Анализ данных

Хашин С.И.
http://math.ivanovo.ac.ru/dalgebra/Khashin/index.html

Ивановский университет

.

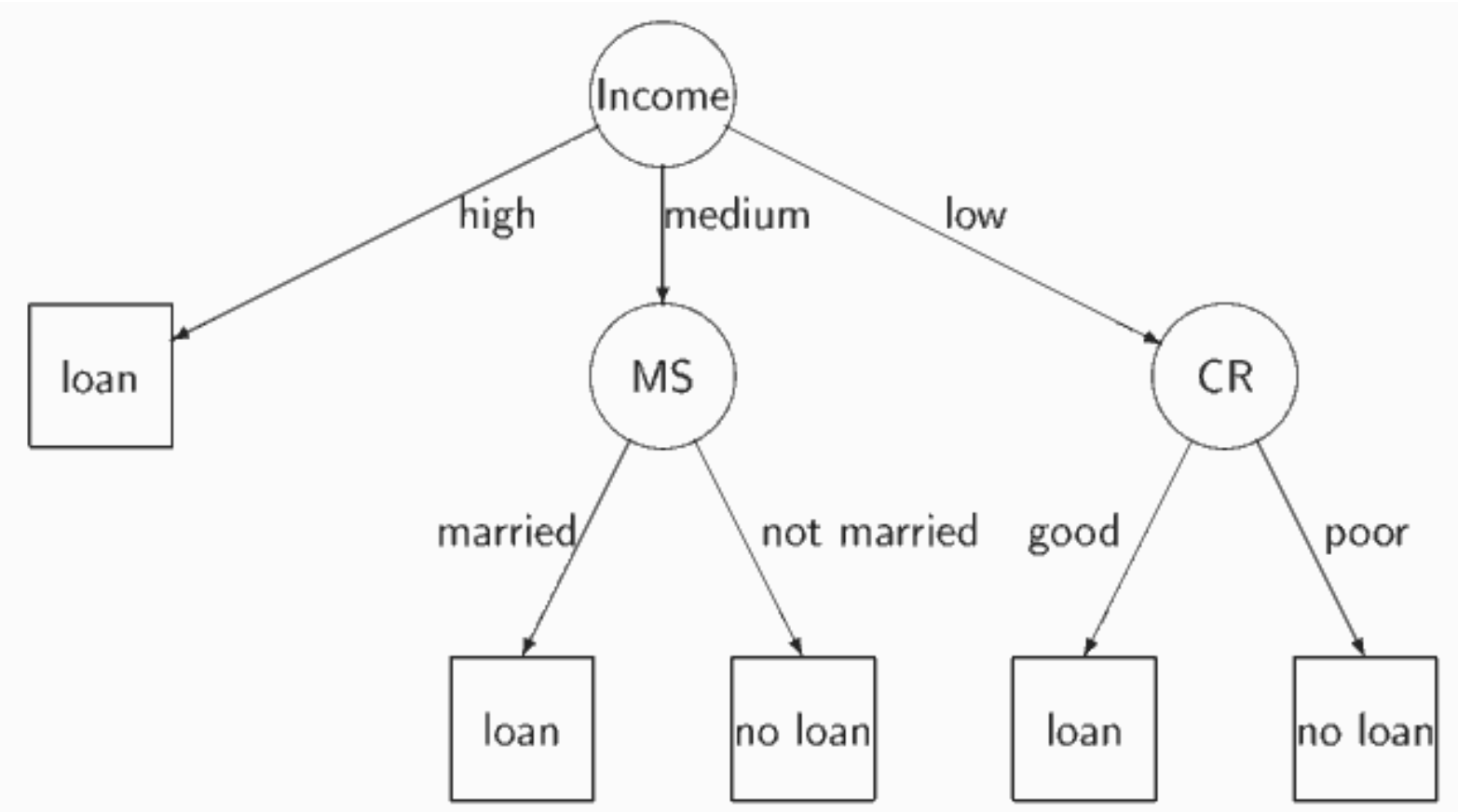
Решающие деревья

.

