меньше, чем меньше разнообразие ответов в X;

$H(X_l)$ – критерий информативности узла в левой ветви;

$H(X_r)$ – критерий информативности узла в правой ветви.

Задача обучения состоит в минимизации критерия информативности и, соответственно, максимизации $Q(X_m, j, t)$ на данном шаге. Последний, по сути, характеризует прирост качества на данном шаге.

В формуле значения критериев информативности нормируются – домножаются на долю объектов, ушедших в соответствующее подмножество: $\frac{|X_l|}{|X_m|}$ – доля объектов из левой ветви, $\frac{|X_r|}{|X_m|}$ – доля объектов из право ветви.

Критерий информативности

В задаче **классификации** есть несколько способов определить критерий информативности.

Обозначим через p_k долю объектов класса k в выборке X:

$$p_k = \frac{1}{|X|}\sum_{i \epsilon X}[y_i = k]$$

будет характеризовать вероятность выдачи класса k.

Энтропийный критерий или *энтропия Шеннона*:

$$H(X) = -\sum_{k=1}^{K} p_k log_2 p_k.$$

Минимум энтропии также достигается когда все объекты относятся к одному классу, а максимум - при равномерном распределении. Стоит отметить, что в формуле полагается, что $p_k log_2 p_k$=0.

Критерий Джини или *индекс Джини* выглядит следующим образом:

$$H(X) = \sum_{k=1}^{K} p_k (1 - p_k) = 1 - \sum_{k=1}^{K} p_k^2,$$

где K - количество классов в наборе данных X.

Его минимум достигается когда все объекты в подмножестве относятся к одному классу, а максимум - при равном содержании объектов всех классов. Критерий информативности Джини можно интерпретировать как вероятность ошибки случайного классификатора.

Что необходимо применять - загрязненность Джини или энтропию? Оба эти показателя приводят к похожим деревьям. Загрязненность Джини слегка быстрее под считывать, поэтому она является хорошим вариантом по умолчанию. Тем не менее, когда разница есть, загрязненность Джини имеет тенденцию изолировать самый часто встречающийся класс в собственной ветви дерева, а энтропия - производить чуть более сбалансированные деревьям.

В случае **регрессии** разброс будет характеризоваться дисперсией или же *среднеквадратичным отклонением*, поэтому критерий информативности будет записан в виде

$$H(X) = \frac{1}{X}\sum_{i \in X}(y_i - \bar{y}(X))^2,$$

или же *среднеабсолютным отклонением*:

$$H(X) = \frac{1}{X}\sum_{i \in X}(|y_i - \bar{y}(X)|),$$

где $\bar{y}(X)$ - среднее значение ответа в выборке X:

$$\bar{y}(X) = \frac{1}{|X|}\sum_{i \in X} y_i.$$

3 Пример расчета прироста информации и выбора лучшего вопроса для разиения

Возьмем датасет cardio.csv. Для задачи классификации используем данные по сердечно сосудистым заболеваниям.

В задаче предлагается предсказать наличие сердечно-сосудистых заболеваний по результатам классического врачебного осмотра. Датасет сформирован 3 групп признаков:

Объективные признаки:

- Возраст (в днях)
- Рост
- Вес
- Пол

Результаты измерения:

- Артериальное давление верхнее и нижнее
- Холестерин (три группы: норма, выше нормы, значительно выше нормы)
- Глюкоза (три группы: норма, выше нормы, значительно выше нормы)

Субъективные признаки (бинарные):

- Курение
- Употребление Алкоголя
- Физическая активность

```
full_df = pd.read_csv('cardio.csv', sep =';')
full_df.head()
```

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	0	50	2	168	62.0	110	80	1	1	0	0	1	0
1	1	55	1	156	85.0	140	90	3	1	0	0	1	1
2	2	51	1	165	64.0	130	70	3	1	0	0	0	1
3	3	48	2	169	82.0	150	100	1	1	0	0	1	1
4	4	47	1	156	56.0	100	60	1	1	0	0	0	0

Для более упрощения расчетов для построения дерева возьмем только 5 объектов и 2 признака. Исходные данные

```
[ ] full_features = ['age', 'gender', 'height', 'weight', 'ap_hi', 'ap_lo',
                     'cholesterol', 'gluc', 'smoke', 'alco', 'active']
    target = ['cardio']
```

```
[ ] features = ['age', 'ap_hi']
```

```
[ ] df = full_df[features + target]
    df = df.head(5)
    df
```

	age	ap_hi	cardio
0	50	110	0
1	55	140	1
2	51	130	1
3	48	150	1
4	47	100	0

Обучим одно дерево решений с помощью sklearn'a.

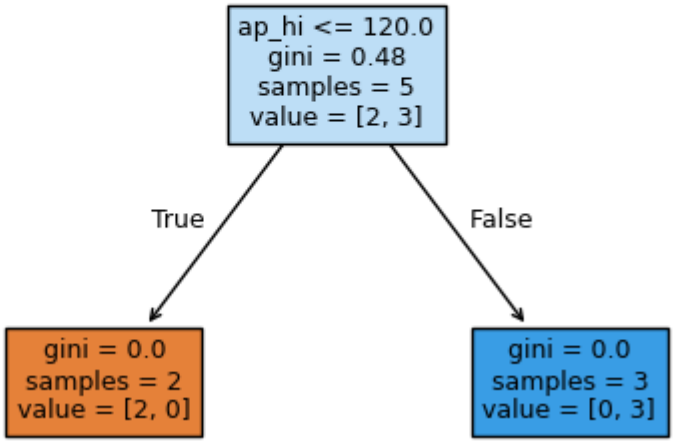
Инициализируем его для задачи классификации и обучим на признаках (X) и целевой переменной (y). По признакам модель будет запоминать закономерности, которые больше влияют на наличие сердечно-сосудистого заболевания.

```
[ ]  tree = DecisionTreeClassifier(random_state=1)
     tree.fit(X, y)
```

```
DecisionTreeClassifier(random_state=1)
```

