



Факультет
математики
и компьютерных
наук
СПбГУ

Машинное обучение

Лекция 3. Байесовский подход. Логические методы классификации

17 сентября 2021

Пятиминутка

1. Выпишите эмпирический риск линейной модели для бинарной классификации
2. Напишите формулу отступа $M_i(w)$
3. Выпишите формулу Байеса

Принцип максимума правдоподобия (напоминание)

Пусть $X \times Y$ — вероятностное пространство, модель (т.е. её параметры w) тоже генерируются некоторым вероятностным распределением.

Формула Байеса

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$A \equiv w, B \equiv Y, X$$

$$P(w|Y, X) = \frac{P(Y, X|w)P(w)}{P(Y, X)} = \frac{P(Y|X, w)P(X|w)P(w)}{P(Y|X)P(X)}$$

$$P(w|Y, X) = \frac{P(Y|w, X)P(w)}{P(Y|X)}$$

Здесь

$P(w|Y, X)$ — апостериорное распределение параметров

$P(w)$ — априорное распределение параметров

$P(Y|X, w)$ — правдоподобие

$$\arg \max_w P(w|X, Y) = \arg \max_w P(Y|w, X)P(w) = \arg \max_w \prod_{i=1}^{\ell} P(y_i|w, x_i)P(w) =$$

$$\arg \max_w \sum_{i=1}^{\ell} \log P(y_i|w, x_i) + \log P(w)$$

Связь правдоподобия и аппроксимации эмпирического риска (напоминание)

- Максимизация правдоподобия (maximum likelihood)

$$L(w) = \sum_{i=1}^{\ell} \log P(y_i|w, x_i) \rightarrow \max_w$$

- Минимизация аппроксимированного эмпирического риска

$$Q(w) = \sum_{i=1}^{\ell} \mathcal{L}(y_i, x_i, w) \rightarrow \min_w$$

- Эти два принципа эквивалентны, если положить

$$-\log P(y_i|w, x_i) = \mathcal{L}(y_i, x_i, w)$$

Модель $P(y|x, w) \equiv$ Модель $g(x, w)$ и $\mathcal{L}(M)$

Вероятностный смысл регуляризации

$P(y|x, w)$ — вероятностная модель данных

$P(w; \gamma)$ — априорное распределение параметров модели, γ — вектор гиперпараметров;

Совместное правдоподобие данных и модели

$$P(X^\ell, w) = P(X^\ell|w)P(w; \gamma)$$

- Принцип максимума апостериорной вероятности (Maximum a Posteriori Probability):

$$L(w) = \log P(X^\ell, w) = \sum_{i=1}^{\ell} \log P(y_i|w, x_i) + \underbrace{\log P(w; \gamma)}_{\text{регуляризатор}} \rightarrow \max_w$$

Чем ещё может помочь логарифм?

```

In [24]: import matplotlib.pyplot as plt
from matplotlib import cm, ticker
from matplotlib.ticker import LinearLocator
from mpl_toolkits.mplot3d import axes3d, Axes3D

import numpy as np

fig = plt.figure()
ax = Axes3D(fig)

# Make data
x = np.arange(-2, 2, 0.25)
y = np.arange(-2, 2, 0.25)
X, Y = np.meshgrid(x, y)
R = np.sqrt(X**2 + Y**2)
Z = R**1.2 / 5

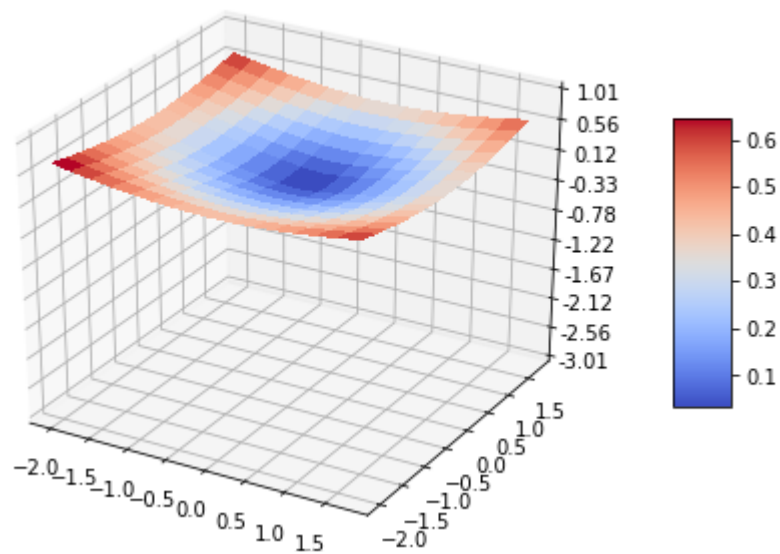
# Plot the surface
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
                      linewidth=0, antialiased=False)

# Customize the z axis.
ax.set_zlim(-3.01, 1.01)
ax.zaxis.set_major_locator(LinearLocator(10))
# A StrMethodFormatter is used automatically
ax.zaxis.set_major_formatter(ticker.StrMethodFormatter("{x:.02f}"))

# Add a color bar which maps values to colors.
fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()

```



```

In [26]: import matplotlib.pyplot as plt
from matplotlib import cm, ticker
from matplotlib.ticker import LinearLocator
from mpl_toolkits.mplot3d import axes3d, Axes3D

import numpy as np

fig = plt.figure()
ax = Axes3D(fig)

# Make data
x = np.arange(-2, 2, 0.25)
y = np.arange(-2, 2, 0.25)
X, Y = np.meshgrid(x, y)
R = np.sqrt(X**2 + Y**2)
Z = np.log(1e-2 + R**1.2 / 5)

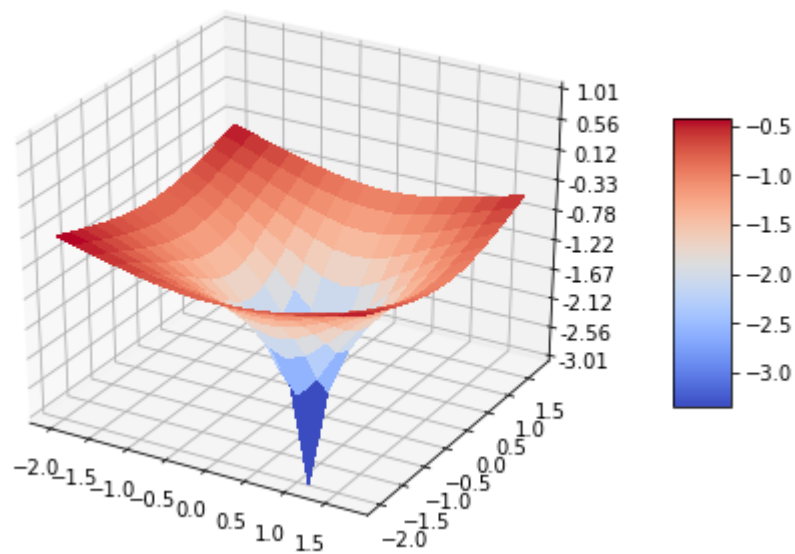
# Plot the surface
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
                      linewidth=0, antialiased=False)

# Customize the z axis.
ax.set_zlim(-3.01, 1.01)
ax.zaxis.set_major_locator(LinearLocator(10))
# A StrMethodFormatter is used automatically
ax.zaxis.set_major_formatter(ticker.StrMethodFormatter("{x:.02f}"))

# Add a color bar which maps values to colors.
fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()