Иваново-2023

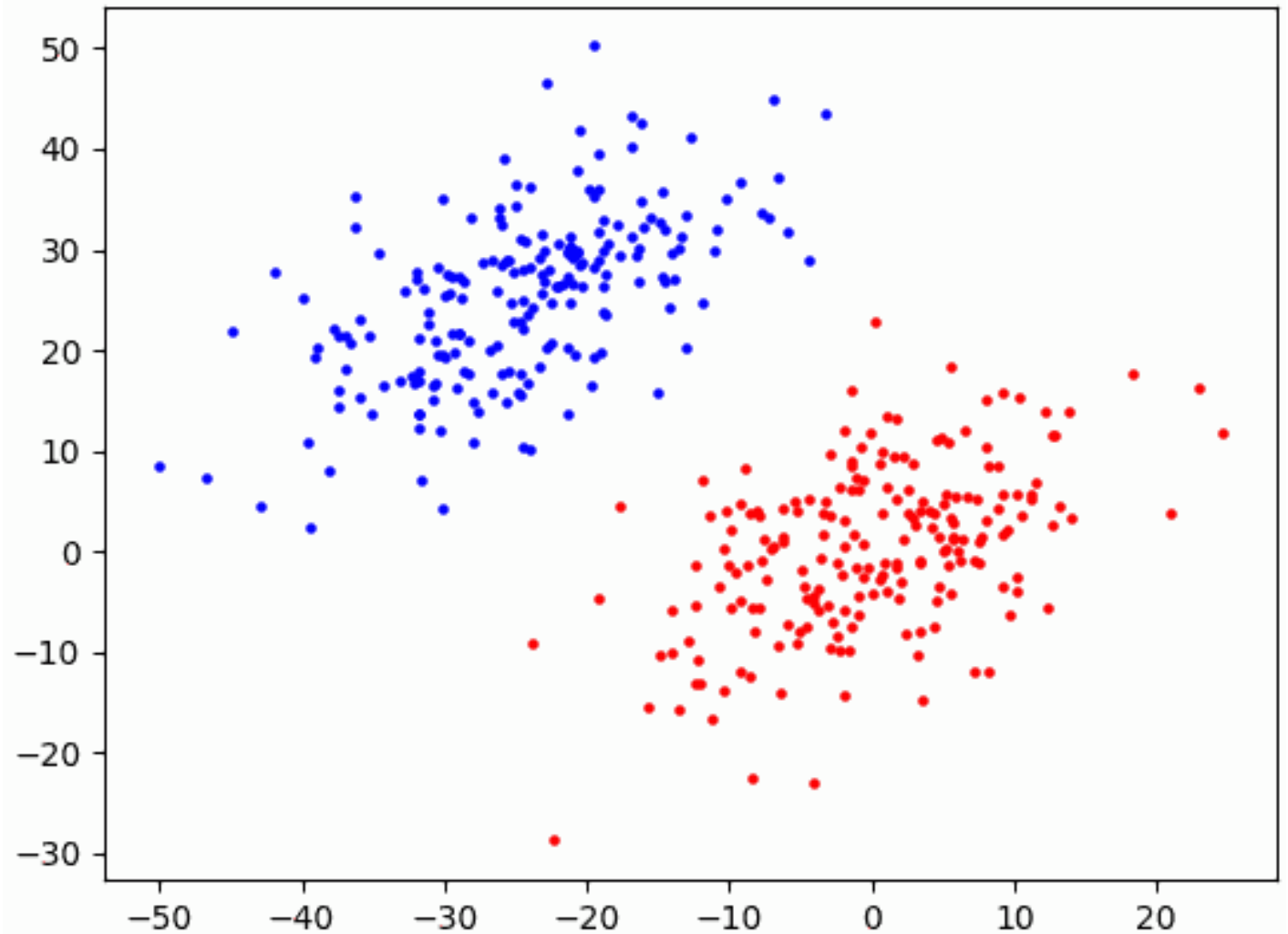
План

Пример решающего дерева



Решающие деревья

Надо разделить синие и красные точки вертикальными и горизонтальными линиями.



Азбука Морзе

Азбука Морзе разработана Сэмюэлем Морзе еще в 1838 году. Правда исходная таблица «кода Морзе» очень отличалась от тех кодов, что используются сегодня. Азбука Морзе является в сущности 'четверичной' – ее алфавит состоит из четырех знаков: точки, тире, паузы, разделяющие буквы и паузы, разделяющей слова. Длительность тире равна трём точкам. Пауза между элементами одного знака — одна точка, между знаками в слове — 3 точки, между словами — 7 точек. Азбука Морзе – первый пример кода с переменной длиной символа.

Энтропия текста

Если источник выдает цифры 0 и 1 с одинаковой вероятностью (1/2), то количество информации, приходящейся на одну цифру равно $-\log(1/2) = 1$ биту.
Предположим, что цифры появляются с различной вероятностью, $P(0) = p$, $P(1) = q = 1 - p$. Тогда количество информации, содержащейся в одной цифре так же будет различным. Если $P(0) = 0.99$ и $P(1) = 0.01$, то количество информации, соответствующей цифре 0 будет равно $-\log(0.99) = 0.0145$ битов, а цифре 1: $-\log(0.01) = 6.64$ бита.

Энтропия текста

Пусть мы имеем достаточно длинное (длины n) сообщение в таком алфавите. В нем 0 появится в среднем $p * n$ раз, а 1 — $q * n = (1 - p) * n$ раз. Учитывая количество информации, содержащейся в каждой цифре, мы можем сказать, что общее количество информации в сообщении длины n равно

$$-p * n * \log(p) - q * n * \log(q) = n * (-p * \log(p) - q * \log(q)),$$

а среднее количество информации, приходящееся на один символ равно

$$-p * \log(p) - q * \log(q).$$

Энтропия текста

Переходя к общему случаю, мы получим следующий результат. Пусть алфавит состоит из N различных символов, которые появляются с вероятностями $p_1, ..., p_N$ **независимо** друг от друга. Тогда среднее количество информации, приходящееся на один символ равно

$$H = -p_1 * \log(p_1) - ... - p_N * \log(p_N).$$

Эта величина называется энтропией заданного потока символов.

