Семинар по решающим деревьям

Теоретическая часть

Предсказания в решающих деревьях

Вспомнить из лекции:

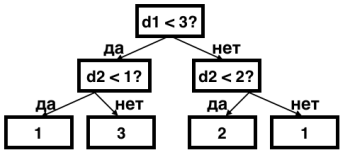
- Что такое рашющее дерево?
- Что такое предикат? Каки бывают предикаты? Какие предикаты чаще всего используются?
- В чем отличие внутренних и листовых вершин решающего дерева?
- Как выполнить предсказание с помощью решающего дерева в задачах многоклассовой классификации и регрессии?

На что обратить внимание:

- В линейной классификации мы рассматривали бинарную и многоклассовую классификацию отдельно, потому что предсказание и обучение выполняется по-разному для этих двух случаев. В решающих деревьях бинарная и многоклассовая классификация выполняются одинаково.
- Предсказания можно сделать для любой точки признакового пространства, то есть для любого возможного объекта, а не только для объекта обучающей выборки.
- Признаки в предикатах различных вершин могут повторяться.

Задача 1

Рассмотрим задачу классификации на три класса по двум признакам и следующее решающее дерево:



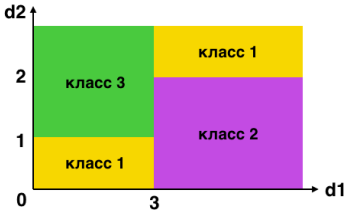
Какое предсказание это решающее дерево вернет для объекта $x = (7, 1.5)$? Под d1 и d2 подразумеваются первый и второй признак.

Решение: Необходимо в каждой вершине проверить значение предиката (Да или Нет) и решить, в левое или правое поддерево пойдет объект. Ответ: 2.

Задача 2.

Нарисуйте разделяющую поверхность для решающего дерева из предыдущей задачи.

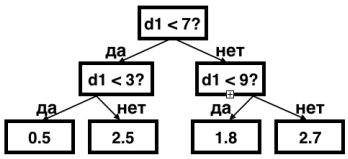
Решение: В данной задаче мы работаем с двумерным признаковым пространством, поэтому можем изобразить разделяющую поверхность на плоскости в координатах $d_1 - d_2$. Одна вершина решающего дерева выполняет разделение пространства по прямым, параллельным осям координат, например, корневой вершине соответствует прямая $d_1 = 3$. Слева от нее левое поддерево, справа - правое поддерево. Далее, мы повторяем ту же самую процедуру в обеих полуплоскостях. Например, для левого поддерева (левая полуплоскость) мы проводим прямую $d_2 = 1$. Повторяем процедуру рекурсивно для всех вершин решающего дерева. В какой-то момент мы приходим в листовые вершины, когда разделять подобласть пространства больше не нужно, а нужно присвоить подобласти метку класса.



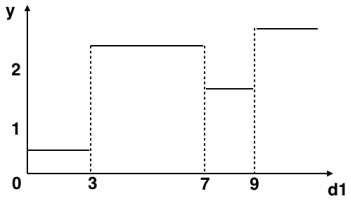
На что обратить внимание: в этой задаче у нас нет обучающей выборки, потому что мы предполагаем, что решающее дерево уже задано, и наша задача - как бы сделать предсказания для всех возможных объектов пространства (отсюда и возникает разделяющая поверхность). Об обучении решающего дерева по конкретной выборке мы поговорим в следующем разделе.

Задача 3

Рассмотрим задачу регрессии по одному признаку. Визуализируйте решающее правило для следующего решающего дерева:

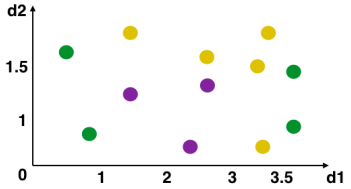


Решение: Визуализации в задаче регрессии можно выполнять так же, как для классификации: делим признаковое пространство на области, и в каждой области возвращаем предсказание. В отличие от классификации, предсказаниями здесь будут не метки класса, а вещественные числа. Чтобы отобразить предсказания нагляднее, можно добавить еще одну ось y. Однако, в случае с двумя признаками, наш график получится трехмерным; рисовать такие графики неудобно. Поэтому в данной задаче предлагается работать с одним признаком и визуализировать решающее правило в осях $d_1 - y$. Признаковое пространство в этом случае будет представленно прямой, и решающее дерево поделит ее на отрезки (и полупрямые по краям). Для каждого отрезка и полупрямой - свое вещественное предсказание. В итоге мы получим кусочно-постоянное рашющее правило.