Теперь визуализируем обученное дерево решений. Дерево решений представляет собой набор вопросов к данным. Узлы (ноды), где как раз-таки находится вопрос - называются **вершинами**, в этом дереве у нас есть одна вершина, где хранится вопрос ap_hi <= 120.

```
plt.figure(figsize=(6, 4))
plot_tree(tree, feature_names=features, filled=True, fontsize = 9);
```



По этому вопросу идет разбиение выборки на две подвыборки: на левую, где условие удовлетворяется (*верхняя цифра давления действительно меньше либо равна 120*) и на правую, где условие не удовлетворяетсся (*где давление больше 120*).

Тем самым получаем ещё два узла, которые уже именуются **листами**, благодаря ним будем получать финальное предсказание модели.

В **левом листе** у нас находится два объекта и они вдвоем 0 класса, значит любой объект, у которого давление будет меньше или равно 120 попадет в этот конечный узел и будет считаться 0 классом.

А в **правом листе** находится три объекта и они все имеют метку класс 1. А значит, если возьмем новый объект, спросим у него давление и оно окажется больше 120, то предскажем ему класс 1. Левая подвыборка, где условие удовлетворяется

```
[ ] df[df['ap_hi'] <= 120]
```

	age	ap_hi	cardio
0	50	110	0
4	47	100	0

Правая подвыборка, где условие не удовлетворяется

3.1 Какие вопросы задает дерево решений?

Как дерево решений понимает, какие вопросы нужно задавать? Дерево задает всевозможные вопросы, по всем признакам, которые присутствуют в датасете. Примеры вопросов разобраны ниже

```
df
```

	age	ap_hi	cardio
0	50	110	0
1	55	140	1
2	51	130	1
3	48	150	1
4	47	100	0

Вопросы с признаком age

Для начала с признаком возраста, age <= 50. Получаем две подвыборки, где слева 3 объекта, а справа 2.

'Left'

	age	ap_hi	cardio
0	50	110	0
3	48	150	1
4	47	100	0

'Right'

	age	ap_hi	cardio
1	55	140	1
2	51	130	1

- age <= 55

Вот этот вопрос нелогичный, так как они никак данные не разбивает, а кладет все объекты в одну подвыборку, так что его не считаем за нормальный вопрос.

Вопрос	age <= 50	age <= 55	age <= 51	age <= 48	age <= 47
Количество элементов левой ветви	3	5	4	2	1
Количество элементов правой ветви	2	-	1	3	4

Т.е. из 5 возможных вопросов, задано будет 4, т.к. самое большое значение возраста (55) не даст разбиения на 2 ветви.

Вопросы с признаком ap_hi

Дальше с признаком ap_hi.

- ap_hi <= 110

Слева 2 объекта, справа 3

'Left'

	age	ap_hi	cardio
0	50	110	0
4	47	100	0

'Right'

	age	ap_hi	cardio
1	55	140	1
2	51	130	1
3	48	150	1

Вопрос	ap_hi<=100	ap_hi<=110	ap_hi<=140	ap_hi<=130	ap_hi<=150
Количество элементов левой ветви	1	2	4	3	5
Количество элементов правой ветви	4	3	1	2	-

Итого получили 8 вопросов по 2 признакам.

3.2 Как дерево решений обучается?

Используя формулы, приведенные выше, рассчитаем прирост информации и критерии информативности.

```
[ ] df
```

	age	ap_hi	cardio
0	50	110	0
1	55	140	1
2	51	130	1
3	48	150	1
4	47	100	0

У нас всё еще есть выборка из 5 людей. У двух из них нет заболевания (класс 0), а у трех оно имеется (класс 1). И мы хотим с помощью дерева

решений найти такое разбиение, которое наиболее качественно разбивает выборку на две подвыборки, при этом в одной должны быть люди только с 0 классом, а во второй должны быть только люди с 1 классом.

Следить за таким порядком в подвыборках будет Энтропия или критерий Джини. Они позволяют посчитать количество неопределенности в данных.

Давайте разберемся в этом занятии с энтропией, а критерий Джини работает практически аналогичным образом.

Начнем с подсчета энтропии в исходной выборке из 5 объектов. У нас два объекта 0 класса и три объекта 1 класса.

Вероятность быть 0 классом составляет 0,4:

$$p_0=2/5=0,4$$

Вероятность быть 1 классом равняется 0,6

$$p_1=3/5=0,6$$

Значит в данной системе очень много неопределенности, ведь практически с вероятностью 50/50 объект может быть или 0 классом или 1. Создавая модель дерево решений, мы хотим задавать такие вопросы, которые будут отделять 1 класс от 0 и класть в отдельные подвыборки.

Посчитаем энтропию.

$$H(X)=-(p_0\log_2 p_0+p_1\log_2 p_1)=-(0,4*\log_2 0,4+0,6*\log_2 0,6)=0,97.$$

Она получается довольно высокой (напомню, что в идеале критерий информативности энтропия должна стремиться к нулю), а значит раз энтропия высокая, то и определенности пока что нет. Разбиваем выборку до этой самой определенности. Список вопросов, которые можем тестировать, уже составлен.

Вопросы с признаком age. Вопрос 1. age <= 50. Получаем две подвыборки, где слева 3 объекта, а справа 2.