Энтропия текста

Для двухсимвольного алфавита, если вероятности появления символов одинаковы и равны $1/2$, то энтропия потока равна 1 бит/символ. Если же $P(0) = 0.99, P(1) = 0.01$, то энтропия потока примерно равна 0.08 бит/символ. Напомню, что эти рассуждения применимы только в потоку независимых символов, когда вероятность появления очередного символа не зависит от предыдущих.

Все сказанное выше относится к случаю, когда все символы в сообщении появляются независимо друг от друга. В реальной жизни это почти всегда не так.

Цепи Маркова

В слове «информа_ия» пропущенной почти наверняка является весьма редкая буква "ц". Это означает, что вероятность появления буквы зависит от предшествующих (и последующих). На практике достаточно хорошей моделью оказывается цепь Маркова. В своем простейшем виде мы просто полагаем, что вероятность появления очередной буквы зависит лишь от предыдущей. Таким образом, чтобы задать Марковскую цепь первого порядка на множестве букв русского алфавита, нам требуется задать 32*32-матрицу P, в которой на (i, j)-м месте стоит p_{ij} - вероятность появления j-й буквы алфавита после i-й.

Код Хаффмана

Код Хаффмана сопоставляет каждому символу из входного алфавита некоторую цепочку битов, длины могут быть различны. Типичный пример - азбука Морзе. Пусть символу a_i сопоставляется цепочка битов b_i. Ясно, что выбор b_i не может быть произвольным.

Пример. Пусть в алфавите A четыре буквы: $A = (a, b, c, d)$, вероятности их появления и коды следующие:

буква	частота	код
a	1/2	0
b	1/4	10
c	1/8	110
d	1/8	111

Цепочка 'abacab' будет закодирована в виде '01001100111'.

Код Хаффмана

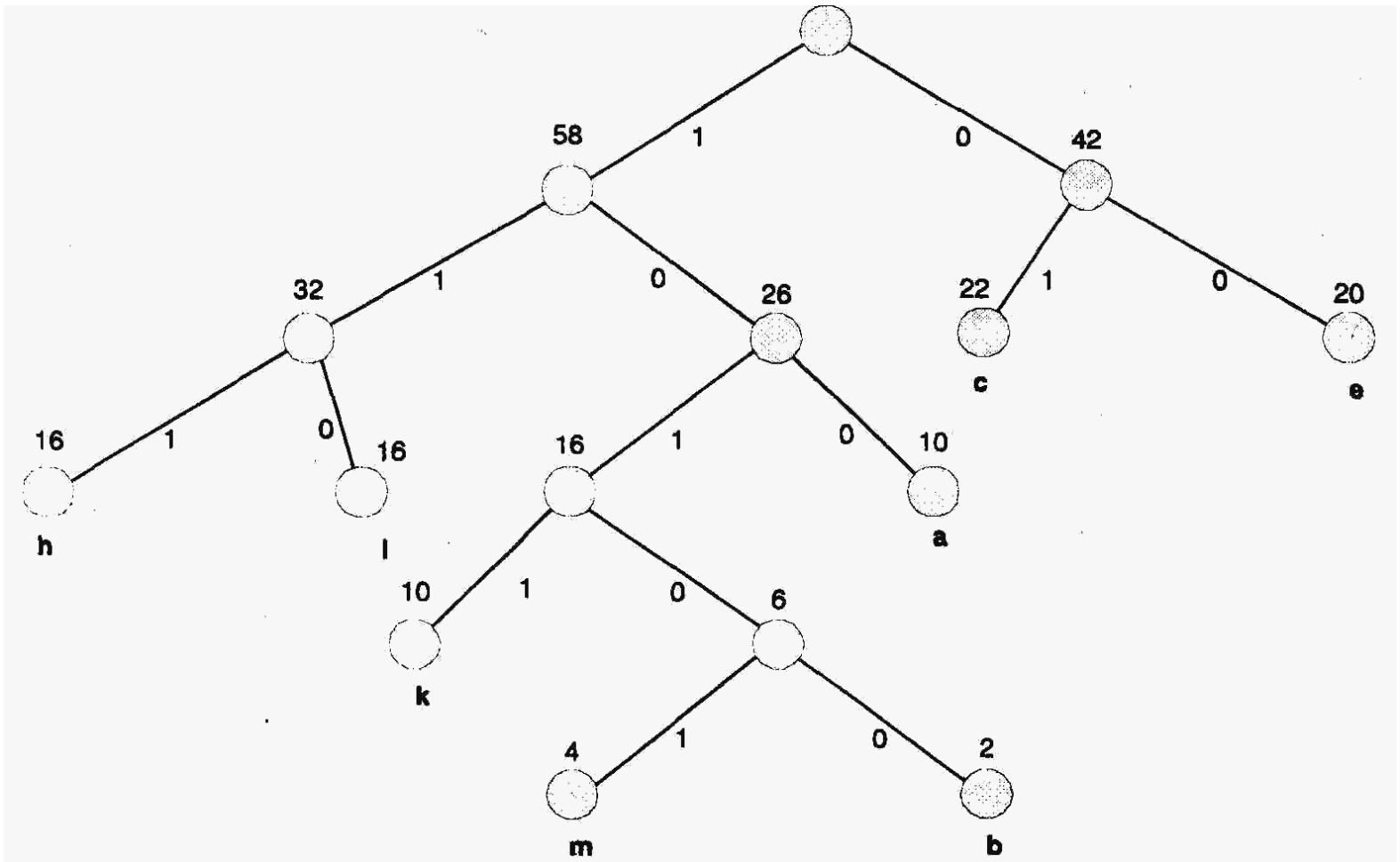
В цепочке из N букв такого алфавита будет в среднем $N/2$ букв 'a', $N/4$ букв 'b' и по $N/8$ букв 'c' и 'd'. Для кодирования такой цепочки потребуется