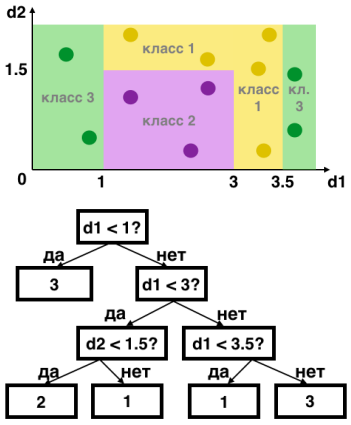
Задача 4

Приведите пример решающего дерева, которое даст нулевую ошибку в задаче классификации по двум признакам, изображенной ниже. Изобразите само решающее дерево, а также изобразите получающуюся разделяющую поверхность на рисунке. Используйте предикаты вида [j-й признак < t].



Решение: В этой задаче нужно разбить вертикальными и горизонтальными линиями признаковое пространство на области так, чтобы в каждой области оказались объекты одного класса. В частности, можно сделать отдельную область для каждого объекта.

Одно из возможных решений:



На что обратить внимание: указанным способом можно для любой непротиворечивой выборки построить решающее дерево, имеющее нулевую ошибку на этой выборке. Этот факт обосновывает склонность решающих деревьев к переобучению. Под непротиворечивостью здесь подразумевается, что в выборке нет двух объектов с одинаковыми признаковыми описаниями, но разными значениями целевой переменной.

Обучение решающего дерева

Непрерывная и дискретная оптимизация. Для обучения моделей, как правило, используются методы оптимизации. В зависимости от того, по каким переменным нужно выполнять оптимизацию, выделяют два вида оптимизации: непрерывную и дискретную. Непрерывная оптимизация выполняется по вещественным числам, часто здесь используют градиентные методы. С помощью непрерывной оптимизации мы обучали линейные модели. Дискретная оптимизация выполняется по конечным множествам, для этого необходимо делать перебор по элементам множества. Делать полный перебор может быть очень долго, поэтому придумывают более эффективные алгоритмы, например один из самых простых - жадный алгоритм. С помощью жадной дискретной оптимизации мы будем обучать решающие деревья. В решающих деревьях нужно выбрать одно из всех возможных разбиений признакового пространства на области. Несмотря на то, что всего разбиений бесконечное число, разбиений, приводящих к разным предсказаниям на обучающей выборке, конечное число - между ними и нужно выбрать одно.

Вспомнить из лекции:

- Рекурсивный алгоритм построения решающего дерева.
- Какие вы знаете критерии останова при построении решающего дерева (когда выборку не нужно делить, а нужно сделать лист)?
- Как выбрать, какое предсказание делать в листе для задачи регрессии и задачи классификации?

Критерии информативности: интуиция. Когда мы строим одну вершину решающего дерева, мы разбиваем выборку на две подвыборки: одна подвыборка будет использоваться при построении левого поддерева, вторая - при построении правого поддерева. Какую цель мы преследуем, когда разбиваем выборку на две подвыборки? Каких свойств мы хотим от получающихся подвыборок? Можно придумать разные ответы, но на практике хорошо работает следующее желаемое свойство: мы хотим, чтобы ответы в обеих подвыборках были как можно менее вариативны. В идеальной ситуации в каждой подвыборке у всех объектов одинаковые ответы, и дальше строить дерево не нужно (каждая подвыборка станет листом). На практике так обычно не получается, иначе это бы уже не было задачей машинного обучения. Однако мы можем постепенно уменьшать вариативность ответов в подвыборках. Итак, зафиксируем следующую постановку задачи: у нас есть вектор правильных ответов Y_v (ответы на всех объектах, попавших в вершину v), и мы хотим измерить вариативность этих ответов.

Как измерить вариативность в задаче регрессии? Для этого можно использовать среднеквадратичное отклонение:

$$H(Y_v) = \sum_{i=1}^{|Y_v|} (Y_{v,i} - \bar{Y}_v)^2,$$

где $\bar{Y}_v$ - среднее вектора Y_v. Можно выбирать аналогичные меры разброса в зависимости от функции потерь, используемой в задаче. Например, при использовании MAE можно заменить квадрат на модуль, а среднее - на медиану.