<table>
<tr><td>'Left'
<table><tr><th></th><th>age</th><th>ap_hi</th><th>cardio</th></tr><tr><td>0</td><td>50</td><td>110</td><td>0</td></tr><tr><td>3</td><td>48</td><td>150</td><td>1</td></tr><tr><td>4</td><td>47</td><td>100</td><td>0</td></tr></table></td><td>вероятность быть 0 классом в левой ветке
$p_0=2/3=0,667$

вероятность быть 1 классом равняется
$p_1=1/3=0,334$</td></tr>
<tr><td colspan="2">$H(X)=-(p_0\log_2 p_0+p_1\log_2 p_1)=-(0,667*\log_2 0,667+0,334*\log_2 0,334)=0,918.$</td></tr>
<tr><td>'Right'
<table><tr><th></th><th>age</th><th>ap_hi</th><th>cardio</th></tr><tr><td>1</td><td>55</td><td>140</td><td>1</td></tr><tr><td>2</td><td>51</td><td>130</td><td>1</td></tr></table></td><td>вероятность быть 0 классом в правой ветке
$p_0=0/2=0$

вероятность быть 1 классом равняется
$p_1=2/2=1$</td></tr>
<tr><td colspan="2">Объединяем вероятности в энтропию
$H(X)=-(p_0\log_2 p_0+p_1\log_2 p_1)=-(0+1*\log_2 1)=0$</td></tr>
</table>

Помня, что в левой ветви оказалось 3 объекта из 5 исходных, а в правой их оказалось 2 тоже из 5, **прирост информации** для вопроса **age <= 50 составит**

$$Q(X_m, j, t) = H(X_m) - \frac{|X_l|}{|X_m|}H(X_l) - \frac{|X_r|}{|X_m|}H(X_r)$$

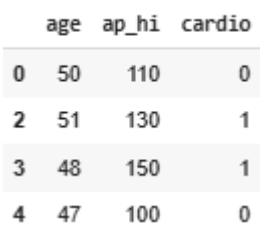
$$Q(X_m,j,t)=0,97-3/5*0,918-2/5*0=0,419$$

Все найденные значения прироста информации будем записывать в таблицу.

Признак	Значение	Прирост информации
age	50	0,419

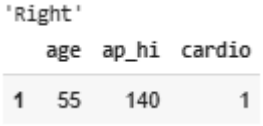
Вопрос 2. age <= 55. Этот вопрос был некорректным, так как он не дал разбиения.

Вопрос 3. age <= 52. Получаем две подвыборки, слева 4 объекта, а справа 1.

<table>
<tr><td>'Left'
 

	age	ap_hi	cardio
0	50	110	0
2	51	130	1
3	48	150	1
4	47	100	0

</td><td>вероятность быть 0 классом в левой ветке
$p_0=2/4=0.5$

вероятность быть 1 классом равняется
$p_1=2/4=0.5$</td></tr>
<tr><td colspan="2">$H(X)=-(p_0\log_2 p_0+p_1\log_2 p_1)=-(0.5*\log_2 0.5+0.5*\log_2 0.5)=1.$</td></tr>
<tr><td>'Right'
 

	age	ap_hi	cardio
1	55	140	1

</td><td>вероятность быть 0 классом в правой ветке
$p_0=0/2=0$

вероятность быть 1 классом равняется
$p_1=2/2=1$</td></tr>
<tr><td colspan="2">Объединяем вероятности в энтропию
$H(X)=-(p_0\log_2 p_0+p_1\log_2 p_1)=-(0+1*\log_2 1)=0.$</td></tr>
</table>

Прирост информации составит

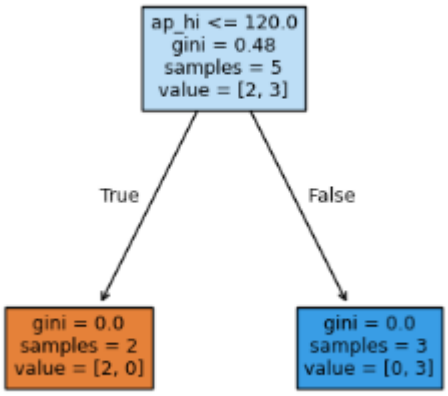
$$Q(X_m,j,t)=0,97-4/5*1-1/5*0=0,17$$

Аналогичным образом задаем вопросы к остальным признакам, перебирая возможные значения каждого признака. Итоговая таблица будет выглядеть следующим образом:

Признак	Значение	Прирост информации
age	50	0.419973
age	52	0.170951
age	48	0.019973
ap_hi	110	0.970951
ap_hi	100	0.321928
ap_hi	140	0.170951
ap_hi	130	0.419973

У вопроса ap_hi <= 110 – у него максимальный прирост информации из всевозможных вопросов, которые мы задавали, соответственно его будем использовать для дерева решений.

Следует обратить внимание, что когда строилось дерево решений из sklearn'a, то самый лучший вопрос получался ap_hi <= 120, а у нас такого вопроса не было.



Дело в том, что вопросы можно задавать не сколько иначе: брать не буквально уникальные значения из признака, а среднее между уникальными отсортированными значениями. В этом наборе данных уникальные значения давления следующие: [100, 110, 130, 140, 150]. Соответственно, вопросы будут следующими:

ap_hi <= 105.0
ap_hi <= 120.0
ap_hi <= 135.0
ap_hi <= 145.0

При этом разбиения мы будем получать такие же, как и при рассмотренных ранее вопросах, к примеру, вопрос по такое методике ap_hi <= 105 (среднее между 100 и 110). Это тоже самое, что и вопрос ap_hi <= 100.