$$1 \cdot N/2 + 2 \cdot N/4 + 3 \cdot N/8 + 3 \cdot N/8 = 7N/4$$

битов. В тоже время при равномерном кодировании (то есть каждая буква – двумя битами) для той же цепочки потребуется $2N$ битов. Другими словами, цена равномерного кодирования – 2 бита на символ, цена кодирования с переменной длиной – 1.75 бита на символ.

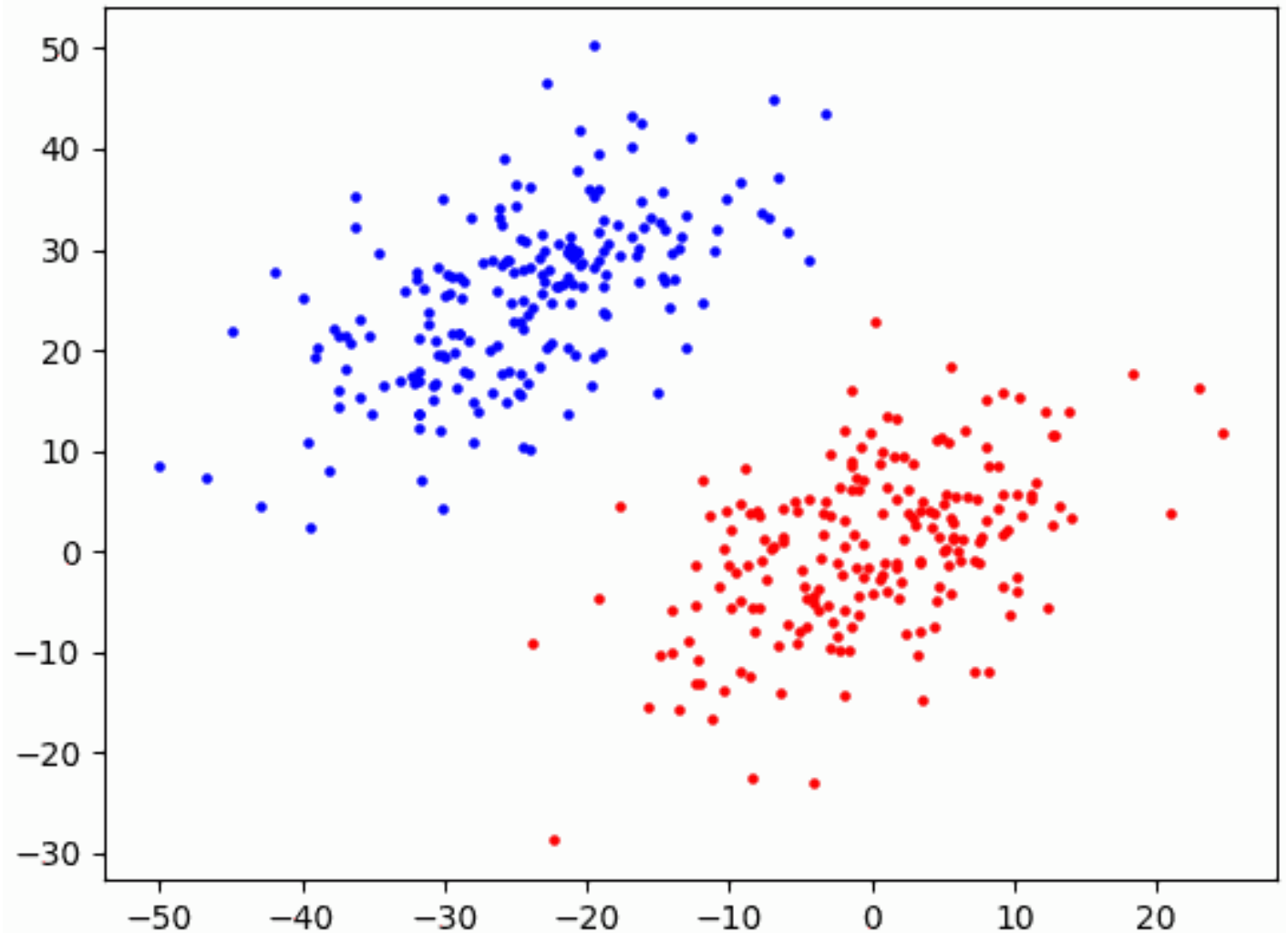
Дерево Фано

c:22 e:20 h:16 i:16 a:10 k:10 m:4 b:2



Решающие деревья

Надо разделить синие и красные точки вертикальными и горизонтальными линиями.



numpy, мелочь

```
a = np.zeros((4,2), dtype=int)
print(a)
```

```
[[0 0]
 [0 0]
 [0 0]
 [0 0]]
```

```
v = np.array((2,3))
К матрице (4,2) прибавляем вектор длины 2:
a += v
print(a)
```

```
[[2 3]
 [2 3]
 [2 3]
 [2 3]]
```

Питон, энтропия

```
def entropy(a):
    if len(a)==0: return 0
    aMin = min(a)
    if aMin<0 or aMin != int(aMin): return None
    aMax = max(a)
    if aMax != int(aMax): return None
    aMax = int(aMax)
    freqs = np.zeros(aMax+1)    # таблица частот
    for a1 in a:
        freqs[int(a1)] += 1
    res = len(a)*np.log2(len(a))
    for f1 in freqs:
        if f1>0:
            res -= f1*np.log2(f1)
    return res
```

с комментариями

```
def entropy(a):
    '''
    :param a: вектор признаков, т.е. чисел (0,1,,,k-1)
    :return: энтропия вектора
    '''
    if len(a)==0: return 0
    aMin = min(a)
    if aMin<0 or aMin != int(aMin): return None
    aMax = max(a)
    if aMax != int(aMax): return None
    aMax = int(aMax)
    freqs = np.zeros(aMax+1)    # таблица частот
    ...
```

Питон, энтропия

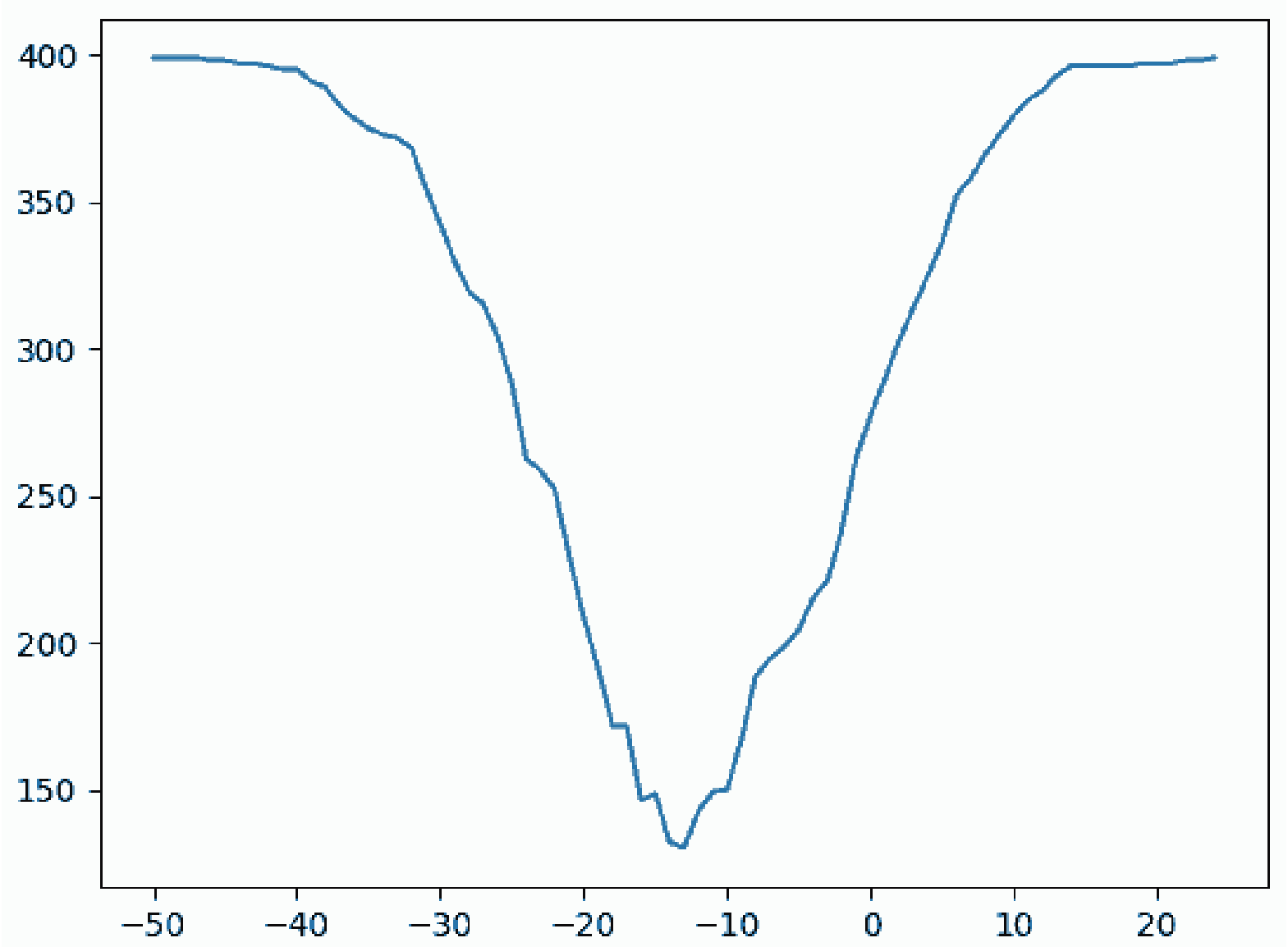
```
def entropy_a(y, x, border):
    '''
    :param y: вектор признаков, т.е. чисел 0..k
    :param x: вектор значений, т.е. действительных чисел
    :param border: действительное число, разбивающее x на дв
        разбиваем вектор x на две части: x<=border, x>borde
    :return: суммарная энтропия разделённого набора
    '''
    yMax = int(max(y))
    res = entropy( y[np.where(x<=border)[0]] )
    res += entropy( y[np.where(x>border)[0]] )
    return res
```