Как измерить вариативность в задаче классификации? Здесь обычно подходят следующим образом: на основе вектора Y_v, состоящего из меток классов $1 \dots K$, вычисляют доли каждого класса: $(p_1, \dots, p_K)$. Если все элементы вектора одинаковые (вариативность наименьшая), то среди p_k будет одна 1, оостальные значения 0 (назовем это случай а). При наименьшей вариативности вектора Y_v все $p_k \approx \frac{1}{k}$ (случай б). Осталось придумать критерий, зависящий от $(p_1, \dots, p_K)$, который минимален в случае а и максимален в случае б. Таким критерием является, например, энтропийный критерий:

$$H(Y_v) = H(p_1, \dots, p_K) = -\sum_{k=1}^{K} p_k \log p_k$$

или критерий Джини:

$$H(Y_v) = H(p_1, \dots, p_K) = \sum_{k=1}^{K} p_k(1 - p_k).$$

В итоге, мы можем измерить вариативность подвыборок, из которых будет строиться левое и правое поддерево. Чтобы построить вершину v, мы будем перебирать все возможные признаки j и все возможные пороги t (всего ℓd вариантов, ℓ - число объектов, d - число признаков), для каждой пары (j, t) мы получим разбиение выборки на две части, им соответствуют векторы правильных ответов Y_ℓ и Y_r. Далее нам нужно сравнить все разбиения и выбрать лучшее. Сравнивать разбиения удобно, когда есть один критерий; у нас же пока два критерия. Скомбинируем вариативность обеих выборок в одном критерие: $$Q(Y***v, j, t) = \frac {|Y***\ell||}{|Y***v|} H(Y***\ell) + \frac {|Y***r|}{|Y***v|} H(Y***r) \rightarrow \min***{j, \,t}$$

Веса $\frac{|Y_\ell|}{|Y_v|}$ и $\frac{|Y_r|}{|Y_v|}$ вводятся для того, чтобы не поощрять отделение одного объекта в отдельную вершину: для таких подвыборок вариативность вектора правильных ответов низкая, и без перевзвешивания алгоритм всегда выбирал бы подобные разбиения. А это нежелательно, потому что вершины из одного объекта позволяют запоминать ответы, что ведет к переобучению.

Итоговый алгоритм построения вершины: перебрать все варианты (j, t), для каждого посчитать критерий $Q(Y_v, j, t)$, выбрать пару с наибольшим значением критерия.

Задача 5

Предположим, мы решаем задачу классификации на три класса по четырем признакам и строим решающее дерево. При построении вершины v мы имеем выборку из семи объектов:

1-й признак	2-й признак	3-й признак	4-й признак	класс
5	8	8	2.5	2
3	3	7	7.7	1
6	7.7	1	1.1	3
3.3	1	2	1.2	1
24	3.9	5	3.9	1
12	10	10.1	8	2
1	2	2	9.1	1

Какой из предикатов лучше: [1-й признак < 6] или [3-й признак < 5] по критерию Джини?

Решение. Для начала посчитаем $Q(Y_v, j, t)$ для $j = 1$ и $t = 6$. Для этого разобьем вектор $Y_v = (2, 1, 3, 1, 1, 2, 1)$ на два вектора согласно предикату [1-й признак < 6]. В первую подвыборку попадают первый, второй, четвертый и седьмой объекты (так как у них первый признак меньше шести), значит, $Y_\ell = (2, 1, 1, 1)$ (класс для соответствующих объектов). Для этой подвыборки считаем доли классов: $p_1 = 0.75$ (три объекта из четырех), $p_2 = 0.25$ (один объект из четырех), $p_3 = 0$ (в Y_ℓ нет класса 3). Для полученного вектора считаем критерий Джини: $H(Y_\ell) = 0.75 \cdot (1 - 0.75) + 0.25 \cdot (1 - 0.25) + 0 = 0.375$.

Аналогично для второй подвыборки: $Y_r = (3, 1, 2)$ (целевые переменные для всех остальных объектов, не удовлетворяющих условию [1-й признак < 6]). Считаем критерий Джини: $H(Y_r) = 3 \cdot \frac{1}{3} \cdot (1 - \frac{1}{3}) = \frac{2}{3} \approx 0.67$.