'Left'

	age	ap_hi	cardio
4	47	100	0

'Right'

	age	ap_hi	cardio
0	50	110	0
1	55	140	1
2	51	130	1
3	48	150	1

Лучшие вопросы можно находить:

– через уникальные значения в признаке
– через среднее арифметическое уникальных значений в признаке

4 Алгорим построения дерева решений

Возьмем выборку чуть побольше (10 значений), чтобы дерево решений имело больше вершин (узлы с вопросами) и листов (узлы с предсказаниями).

```
X = full_df.iloc[:10][features]
y = full_df.iloc[:10][target]
```

Первым делом получился вопрос ap_hi <= 125, в этой вершине находится 10 объектов (samples=10), из которых 6 относится к классу 0 и 4 относятся к классу 1 (value=[6, 4]). Тут энтропия высокая, равная 0.971.

Затем задается вопрос, где получился самый большой прирост информации и делается разбиение на две подвыборки.

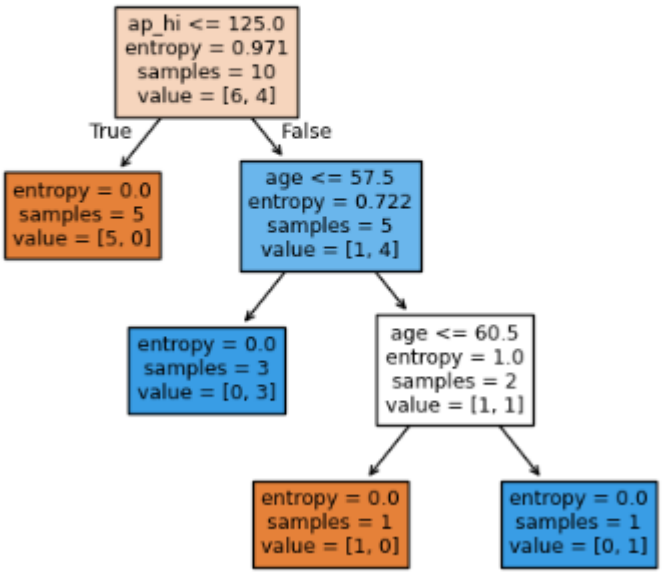
Левая получилась из 5 объектов (samples=5), они все являются объектами 0 класса, так что у них энтропия равняется 0, определено всё, если объект оказывается в этом листе, то 100% он будет нулевым классом, сделать энтропию меньше нуля невозможно, поэтому перейдем к правой.

В ней так же 5 объектов, но один – это 0 класс, а четверо – это объекты 1 класса, так что энтропия не нулевая, поэтому здесь продолжаем задавать вопросы, с помощью перебора, постоянно считая энтропию и объединяя их в прирост информации, где он получился больше, этот вопрос и берем.

Оказалось, что самый большой прирост информации с вопросом age <= 58.

И так далее, пока не встретим какой-либо критерий останова.

```
tree = DecisionTreeClassifier(criterion='entropy')
tree.fit(X, y)

plt.figure(figsize=(6, 5))
plot_tree(tree, feature_names=features, filled=True, fontsize = 9);
```



5 Критерии останова

Будем работать с набором данных для задачи регрессии (целевая переменная - стоимость дома) california_housing, который можно получить из стандартных датасетов в sklearn'e. Для простоты понимания ограничимся десятью объектами из выборки.

```
from sklearn.datasets import fetch_california_housing

data = fetch_california_housing()

X = data.data
features = data.feature_names
y = data.target

df_full = pd.DataFrame(X, columns=features)
df_full['target'] = y

df = df_full.iloc[:10]
```

Разобьем выборку на две: обучающую и тестовую.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    df[features],
    df['target'],
    test_size=0.2,
    shuffle=True,
    random_state=3
)

X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((8, 8), (8,), (2, 8), (2,))
```

5.1 Обучение дерева решений

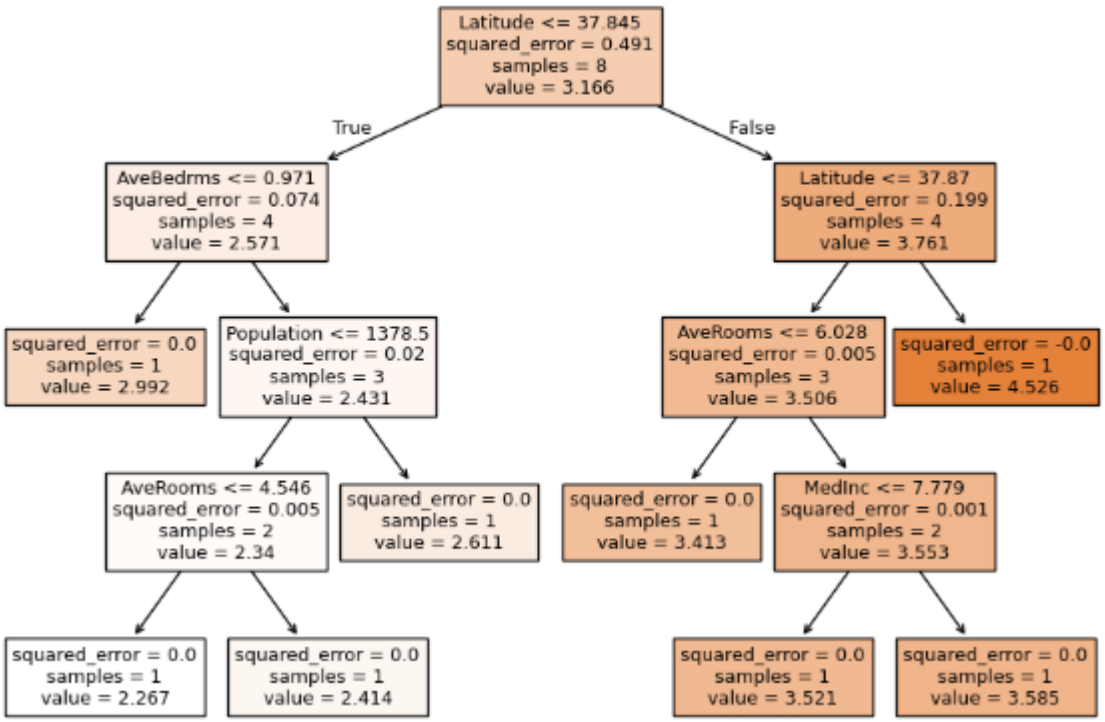
Инициализируем дерево решений для задачи регрессии и обучим на обучающей выборке (X_train) и целевой переменной для обучающих объектов (y_train).