Питон, энтропия

```
def entropy_min(y, x):
    '''
    :param y: вектор признаков, т.е. чисел 0..k
    :param x: вектор значений, т.е. действительных чисел
    :return: (border, entropy): при каком разбиении энтропия
    '''
    xMin = int(min(x))
    xMax = int(max(x))
    entr_min = entropy(y)          # общая энтропия
    bord_min=-1                    # при каком border будет мин
    for border in range(xMin, xMax):
        e1 = entropy_a(y, x, border)
        if e1<entr_min:
            entr_min = e1
            bord_min = border
    return bord_min, entr_min
```

Решающие деревья

Зависимость энтропии от выбранной вертикальной линии:



Вот средняя часть тех же данных в табличном виде:

-16.0	-15.0	-14.0	-13.0	-12.0	-11.0	-10.0	-9.0
146.31	148.41	131.94	130.08	142.42	149.10	149.86	166.71

Таким образом, первый разрез делаем по вертикали при уровне $x = -13$.

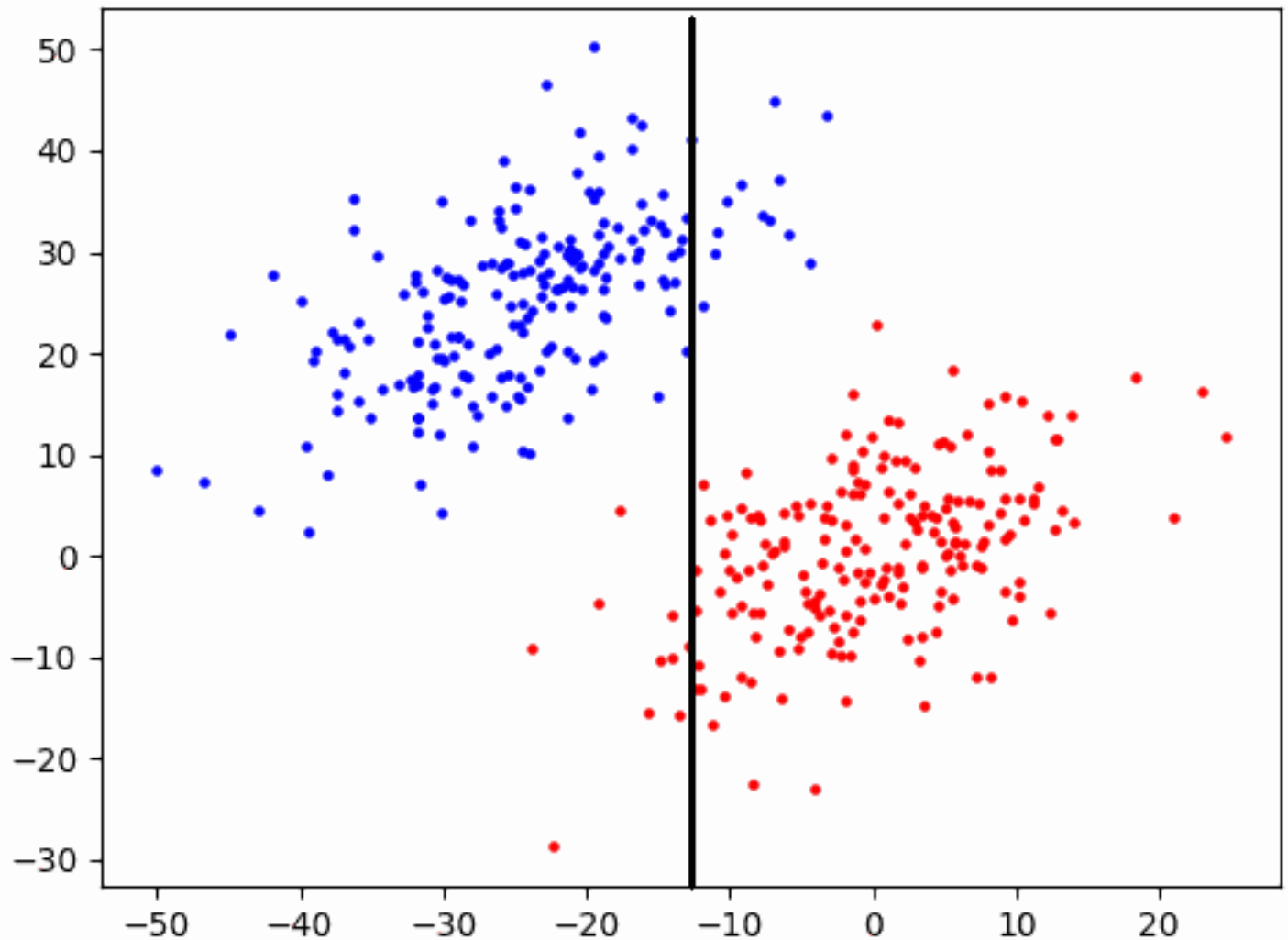
Питон, энтропия

```
csv = np.loadtxt('entropy0.csv', skiprows=1, delimiter=',')
Y = csv[:,0].astype(int)
X = csv[:,1:]
bX0, entr0 = entropy_min(Y, X[:,0])

idx = np.where(X[:,0]<=bX0)[0]        # индексы, где X[:,0]
Y0, X0 = Y[idx], X[idx]

idx = np.where(X[:,0]>bX0)[0]        # индексы, где X[:,0] >
Y1, X1 = Y[idx], X[idx]

bX00, entr00 = entropy_min(Y0, X0[:,1])
bX01, entr01 = entropy_min(Y1, X1[:,1])
```

Питон, энтропия

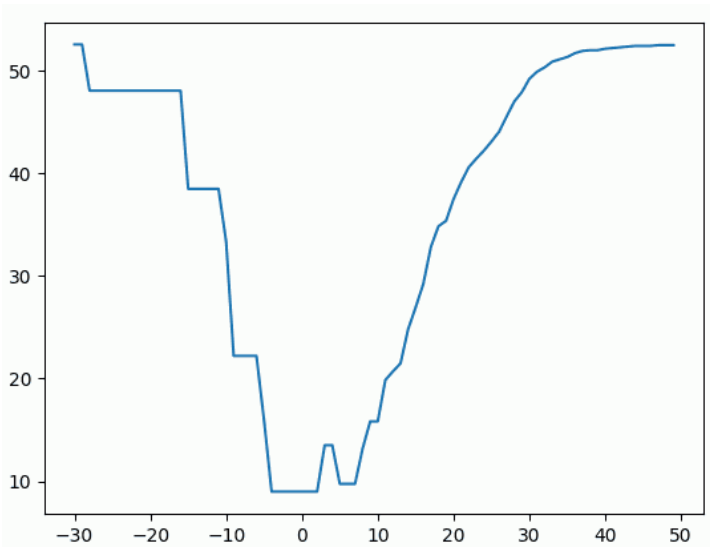
```
print(f'bX00={bX00:4.1f}, entr01={entr00:8.2f}')
bX0=-13.0, entr0=  130.08

print(f'bX01={bX01:4.1f}, entr01={entr01:8.2f}')
bX00=-4.0, entr01=    8.98
bX01=19.0, entr01=    5.40

print(f'total entropy={entr00+entr01:8.2f}')
total entropy=   14.37
```