Считаем итоговое значение критерия по формуле:

$$Q(Y_v, 1, 6) = \frac{4}{7} \cdot 0.375 + \frac{3}{7} \cdot 0.67 \approx 0.5014$$

Все то же самое нужно повторить для $j = 3$ и $t = 5$, получится $Q(Y_v, 3, 5) = 0.4761$.

В итоге значение критерия меньше у второго разбиения, поэтому ответ: лучше предикат [3-й признак < 5].

Вопрос: Что является параметрами и гиперпараметрами решающих деревьев?

Решение:

Когда мы говорим о семействе алгоритмов машинного обучения, мы задаем алгоритм, как по входному объекту сделать предсказание. В этом алгоритме обычно участвуют две группы величин: признаки объекта (даны) и параметры (неизвестные величины). В ходе обучения мы настраиваем параметры по обучающей выборке, чтобы получить конкретный алгоритм предсказания. В алгоритмах предсказания и обучения могут также присутствовать величины, которые мы должны задать сами, не по данным; их называют гиперпараметры.

В решающих деревьях мы задали алгоритм предсказания как проход вниз по дереву, в каждой вершине которого находится предикат или предсказание. Вот эти придикаты и предсказания и есть параметры решающего дерева (они настраиваются по обучающей выборке). Для самого популярного вида предикатов [j-й признак < t] параметрами являются признаки и пороги в каждой внутренней вершине, а также константные предсказания в листовых вершинах.

Гиперпараметрами решающего дерева являются все детали обучения: критерии останова, критерий информативности и т. д. Используя разные критерии останова и криерии информативности, мы получим разные

Практическая часть

В практической части мы реализуем часть алгоритма обучения решающего дерева: мы выполним построение корневой вершины дерева на конкретных данных. Мы воспользуемся алгоритмом, который разбирался в теоретической части (перебор всех признаков и порогов и подсчет критерия Q). После выбора оптимального признака и порога, мы визуализируем разделение выборки по этому признаку и порогу.

Будем использовать данные boston. В этой задаче нужно предсказать стоимость жилья (задача регрессии).

In [9]:
```
import pandas as pd
import numpy as np
from sklearn.datasets import load_boston
import re
from sklearn.model_selection import train_test_split
```

In [10]:
```
boston = load_boston()
```

In [11]:
```
boston.keys()
```

Out[11]:
```
dict_keys(['data', 'target', 'feature_names', 'DESCR'])
```

Разделим выборку на обучение и контроль:

In [12]:
```
X_tr, X_te, y_tr, y_te = train_test_split(boston["data"], boston["target"])
```

Преобразуем в pandas-датафрейм:

In [13]:
```
data_train = pd.DataFrame(X_tr, columns=boston["feature_names"])
data_test = pd.DataFrame(X_te, columns=boston["feature_names"])
```

In [14]:
```
data_train.head()
```

Out[14]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.03578	20.0	3.33	0.0	0.4429	7.820	64.5	4.6947	5.0	216.0	14.9	387.31	3.76
1	0.21124	12.5	7.87	0.0	0.5240	5.631	100.0	6.0821	5.0	311.0	15.2	386.63	29.93
2	0.02543	55.0	3.78	0.0	0.4840	6.696	56.4	5.7321	5.0	370.0	17.6	396.90	7.18
3	1.19294	0.0	21.89	0.0	0.6240	6.326	97.7	2.2710	4.0	437.0	21.2	396.90	12.26
4	0.13058	0.0	10.01	0.0	0.5470	5.872	73.1	2.4775	6.0	432.0	17.8	338.63	15.37

Число объектов и признаков:

In [15]:
```
data_train.shape
```

Out[15]:
```
(379, 13)
```

In [16]:
```
from matplotlib import pyplot as plt
%matplotlib inline
```

Далее мы будем реализовывать первый шаг в построении решающего дерева - выбор признака и порога для разделения в корневой веришне дерева. Мы будем писать максимально понятный, но не оптимальный код: в sklearn алгоритм реализовано более эффективно.