```



Примеры: априорные распределения Гаусса и Лапласа

Пусть веса w_j независимы, $Ew_j = 0$, $Dw_j = C$

- Распределение Гаусса и квадратичный (L_2) регуляризатор:

$$P(w; C) = \frac{1}{(2\pi C)^{n/2}} \exp\left(-\frac{\|w\|_2^2}{2C}\right), \|w\|_2^2 = \sum_{j=1}^n w_j^2,$$

$$-\log P(w; C) = \frac{1}{2C} \|w\|_2^2 + \text{const}$$

- Распределение Лапласа и абсолютный (L_1) регуляризатор:

$$P(w; C) = \frac{1}{(2C)^n} \exp\left(-\frac{\|w\|_1}{C}\right), \|w\|_1 = \sum_{j=1}^n |w_j|,$$

$$-\log P(w; C) = \frac{1}{C} \|w\|_1 + \text{const}$$

C — гиперпараметр, $\tau = \frac{1}{C}$ — коэффициент регуляризации

Двухклассовая логистическая регрессия

- Линейная модель классификации для двух классов $Y = \{-1, 1\}$:

$$a(x) = \text{sign} \langle w, x \rangle, \quad x, w \in \mathbb{R}^n, \quad \text{отступ } M = \langle w, x \rangle y$$

- Логарифмическая функция потерь:

$$\mathcal{L}(M) = \log(1 + e^{-M})$$

- Модель условной вероятности:

$$P(y|x, w) = \sigma(M) = \frac{1}{1 + e^{-M}},$$

где $\sigma(M)$ — сигмоидная функция, важное свойство: $\sigma(M) + \sigma(-M) = 1$

- Задача обучения регуляризованной логистической регрессии (минимизация аппроксимированного эмпирического риска):

$$Q(w) = \sum_{i=1}^{\ell} \log(1 + \exp(-\langle w, x_i \rangle y_i)) + \frac{\tau}{2} \|w\|_2^2 \rightarrow \min_w$$

```

In [2]: import matplotlib.pyplot as plt
import numpy as np

fig, (ax1, ax2) = plt.subplots(1, 2)

ax1.set_xlabel('M')
ax2.set_xlabel('M')

x1 = np.linspace(-3.5, 3.5, num=1000)
L_M = np.log2(1 + np.exp(-x1)) # what if np.log used?
ax1.plot(x1, L_M, 'r', label='loss')
ax1.plot(x1, x1 < 0, 'b', label='acc loss')

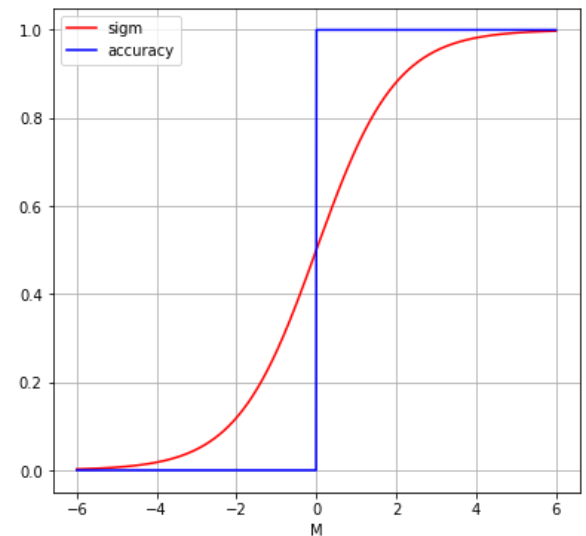
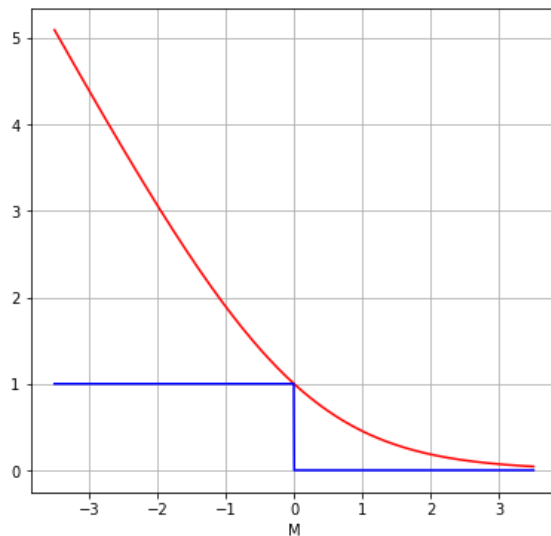
x2 = np.linspace(-6, 6, num=1000)
sigm_M = 1 / (1 + np.exp(-x2))
ax2.plot(x2, sigm_M, 'r', label='sigm')
ax2.plot(x2, x2 > 0, 'b', label='accuracy')

ax1.grid(True)
ax2.grid(True)

fig.set_size_inches(14, 6)

plt.legend(loc='best')
plt.show()

```



Многоклассовая логистическая регрессия

Линейный классификатор при произвольном числе классов $|Y|$:

$$a(x) = \arg \max_{y \in Y} \langle w_y, x \rangle, \quad x, w_y \in \mathbb{R}^n$$

Вероятность того, что объект x относится к классу y :

$$P(y|x, w) = \frac{\exp \langle w_y, x \rangle}{\sum_{z \in Y} \exp \langle w_z, x \rangle} = \underset{y \in Y}{softmax} \langle w_y, x \rangle$$

Функция $softmax : \mathbb{R}^Y \rightarrow \mathbb{R}^Y$ переводит произвольный вектор в нормированный вектор дискретного распределения.

Максимизация правдоподобия (log-loss) с регуляризацией:

$$L(w) = \sum_{i=1}^{\ell} \log P(y_i | w, x_i) - \frac{\lambda}{2} \sum_{y \in Y} \|w_y\|^2 \rightarrow \max_w$$

Пример. Бинаризация признаков и скоринговая карта

Задача кредитного скоринга

- x_i — заемщики
- y_i — вернёт кредит (+1) или нет (-1)

Признак j	Интервал k	w_{jk}
возраст	до 25	5
	25-40	10
	40-50	15
	50+	5
собственность	владелец	20
	совладелец	15
	съемщик	10
	другое	5
работа	руководитель	15
	менеджер	10
	служащий	5
	другое	0
стаж	менее 1 года	0
	1..3	5
	3..10	10
	10+	15

Бинаризация признаков $f_j(x)$: $b_{jk}(x) = [f_j(x) \text{ из } k\text{-го интервала}]$

Линейная модель классификации

$$a(x, w) = \text{sign}(\sum_{j,k} w_{jk} b_{jk}(x) - w_0)$$

Вес признака w_{jk} равен его вкладу в общую сумму баллов (score).

Оценивание рисков в скоринге

Логистическая регрессия не только определяет веса w , но и оценивает апостериорные вероятности классов

$$P(y|x) = \frac{1}{1 + \exp(-\langle w, x \rangle y)}$$

Оценка риска (математического ожидания) потерь объекта x :

$$R(x) = \sum_{y \in Y} D_{xy} P(y|x)$$

где D_{xy} — величина потери для объекта x с исходом y . Оценка говорит о том, сколько мы потеряем в среднем.