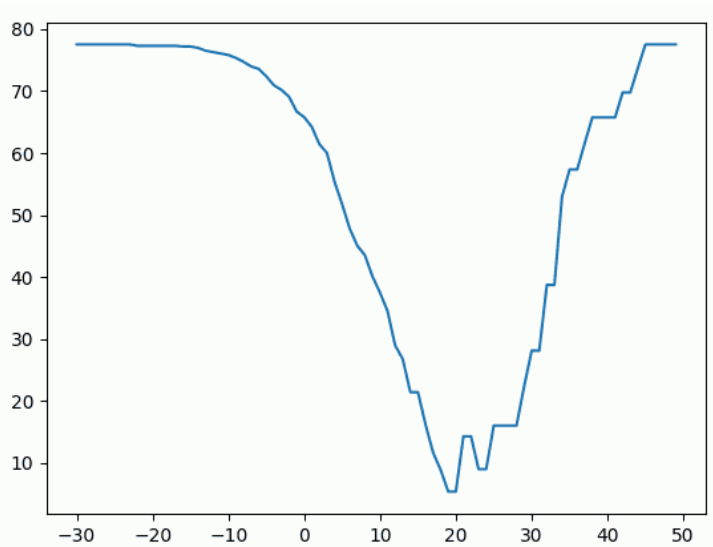
Решающие деревья



Решающие деревья

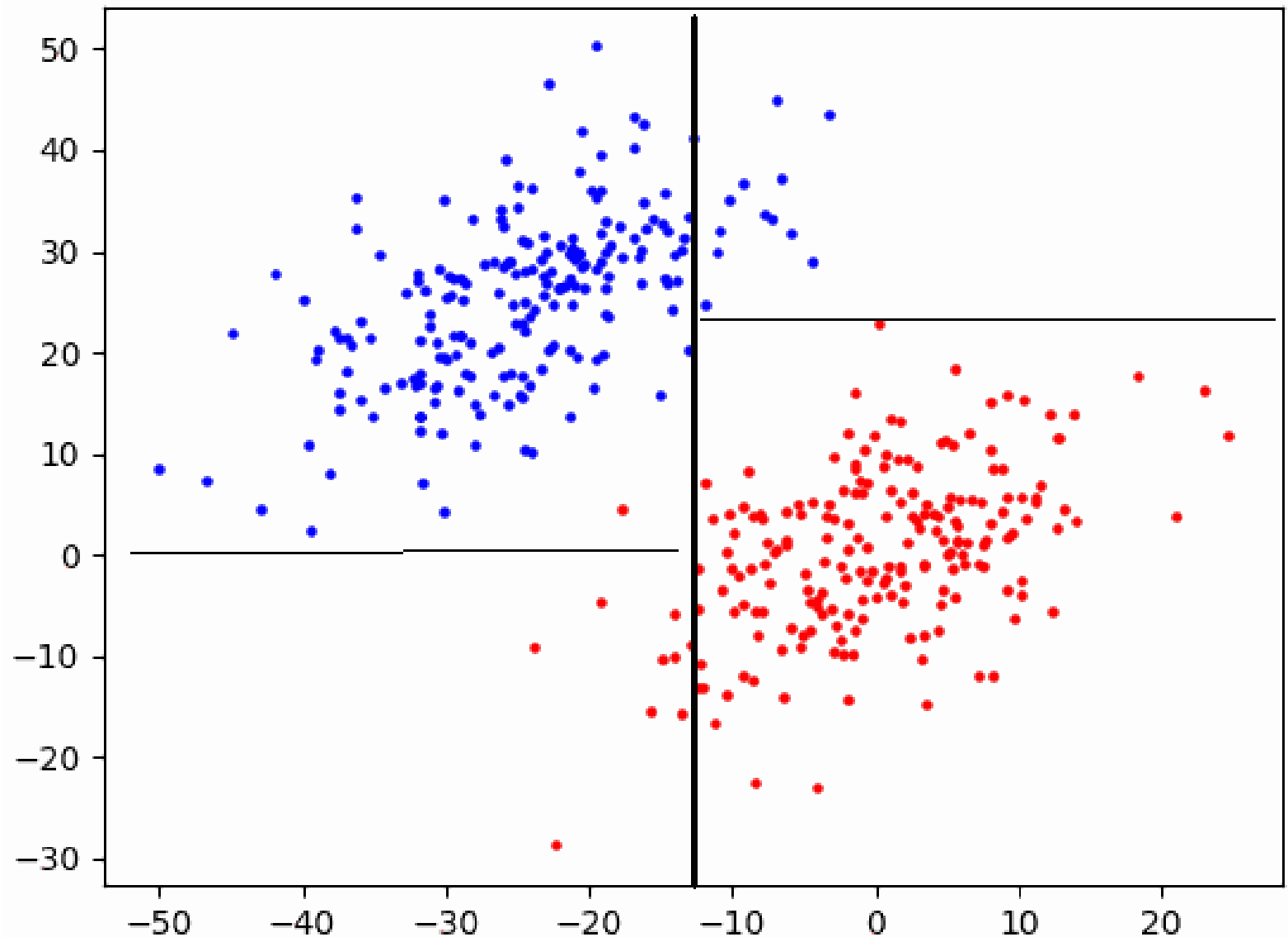
Зависимость энтропии от горизонтальных линий:





Решающие деревья

Итоговое разбиение:



Решающие деревья