```
from sklearn.tree import DecisionTreeRegressor


tree = DecisionTreeRegressor(random_state=1)
tree.fit(X_train, y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor(random_state=1)
```

После визуализации дерево решений получилось не очень большим. Получилось 7 узлов с вопросами и 8 листов с предсказаниями, кстати именно столько объектов и было в обучающей выборке.

```
from sklearn.tree import plot_tree


plt.figure(figsize=(10, 7))
plot_tree(tree, feature_names=features, filled=True);
```



Узнаем, насколько дерево решений обучилось хорошо, для этого сделаем предсказания моделью для обучающей выборке и для тестовой, а затем посчитаем метрику качества – средне-квадратичную ошибку.

```
MSE на обучении 0.00
MSE на тесте 0.26
```

Метрика на обучении получилась очень маленькая, равная нулю, это говорит о том, что во все истинные значения модель идеально попала.

Все значения целевого признака из обучения полностью совпадают с предсказанными значениями:

	Реальное значение	**Предсказанное значение**
1	3,585	3,585
2	3,521	3,521
9	2,611	2,611
6	2,992	2,992
7	2,414	2,414

0	4,526	4,526
3	3,413	3,413
8	2,267	2,267

На тестовой выборке не все идеально:

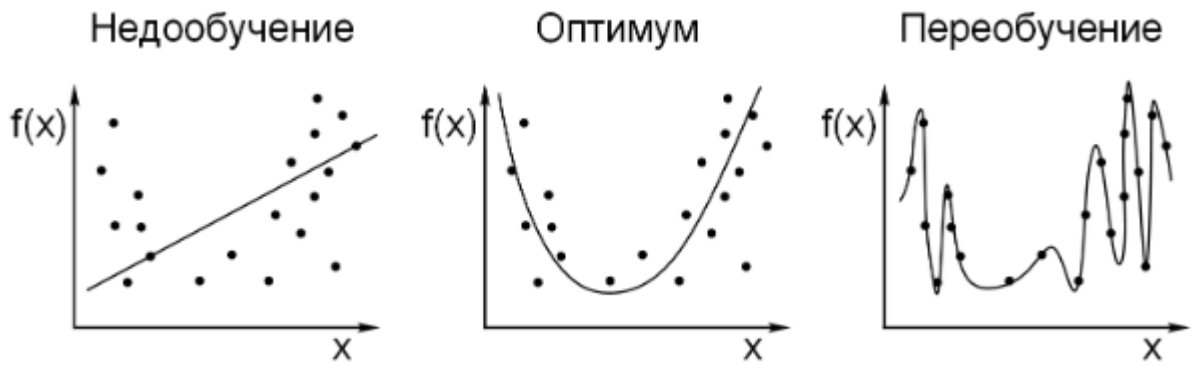
	Реальное значение	**Предсказанное значение**
5	2,697	3,413
4	3,422	3,521

На тесте отклонения истинных значений от предсказанных больше – это и отображается в метрике MSE на тестовых данных.

5.2 Зачем нужны критерии останова?

Когда есть разница между метриками качества на обучении и тесте, это говорит о переобучении – явлении, когда построенная модель хорошо объясняет примеры из обучающей выборки, но относительно плохо работает на примерах, не участвовавших в обучении (на примерах из тестовой выборки).