In [17]:
```
# чтобы было удобно сортировать объекты вместе с целевым вектором,
# допишем его в датафрейм
data_train["target"] = y_tr
```

In [18]:
```
# чтобы было удобно перебирать порог на первый признак,
# сортируем датафрейм по нему
data_train.sort_values("CRIM", inplace=True)
```

In [19]:
```
data_train.head()
```

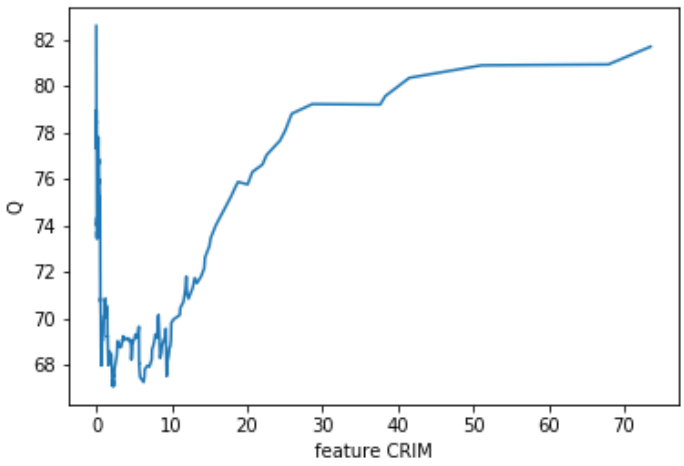
Out[19]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
272	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
77	0.01096	55.0	2.25	0.0	0.389	6.453	31.9	7.3073	1.0	300.0	15.3	394.72	8.23
66	0.01301	35.0	1.52	0.0	0.442	7.241	49.3	7.0379	1.0	284.0	15.5	394.74	5.49
154	0.01381	80.0	0.46	0.0	0.422	7.875	32.0	5.6484	4.0	255.0	14.4	394.23	2.97
140	0.01432	100.0	1.32	0.0	0.411	6.816	40.5	8.3248	5.0	256.0	15.1	392.90	3.95

In [20]:
```
# перебираем все возможные разбиения по первому признаку
# для каждого разбиения вычисляем Q, как в теоретической части:
# сначала для левого и правого поддерева считаем критерий информативности – дисперсию ответов
# затем складываем значения критерия информативности с весами
quals = []
for i in range(data_train.shape[0]):
    quality = data_train["target"][:i].std()**2 * i/data_train.shape[0] + \
    data_train["target"][i:].std()**2 * (1-i/data_train.shape[0])
    quals.append(quality)
```

In [22]:
```
# рисуем график порог на первый признак – критерий Q
plt.plot(data_train["CRIM"], quals)
plt.xlabel("feature CRIM")
plt.ylabel("Q")
```

Out[22]:
```
Text(0,0.5,'Q')
```



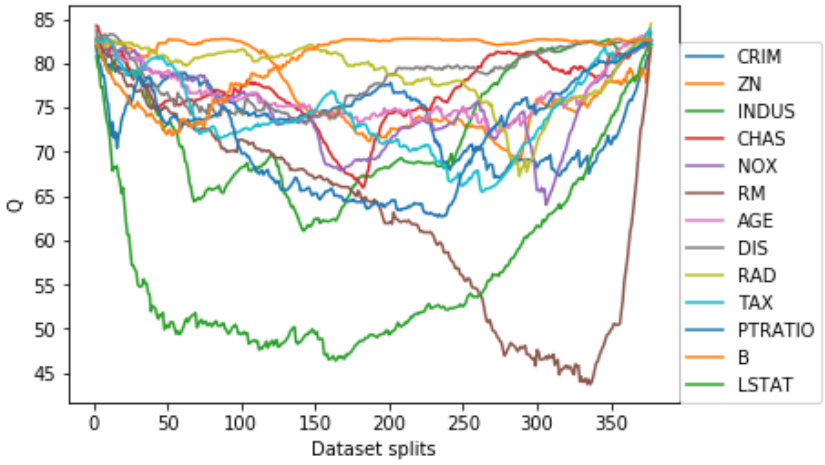
Чем меньше, тем лучше, поэтому оптимум остигается где-то в CRIM = 7.

Теперь повторим процедуру вычисления критерия Q для каждого признака.

Обратите внимание: чтобы было удобно сравнивать значение критерия для разных признаков, мы все рисуем на одном графике. Но шкала (множество значений) у каждого признака своя. Так что мы будем откладывать по оси x просто числа от 0 до длины выборки, и величину оптимального порога по графику будет определить нельзя. По графику мы сможем определить только оптимальный признак для разделения.