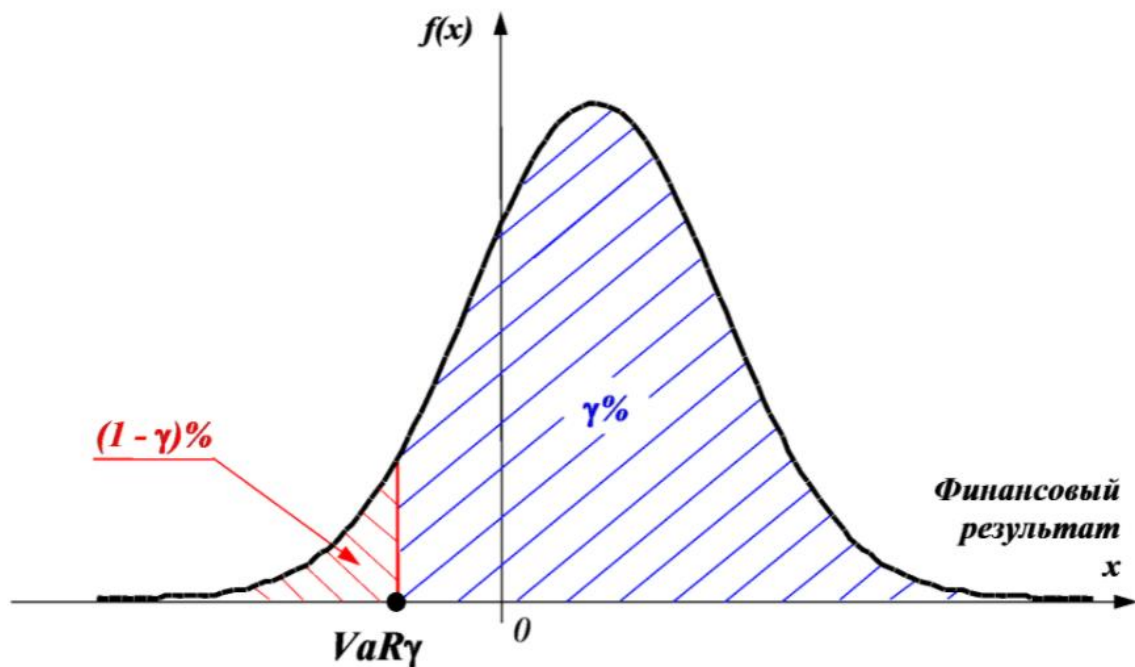
Вопрос 1: Как оценить, сколько мы потеряем в худшем случае?

Методика VaR (Value at Risk)

Стохастическое моделирование: $N = 10^4$ раз

- для каждого x_i разыгрывается исход $y_i \sim P(y|x_i)$
- вычисляется сумма потерь по портфелю $V = \sum_{i=1}^{\ell} D_{x_i y_i}$

99%-квантиль эмпирического распределения потерь определяет величину резервируемого капитала.



Логическая закономерность

$X^\ell = (x_i, y_i)_{i=1}^\ell \subset X \times Y$ — обучающая выборка, $y_i = y(x_i)$

Логическая закономерность (правило, rule) — это предикат $R : X \rightarrow \{0, 1\}$, удовлетворяющий двум требованиям:

1. интерпретируемость:

а) R записывается на естественном языке;

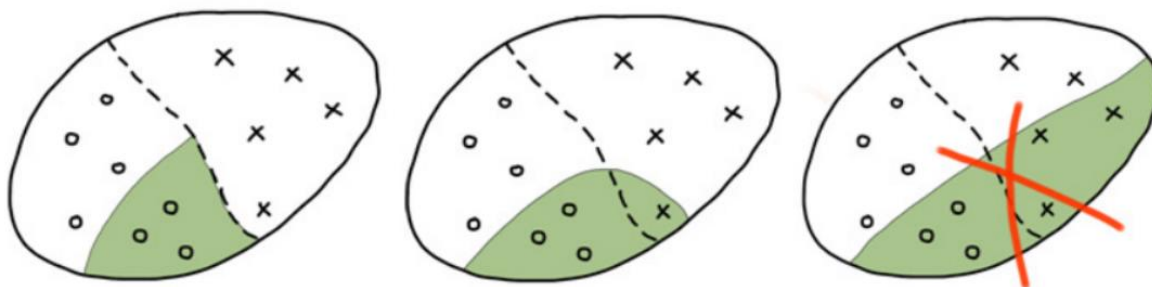
б) R зависит от небольшого числа признаков (1-7);

2. информативность относительно одного из классов $c \in Y$:

$$p_c(R) = \# \{x_i : R(x_i) = 1 \text{ и } y_i = c\} \rightarrow \max$$

$$n_c(R) = \# \{x_i : R(x_i) = 1 \text{ и } y_i \neq c\} \rightarrow \min$$

Если $R(x) = 1$, то говорят « R выделяет x » (R covers x).



Требование интерпретируемости

1) $R(x)$ записывается на естественном языке

2) $R(x)$ зависит от небольшого числа признаков (1-7)

Пример (из области медицины)

Если «возраст > 60» и «пациент ранее перенёс инфаркт», то операцию не делать, риск отрицательного исхода 60%.

Пример (из области кредитного скоринга)

Если «в анкете указан домашний телефон» и «зарплата > \$2000» **и** «сумма кредита < \$5000» то кредит можно выдать, риск дефолта 5%

Основные шаги индукции правил (rule induction)

1. Выбор семейства правил для поиска закономерностей
2. Порождение правил (rule generation)
3. Отбор правил-закономерностей (rule selection)

Закономерность — интерпретируемый высокоинформативный одноклассовый классификатор с отказами.

Выбор семейства правил

1. Пороговое условие (решающий пень, decision stump):

$$R(x) = [f_j(x) \leq a_j] \text{ или } [a_j \leq f_j(x) \leq b_j]$$

1. *Конъюнкция* пороговых условий:

$$R(x) = \bigwedge_{j \in J} [a_j \leq f_j(x) \leq b_j]$$

1. *Синдром* — выполнение не менее d условий из $|J|$ (при $d = |J|$ это конъюнкция, при $d = 1$ — дизъюнкция):

$$R(x) = \left[\sum_{j \in J} [a_j \leq f_j(x) \leq b_j] \geq d \right]$$

Параметры J, a_j, b_j, d , настраиваются по обучающей выборке путём оптимизации *критерия информативности*.

Выбор семейства правил

1. Полуплоскость — линейная пороговая функция:

$$R(x) = \left[\sum_{j \in J} w_j f_j(x) \geq w_0 \right]$$

1. *Шар* — пороговая функция близости:

$$R(x) = [(\rho(x, x_0) \leq w_0]$$

ABO — алгоритмы вычисления оценок [Ю.И. Журавлёв, 1971]

$$\rho(x, x_0) = \max_{j \in J} w_j |f_j(x) - f_j(x_0)|$$

SCM — машины покрывающих множеств [M. Marchand, 2001]

$$\rho(x, x_0) = \sum_{j \in J} w_j |f_j(x) - f_j(x_0)|^\gamma$$

Параметры J, w_j, w_0, x_0, γ , настраиваются по обучающей выборке путём оптимизации *критерия информативности*.

Порождение правил

Вход: обучающая выборка X^ℓ

Выход: множество закономерностей Z

1: начальное множество правил Z

2: **повторять**

3: $Z' :=$ множество локальных модификаций правил $R \in Z$

4: удалить слишком похожие правила из $Z \cup Z'$

5: $Z :=$ наиболее информативные правила из $Z \cup Z'$

6: **пока** правила продолжают улучшаться

7: **вернуть** Z

Частные случаи оптимизации

- генетические (эволюционные) алгоритмы
- управляемый локальный поиск (GLS) — меняем штрафы в процессе поиска
- метод ветвей и границ — отсекаем области, заведомо не содержащие оптимальных решений

Вопрос 2: Где используются упомянутые частные случаи?