Переобучение визуально показано на картинке ниже, в этом случае модель слишком хорошо запомнила обучающую выборку, искуственно подогналась под примеры и не стала изучать общую закономерность в данных.



Деревья решений страдают от переобучения чаще, модель за счет большого количества вопросов может повторять довольно сложные закономерности, иногда настолько хорошо, что запоминает всё, что видела в обучающем наборе данных.

5.3 Какие есть критерии останова у дерева решений?

Чтобы посмотреть, какие критерии останова есть у модели, можно посмотреть на её инициализацию и вывести список аргументов.

Здесь есть как аргументы, относящиеся к особенностям построения дерева, к примеру criterion – функция потерь, с помощью которой ищется самое лучшее разбиение или же random_state, который отвечает за фиксацию псевдорандома.

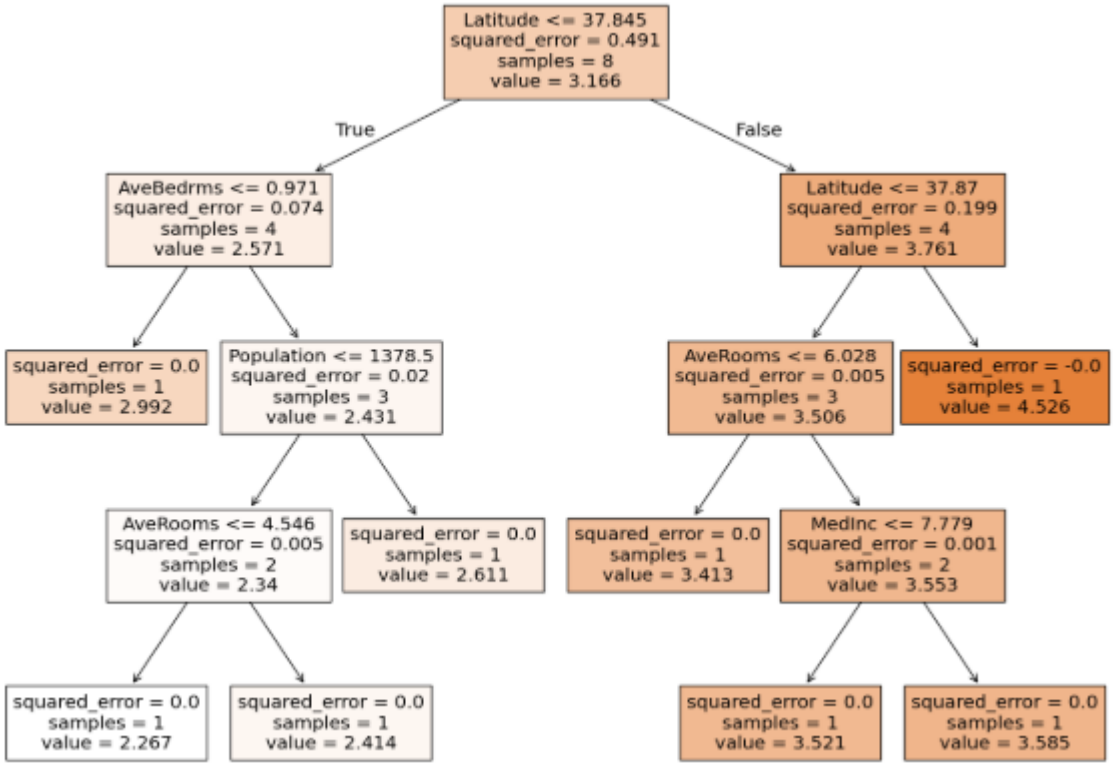
Есть и аргументы, которые помогают бороться с переобучением. Из основных:

- max_depth
- min_samples_leaf
- max_leaf_nodes

5.3.1 max_depth

Деревья решений очень любят задавать вопросы к данным, и они настолько много задают вопросов, что становятся очень сложными и ветвистыми, критерий останова по максимальной глубине дерева призван помочь с этим, за счет ограничения уровней с вопросами.

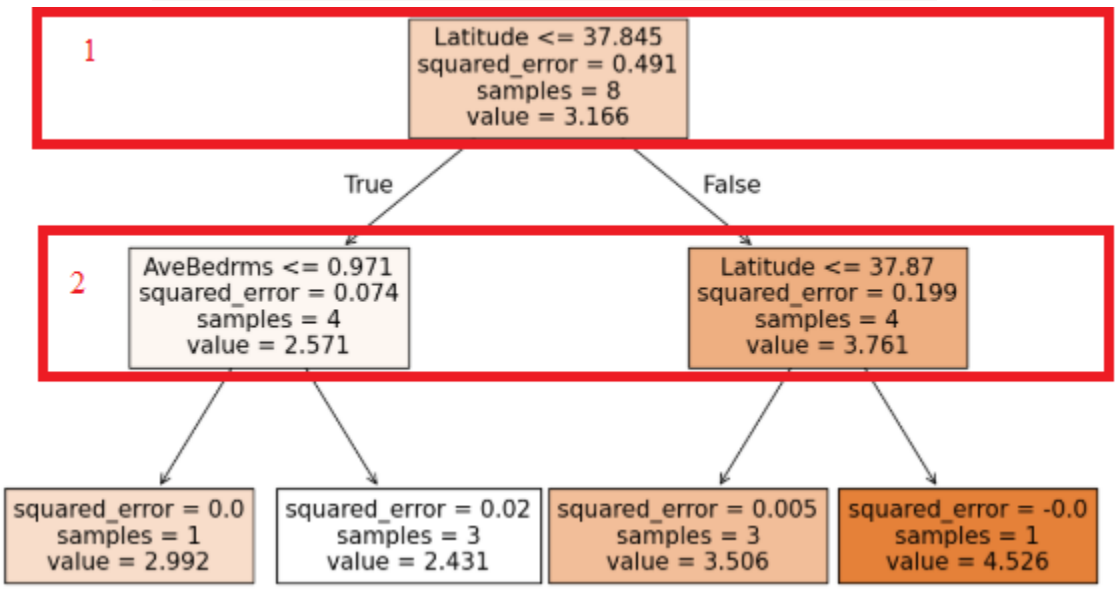
Когда обучили дерево решений первый раз, то не писали аргумент max_depth, в этом случае глубина может быть сколь угодно большой. В этом случае глубина вышла 4. При этом max_depth отвечает именно за уровни с вопросами, а не их количество.



Попытаемся снизить переобучение с помощью max_depth, поставим ему значение поменьше, тем самым запретим дереву быть слишком сложным.

Если поставить, к примеру, значение равное 2,то получим только два уровня с вопросами, вопросов – 3, а листьев с предсказаниями – 4.

```
tree = DecisionTreeRegressor(random_state=1, max_depth=2)
tree.fit(X_train, y_train)