In [23]:
```
for feat in data_train.columns[:-1]:
    quals = []
    data_train.sort_values(feat, inplace=True)
    for i in range(data_train.shape[0]):
        quality = data_train["target"][:i].std()**2 * i/data_train.shape[0] + dat
a_train["target"][i:].std()**2 * (1-i/data_train.shape[0])
        quals.append(quality)
    plt.plot(quals, label=feat)
plt.xlabel("Dataset splits")
plt.ylabel("Q")
plt.legend(loc=(1, 0))
```

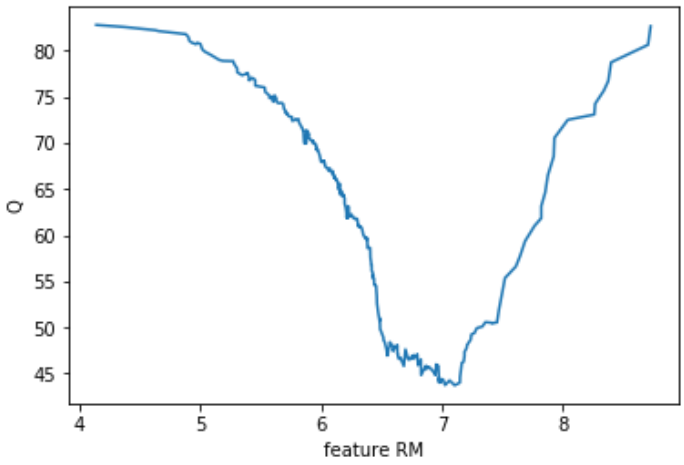
Out[23]: <matplotlib.legend.Legend at 0x1a1e43a438>



Наименьшее значение критерия из всех линий достигается на коричневой линии RM. Нарисуем для него график отдельно (уже с его осью значений порога):

In [42]:
```
feat = "RM"
quals = []
data_train.sort_values(feat, inplace=True)
for i in range(data_train.shape[0]):
    quality = data_train["target"][:i].std()**2 * i/data_train.shape[0] + \
    data_train["target"][i:].std()**2 * (1-i/data_train.shape[0])
    quals.append(quality)
plt.plot(data_train[feat], quals)
plt.xlabel("feature "+feat)
plt.ylabel("Q")
```

Out[42]:
```
Text(0,0.5,'Q')
```



Оптимальная величина порога:

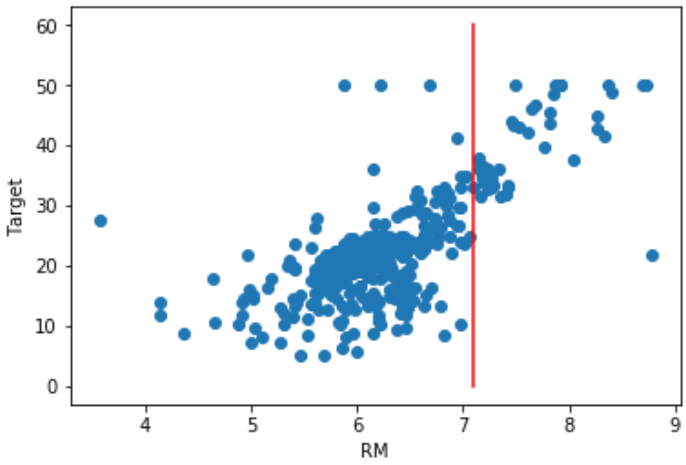
In [44]:
```
not_nan_mask = np.logical_not(np.isnan(quals))
quals = np.array(quals)[not_nan_mask]
threshs = data_train[feat].values[not_nan_mask]
thresh = threshs[np.argmin(quals)]
print(thresh)
```

```
7.104
```

Нарисуем выборку в осях RM - target и изобразим порог, по которому мы выполнили разделение:

In [45]:
```
plt.scatter(data_train["RM"], data_train["target"])
plt.plot([thresh, thresh], [0, 60], color="red")
plt.xlabel("RM")
plt.ylabel("Target")
```

Out[45]: Text(0,0.5,'Target')



Видно, что точки справа от красной линии лежат почти все выше 30, а слева - ниже, т. е. этот признак действительно очень хорошо разделяет выборку.