Локальные модификации правил

Пример. Семейство конъюнкций пороговых условий:

$$R(x) = \bigwedge_{j \in J} [a_j \leq f_j(x) \leq b_j]$$

Локальные модификации конъюнктивного правила:

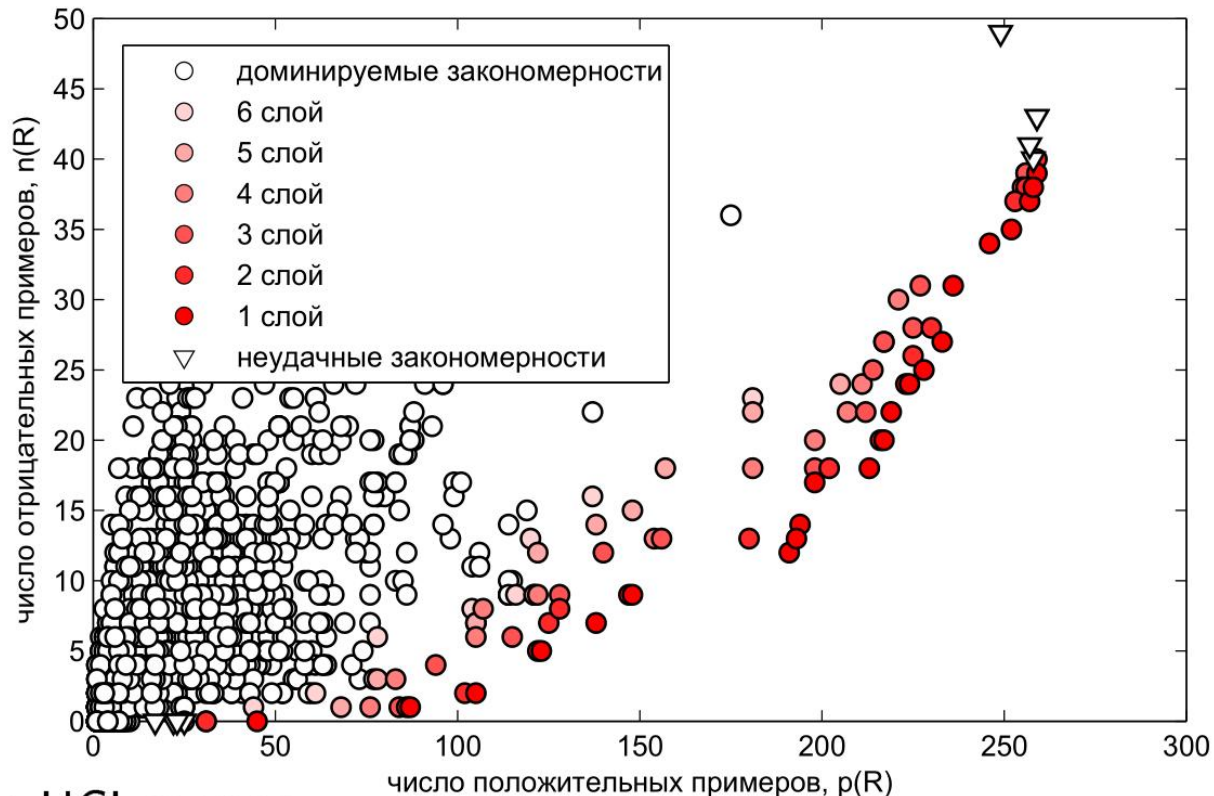
- варьирование одного из порогов a_j или b_j
- варьирование обоих порогов a_j и b_j одновременно
- добавление признака f_j в J с варьированием порогов a_j, b_j
- удаление признака f_j из J

При удалении признака (pruning) информативность обычно оценивается по контрольной выборке (hold-out)

Для оптимизации множества J подходят те же методы, что и для отбора признаков (feature selection)

Отбор закономерностей по паре критериев $p_c \rightarrow \max, n_c \rightarrow \min$

Парето-фронт — множество неулучшаемых закономерностей (точка неулучшаема, если правее и ниже неё точек нет)



задача UCI:german

Проблема: хотелось бы иметь один скалярный критерий

Комплексные признаки информативности

- энтропийный критерий прироста информации:

$$IGain(p, n) = h\left(\frac{P}{\ell}\right) - \frac{p+n}{\ell} h\left(\frac{p}{p+n}\right) - \frac{\ell-p-n}{\ell} h\left(\frac{P-p}{\ell-p-n}\right) \rightarrow \max$$

$h(q) = -q \log_2 q - (1 - q) \log_2 (1 - q)$ — энтропия «распределения двух классов»

- критерий Джини (Gini impurity):

$$IGini(p, n) = IGain(p, n) \text{ при } h(q) = 4q(1 - q)$$

- точный статистический тест Фишера (Fisher's Exact Test):

$$IStat(p, n) = -\frac{1}{\ell} \log_2 \frac{C_P^p C_N^n}{C_{P+N}^{p+n}} \rightarrow \max \text{ — это просто вероятность реализации такой пары } p, n$$

- критерий бустинга [Cohen, Singer, 1999 \(http://www.cs.utsa.edu/~bylander/cs6243/aaai-99-slipper.pdf\)](http://www.cs.utsa.edu/~bylander/cs6243/aaai-99-slipper.pdf)

$$\sqrt{p} - \sqrt{n} \rightarrow \max$$

- нормированный критерий бустинга:

$$\sqrt{\frac{p}{P}} - \sqrt{\frac{n}{N}} \rightarrow \max$$


```

In [27]: import matplotlib.pyplot as plt

_, ax = plt.subplots()

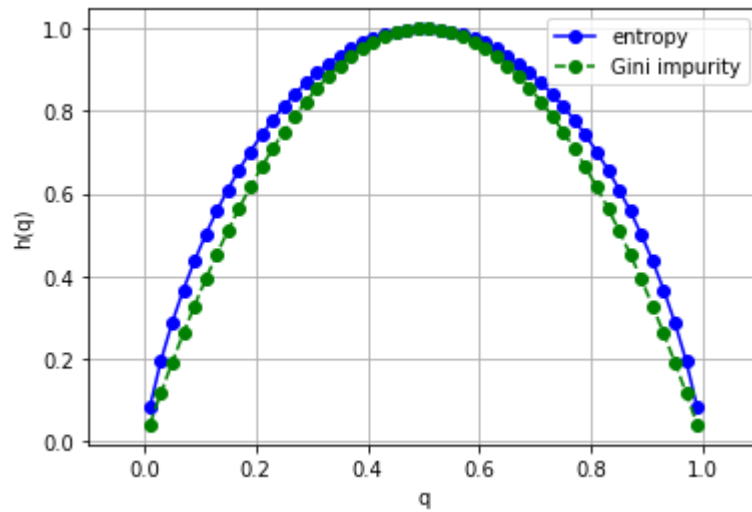
q = np.linspace(0.01, 0.99, num=50)
h_q = -q * np.log2(q) - (1 - q) * np.log2(1 - q)
h_q_2 = 4 * q * (1 - q)

ax.set_xlabel('q')
ax.set_xlim(-0.1, 1.1)
ax.set_ylabel('h(q)')

ax.plot(q, h_q, 'bo-', label='entropy')
ax.plot(q, h_q_2, 'go--', label='Gini impurity')
ax.grid(True)

plt.legend(loc='best')
plt.show()

```



Комбинированное использование закономерностей

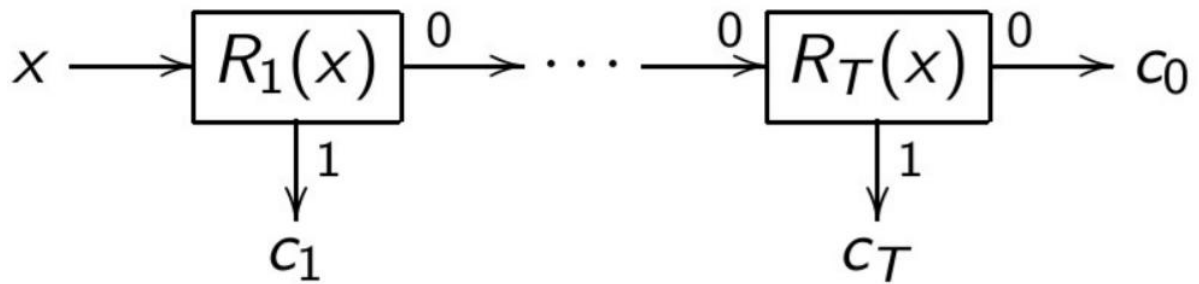
- Взвешенное голосование (линейный классификатор с весами w_{yt}):

$$a(x) = \arg \max_{y \in Y} \sum_{t=1}^{T_y} w_{yt} R_{yt}(x)$$

- Простое голосование (комитет большинства)

$$a(x) = \arg \max_{y \in Y} \frac{1}{T_y} \sum_{t=1}^{n_y} R_{yt}(x)$$

- Решающий список (комитет старшинства), $c_t \in Y$:



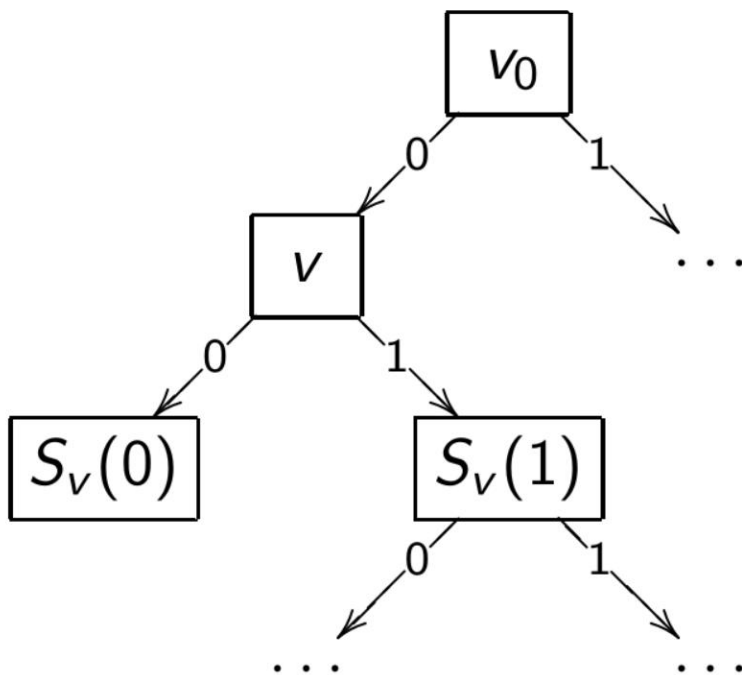
Решающее дерево (Decision Tree)

Решающее дерево — алгоритм классификации $a(x)$, задающийся деревом (связным ациклическим графом)

1. $V = V_{\text{внутр}} \cup V_{\text{лист}}$, $v_0 \in V$ — корень дерева
2. $v \in V_{\text{внутр}}$: функции $f_v : X \rightarrow D_v$ и $S_v : D_v \rightarrow V$, $|D_v| < \infty$
3. $v \in V_{\text{лист}}$: метка класса $y_v \in Y$

Частный случай: $D_v = \{0, 1\}$ — бинарное решающее дерево

Пример: $f_v(x) = [f_j(x) \geq \theta_j]$



Пример

Ирисы Фишера (https://ru.wikipedia.org/wiki/Ирисы_Фишера) – классический датасет для классификации.

Классы: Ирис щетинистый (*Iris setosa*), Ирис виргинский (*Iris virginica*), Ирис разноцветный (*Iris versicolor*)

Признаки: Длина/ширина чашелистика, Длина/ширина лепестка.



```
In [3]: import numpy as np

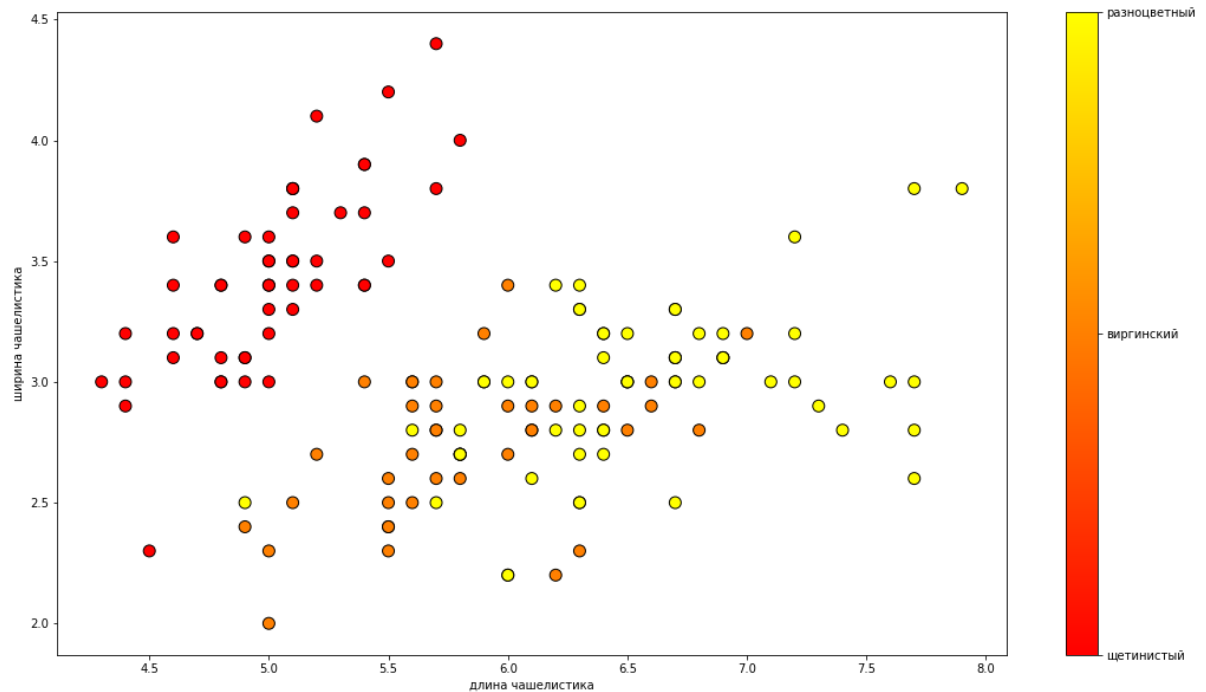
from sklearn.datasets import load_iris
X, y = load_iris(return_X_y=True)

features = np.array(["длина чашелистика", "ширина чашелистика",
                    "длина лепестка", "ширина лепестка"])
labels = np.array(["щетинистый", "виргинский", "разноцветный"])
```

```
In [1]: def show_legend():
        cb = plt.colorbar()
        loc = [0, 1, 2]
        cb.set_ticks(loc)
        cb.set_ticklabels(labels)
```

```
In [7]: import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(18.0, 10.0))
plt.scatter(X[:, 0], X[:, 1], c=y, s=100, cmap='autumn', edgecolors="black")
plt.xlabel(features[0])
plt.ylabel(features[1])
show_legend()
```



```
In [8]: from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state=0, C=1.0).fit(X, y)
clf.predict(X)
```

C:\Users\avalur\Anaconda3\lib\site-packages\sklearn\linear_model_logistic.p
y:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)

```
Out[8]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1,
               1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2,
               2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
               2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```
In [10]: clf.predict_proba(X[:2, :])
```

```
Out[10]: array([[9.81797141e-01, 1.82028445e-02, 1.44269293e-08],  
               [9.71725476e-01, 2.82744937e-02, 3.01659208e-08]])
```

```
In [11]: clf.score(X, y)
```

```
Out[11]: 0.9733333333333334
```

Вопрос 3: Как считается score?