plt.figure(figsize=(13, 8))
plot_tree(tree, feature_names=features, filled=True);
```



5.3.2 min_samples_leaf

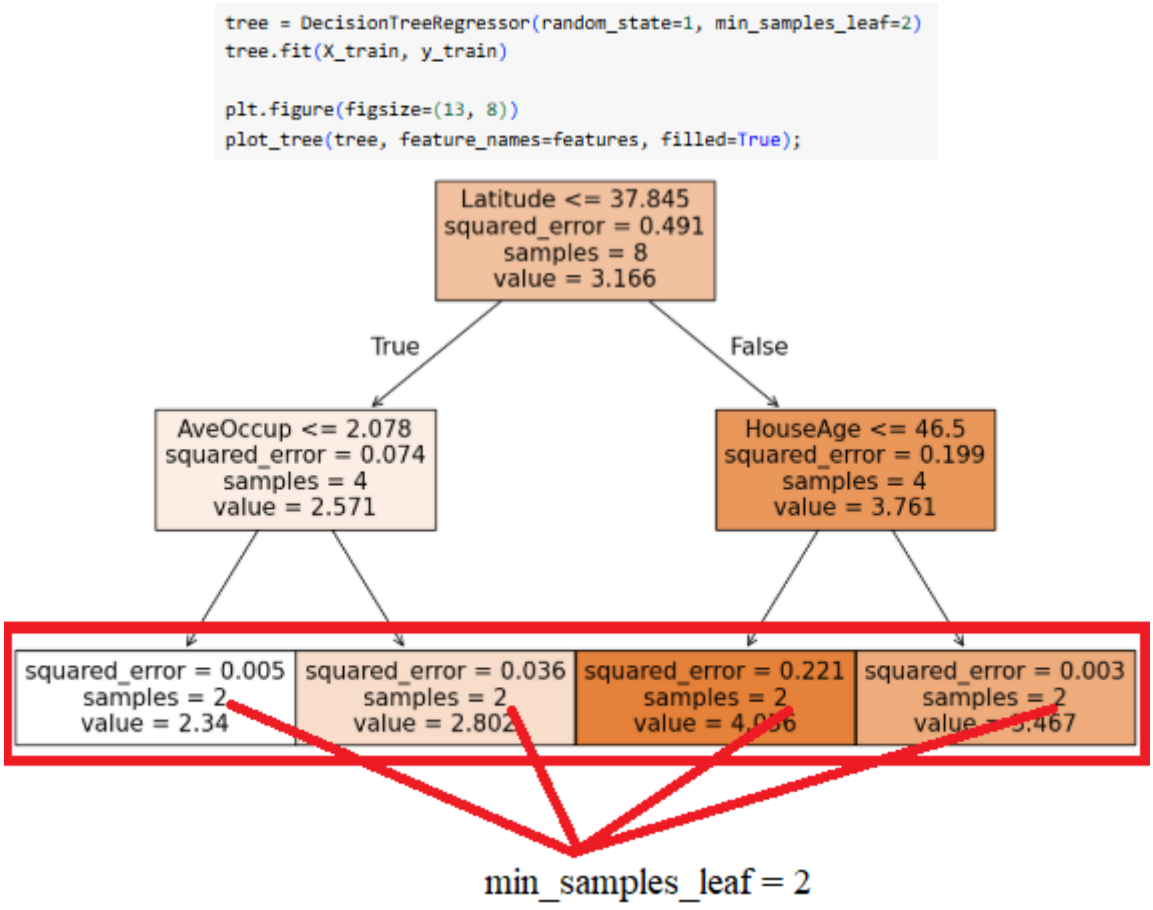
min_samples_leaf – минимальное количество объектов в одном листе.

Критерий min_samples_leaf способствует улучшению обобщающей способности моделей, основанных на деревьях решений. Деревья решений склонны детализировать данные, стремясь разделить каждую выборку в отдельный лист, что может привести к переобучению. Установка минимального количества объектов в листе служит механизмом остановки, предотвращая излишнюю детализацию модели и тем самым снижая вероятность запоминания шумов в данных. Это позволяет достичь более устойчивых и надежных предсказаний, улучшая обобщающую способность модели на новых, невидимых данных.

По умолчанию min_samples_leaf равен 1, а значит мы позволяем дереву строить листья с одним объектом в листе, из-за этого получаем идеальную подгонку обучающей выборки и далеко неидеальную метрику на тесте.

Изменим аргумент на значение 2, чтобы дерево пыталось строить более обобщенную модель.

Видим, что теперь samples в каждом конечном узле равно два, значит там находится по два объекта.



5.3 max_leaf_nodes

max_leaf_nodes – максимальное количество листьев.

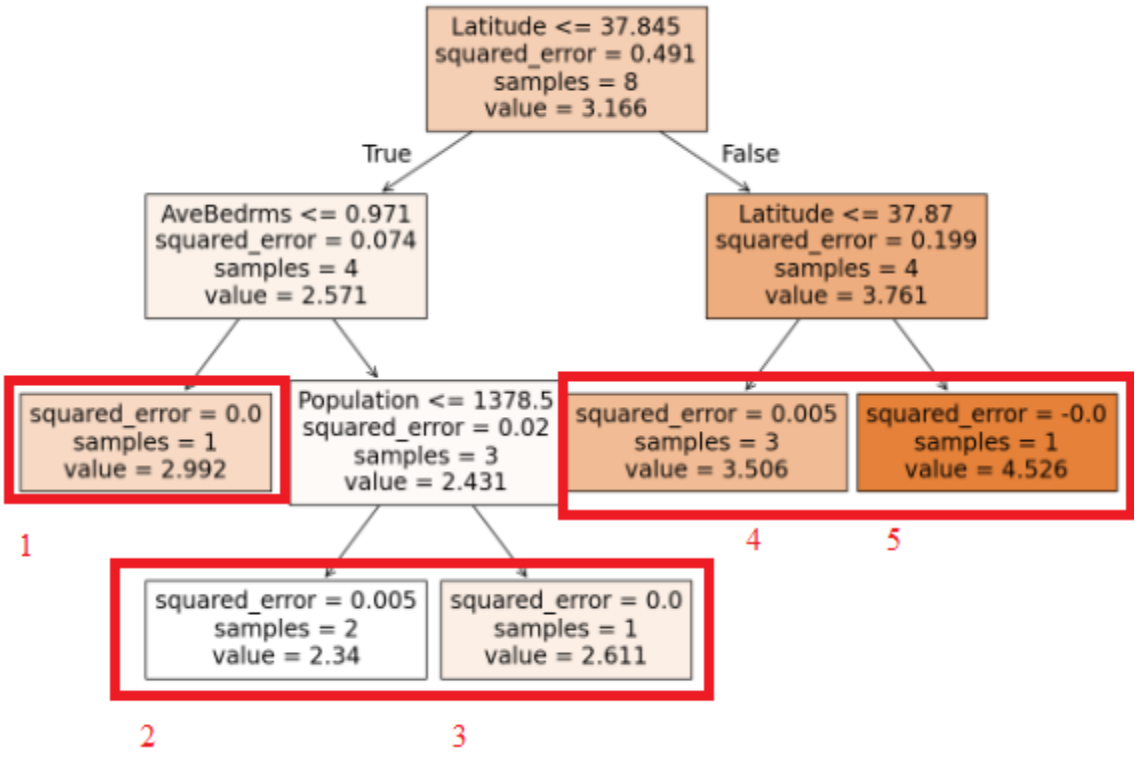
Чем больше листьев, тем больше переобучение, потому что модель более сложная.

По умолчанию max_leaf_nodes равен None, а значит ничем не ограничен и дерево будет строиться пока будет возможность подгоняться под данные.

В самом первом, самом переобученном дереве было 8 листьев, ровно столько же, сколько объектов в обучении, сделаем количество листьев поменьше, чтобы и переобучения было поменьше.

И видим, что теперь количество листьев действительно равно 5.