Ответ (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.score)
mean accuracy

Напишем вспомогательную функцию, которая будет возвращать решетку для дальнейшей красивой визуализации

```
In [12]: def get_grid(data):  
         x_min, x_max = data[:, 0].min() - 1, data[:, 0].max() + 1  
         y_min, y_max = data[:, 1].min() - 1, data[:, 1].max() + 1  
         return np.meshgrid(np.arange(x_min, x_max, 0.01),  
                             np.arange(y_min, y_max, 0.01))
```

Напишем функцию для обучения на данных, предсказания ответа для каждой точки решетки и визуализации результата. Т.к. данные гораздо лучше разделяются по лепесткам, визуализируем только их (параметры чашелистика возьмем средние).

```

In [13]: def plot_model(X, y, clf, proba=False, legend=True):
    clf.fit(X, y)
    xx2, xx3 = get_grid(X[:, [2,3]])
    shape = xx2.ravel().shape
    grid_xs = np.c_[np.full(shape, X[:,0].mean()),
                    np.full(shape, X[:,1].mean()),
                    xx2.ravel(),
                    xx3.ravel()]
    predicted = clf.predict(grid_xs).reshape(xx2.shape)
    if proba:
        probs = clf.predict_proba(grid_xs)
        confidence = probs.max(axis=1).reshape(xx2.shape)
        mesh = plt.pcolormesh(xx2, xx3, confidence, cmap='autumn')
    else:
        mesh = plt.pcolormesh(xx2, xx3, predicted, cmap='autumn')
    plt.scatter(X[:, 2], X[:, 3], c=y, s=150, cmap='autumn', alpha=0.7, edgeco
lors="black")
    plt.ylim([xx3.min(),xx3.max()])
    plt.xlim([xx2.min(),xx2.max()])
    plt.xlabel(features[2])
    plt.ylabel(features[3])
    if legend:
        if not proba:
            show_legend()
        else:
            cb = plt.colorbar(mesh)
    return clf

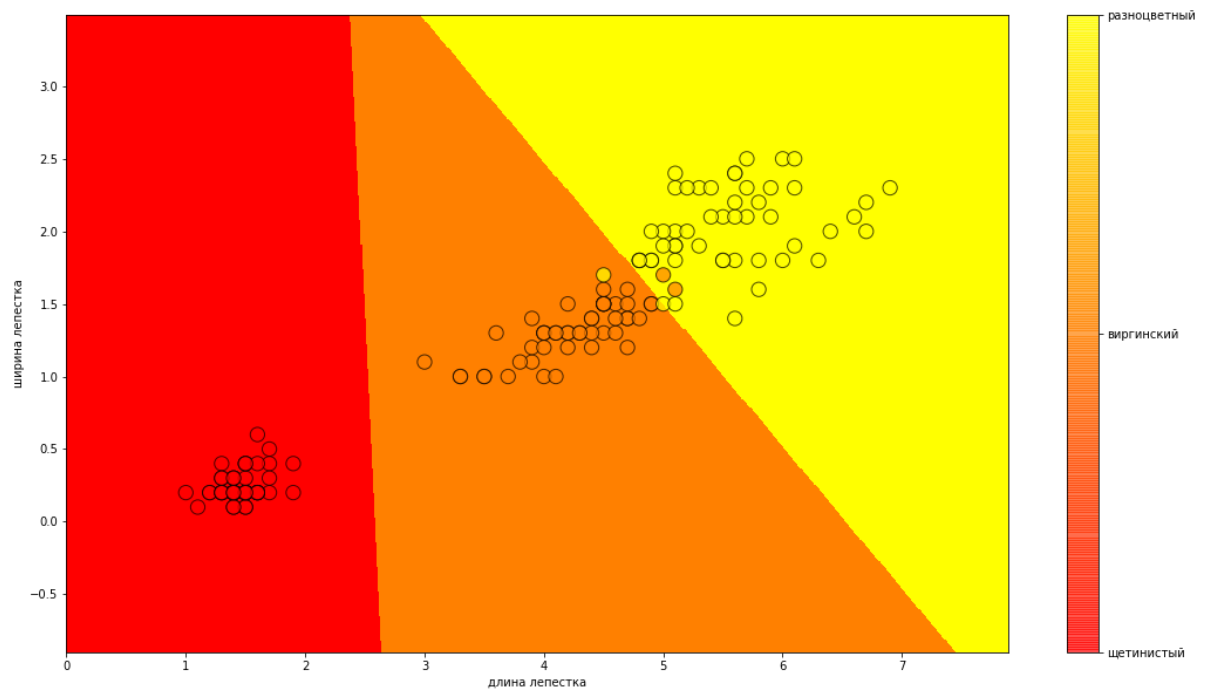
```

```
In [14]: plt.figure(figsize=(18.0, 10.0))
plot_model(X, y, LogisticRegression())
```

```
C:\Users\avalur\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
[https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
sion
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)

```
Out[14]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, l1_ratio=None, max_iter=100,
multi_class='auto', n_jobs=None, penalty='l2',
random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
warm_start=False)
```

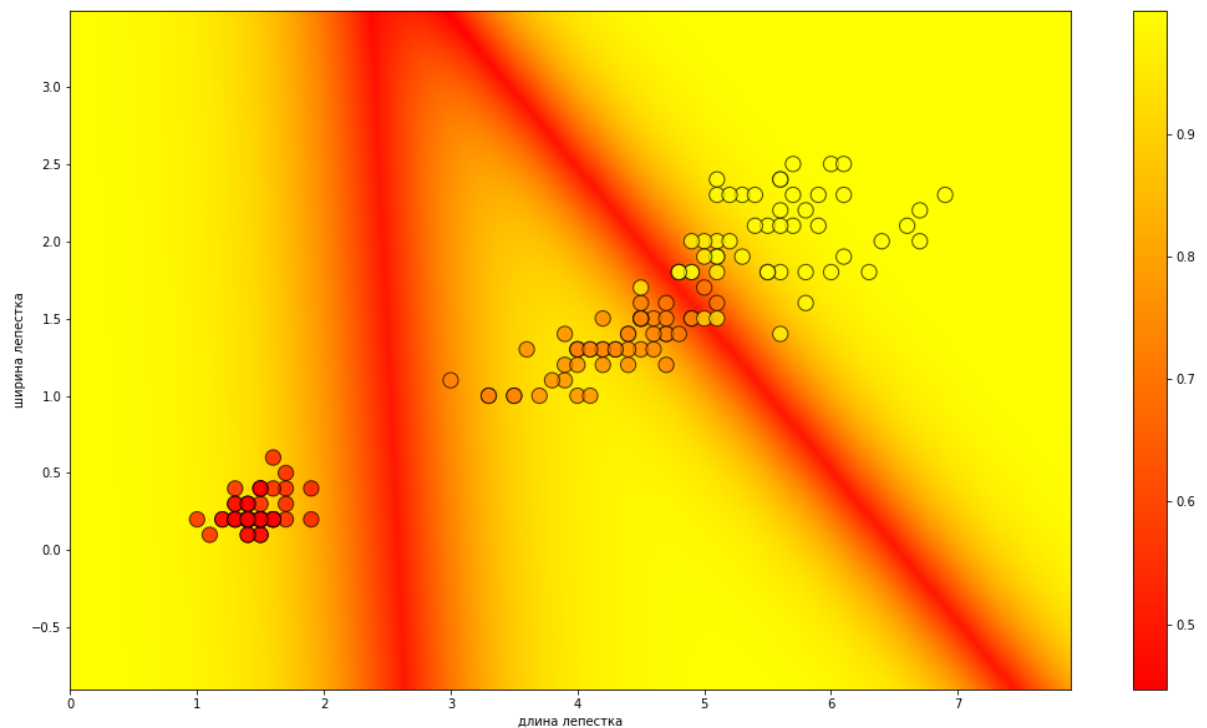



```
In [15]: plt.figure(figsize=(18.0, 10.0))
plot_model(X, y, LogisticRegression(), proba=True)
```

```
C:\Users\avalur\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)

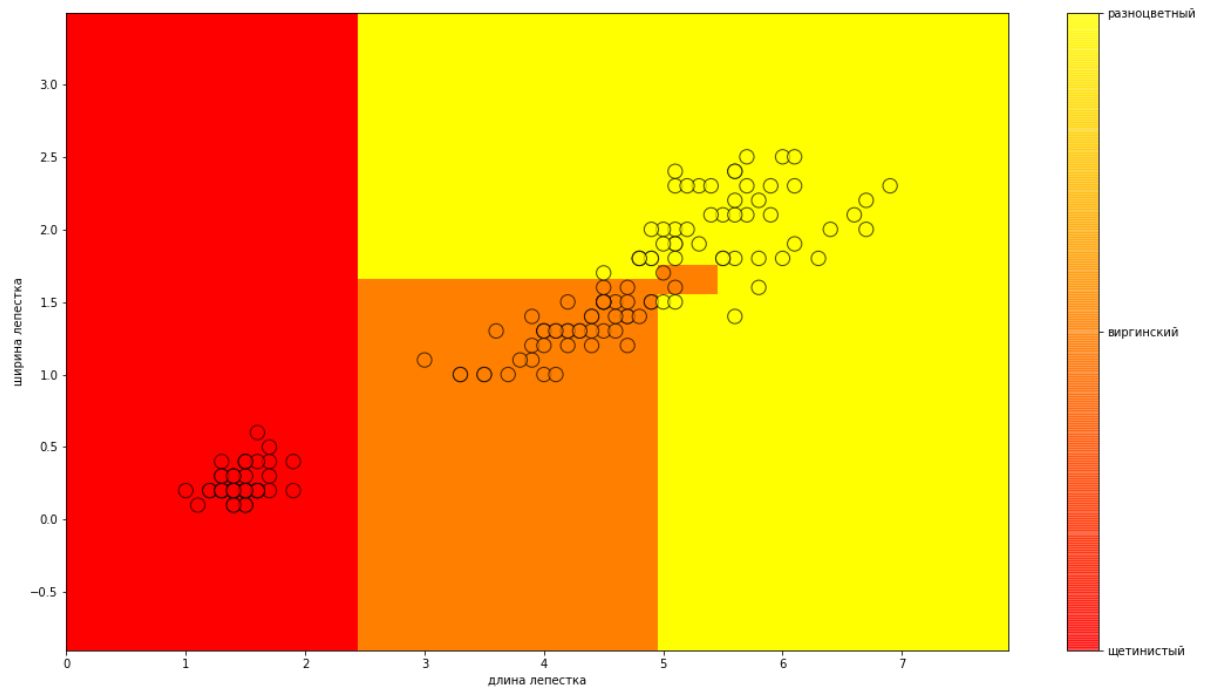
```
Out[15]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, l1_ratio=None, max_iter=100,
multi_class='auto', n_jobs=None, penalty='l2',
random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
warm_start=False)
```



Решающие деревья

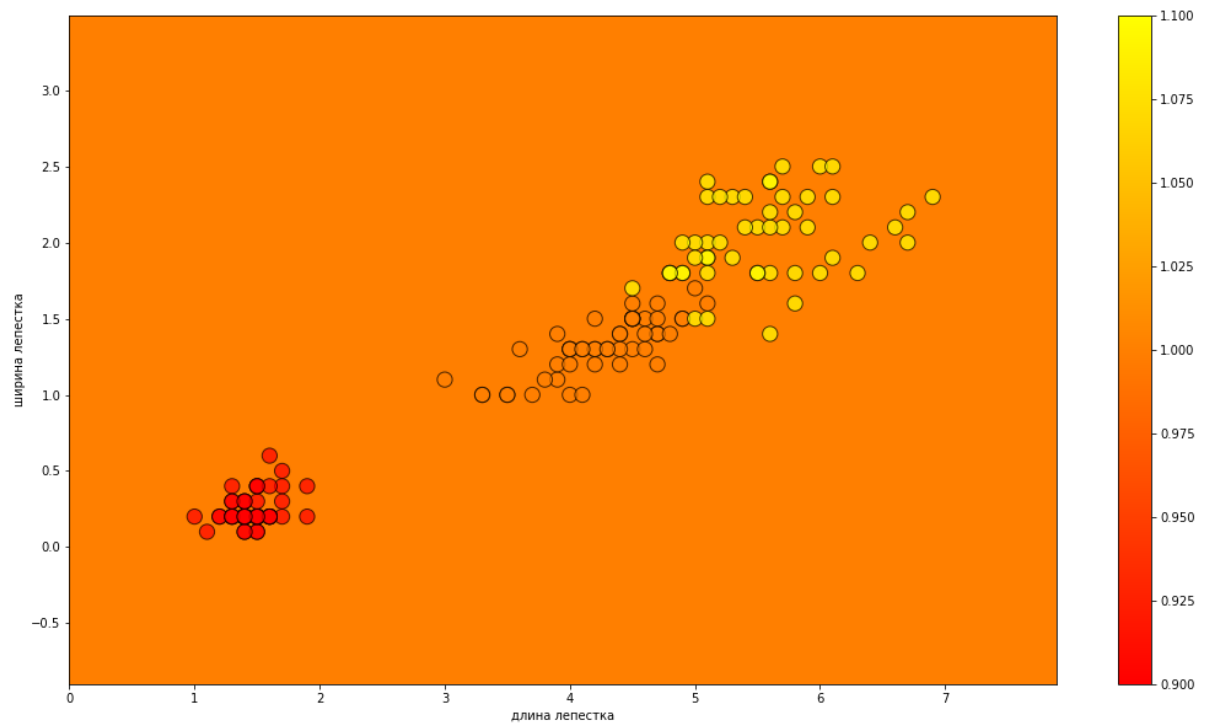
```
In [16]: from sklearn.tree import DecisionTreeClassifier
plt.figure(figsize=(18.0, 10.0))
plot_model(X, y, DecisionTreeClassifier())
```

```
Out[16]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                                max_depth=None, max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort='deprecated',
                                random_state=None, splitter='best')
```



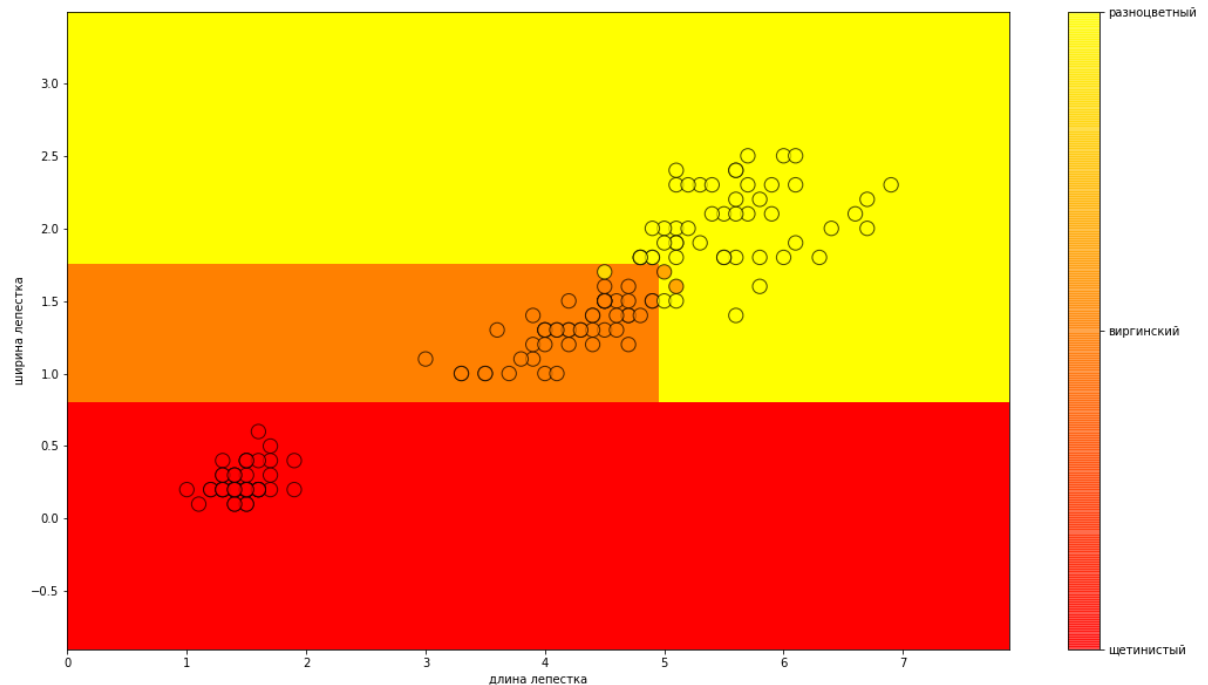
```
In [17]: plt.figure(figsize=(18.0, 10.0))  
plot_model(X, y, DecisionTreeClassifier(), proba=True)
```

```
Out[17]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',  
                                max_depth=None, max_features=None, max_leaf_nodes=None,  
                                min_impurity_decrease=0.0, min_impurity_split=None,  
                                min_samples_leaf=1, min_samples_split=2,  
                                min_weight_fraction_leaf=0.0, presort='deprecated',  
                                random_state=None, splitter='best')
```