```
tree = DecisionTreeRegressor(random_state=1, max_leaf_nodes=5)
tree.fit(X_train, y_train)

plt.figure(figsize=(15, 10))
plot_tree(tree, feature_names=features, filled=True);
```



5.4 Как пользоваться критериями останова?

Данные критерии конечно можно использовать вместе для более высокого качества работы модели, да и они все связаны между собой: чем меньше уровней с вопросами (max_depth), тем меньше вопросов и меньше

листьев (max_leaf_nodes), тем больше количество объектов в одном листе (min_samples_leaf).

Возьмем весь набор данных и попытаемся подобрать самые лучшие параметры дерева решений.

```
X_train, X_test, y_train, y_test = train_test_split(
    df_full[features],
    df_full['target'],
    test_size=0.2,
    shuffle=True,
    random_state=3
)

X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((16512, 8), (16512,), (4128, 8), (4128,))
```

Пока обучимся на параметрах по умолчанию:

- max_depth=None
- min_samples_leaf=1
- max_leaf_nodes=None

```
tree = DecisionTreeRegressor(random_state=1)
tree.fit(X_train, y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor(random_state=1)
```

И видим переобучение, метрика на обучении идеальная, мы каждый объект предсказали правильно, а вот на тесте – нет. Будем исправлять.

```
MSE на обучении 0.00
MSE на тесте 0.52
```

Пойдем по порядку и будем изменять максимальную глубину, возьмем случайное число и поставим.

Разница в метриках стала меньше, а значит переобучение тоже уменьшилось, плюс метрика на тесте стала лучше.

```
tree = DecisionTreeRegressor(random_state=1,
                             max_depth=15,
                             min_samples_leaf=1,
                             max_leaf_nodes=None)
tree.fit(X_train, y_train)

pred_train = tree.predict(X_train)
pred_test = tree.predict(X_test)

mse_train = mean_squared_error(y_train, pred_train)
mse_test = mean_squared_error(y_test, pred_test)

print(f'MSE на обучении {mse_train:.2f}')
print(f'MSE на тесте {mse_test:.2f}')
```

```
MSE на обучении 0.04
MSE на тесте 0.48
```

Давайте изменим с максимальным количеством листьев, поставим число 500.

```
tree = DecisionTreeRegressor(random_state=1,
                             max_depth=15,
                             min_samples_leaf=1,
                             max_leaf_nodes=500)
tree.fit(X_train, y_train)

pred_train = tree.predict(X_train)
pred_test = tree.predict(X_test)

mse_train = mean_squared_error(y_train, pred_train)
mse_test = mean_squared_error(y_test, pred_test)

print(f'MSE на обучении {mse_train:.2f}')
print(f'MSE на тесте {mse_test:.2f}')
```

```
MSE на обучении 0.19
MSE на тесте 0.40
```

Метрики стали лучше, тенденция отличная, переобучения меньше. Действуем дальше и можем изменить минимальное количество объектов в одном листе, сделаем его больше.

Сведем результаты экспериментов в таблицу

Шаг	Критерий	Значение	MSE на обучении	MSE на тесте
0	-	-	0,00	0,52
1	max_depth	15	0,04	0,48
2	max_leaf_nodes	500	0,19	0,40
3	min_samples_leaf	10	0,22	0,35
4	max_depth	13	0,23	0,36
5	max_depth	14	0,23	0,35

6	min_samples_leaf max_leaf_nodes	24 400	0,28	0,34

Рекомендут изменять только один параметр за раз и следить, за метриками:

– если стало хуже, нужно вернуться на предыдущий шаг

– если стало лучше, вы сделали всё верно.

Нужно нащупывать границы в параметрах, когда модель начинает вести себя лучше и не переобучаться.

6 Итог

Посмотрели на основные критерии останова в модели дерево решений. Они помогают снизить сложность модели, а значит снизить переобучение, следовательно – увеличить качество модели.

Критерий останова	Смысл	Что делать, что бы уменьшить переобучение
max_depth	**максимальное кол-во уровней с вопросами** чем глубже дерево, тем оно сложнее	уменьшать
min_samples_leaf	**минимальное кол-во объектов в одном листе** если в листе один объект - это скорее всего сложная модель с низким уровнем обощения	увеличивать
max_leaf_nodes	**максимальное вол-во листьев** чем больше листьев, тем больше вероятность сделать индивидуальный лист для объекта	уменьшать