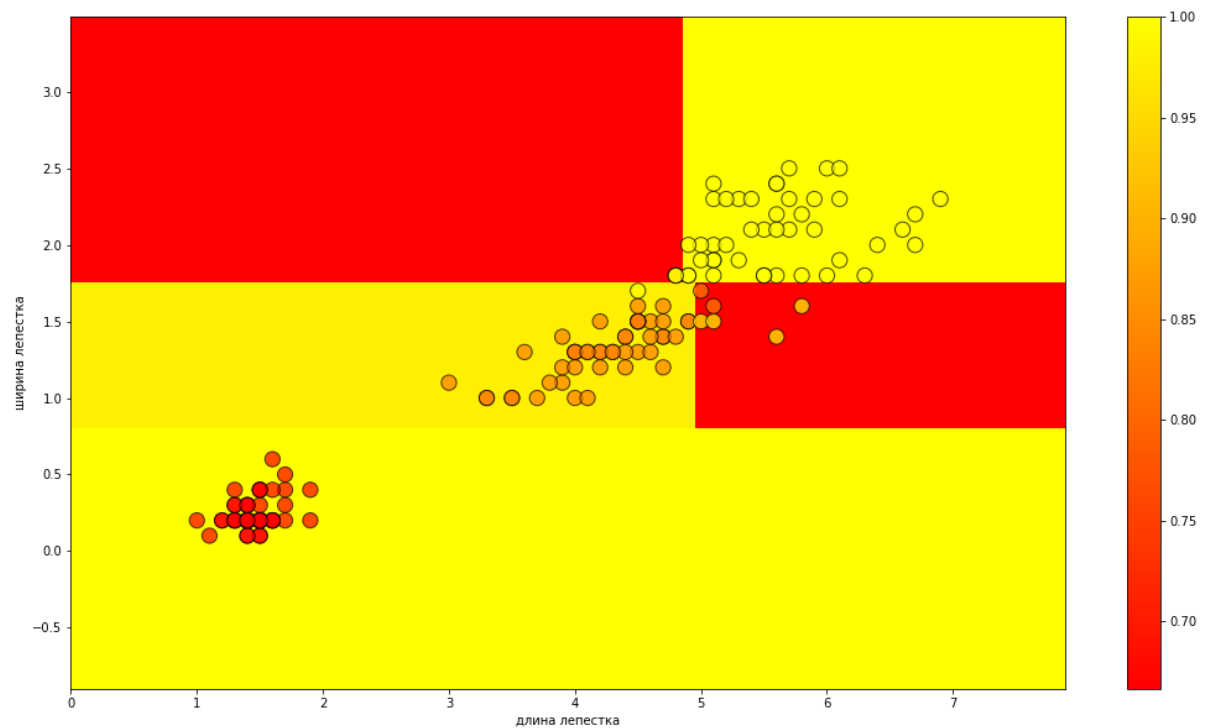
Вопрос 4: Уверенность модели везде близка к 1.0. Что произошло?

```
In [19]: plt.figure(figsize=(18.0, 10.0))
clf = plot_model(X, y, DecisionTreeClassifier(max_depth=3))
```



```
In [20]: plt.figure(figsize=(18.0, 10.0))
plot_model(X, y, DecisionTreeClassifier(max_depth=3), proba=True)
```

```
Out[20]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
max_depth=3, max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, presort='deprecated',
random_state=None, splitter='best')
```

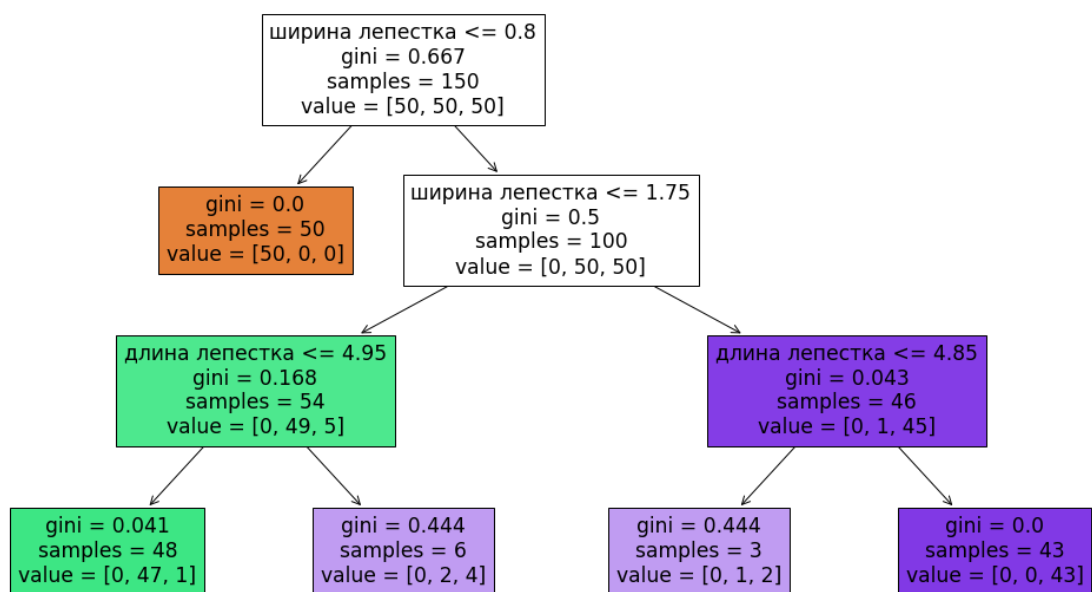


Визуализация структуры дерева

```
In [22]: from sklearn.tree import plot_tree

def draw_decision_tree(clf, column_names):
    plt.figure(figsize=(18,10))
    plot_tree(clf, filled=True, feature_names=column_names)
    plt.show()
```

```
In [23]: draw_decision_tree(clf, features)
```



Обучение решающего дерева. Алгоритм ID3

$v_0 := TreeGrowing(X^\ell)$

1: **ФУНКЦИЯ** $TreeGrowing(U \subset X^\ell) \mapsto$ корень дерева v

2: **если** $StopCriteria(U)$ **то**

3: **вернуть** новый лист v , взяв $y_v := Major(U)$

4: найти признак, наиболее выгодный для ветвления дерева:

$$f_v = \arg \max_{f \in F} Gain(f, U)$$

5: **если** $Gain(f_v, U) < G_0$ **то**

6: **вернуть** новый лист v , взяв $y_v := Major(U)$

7: создать новую внутреннюю вершину v с функцией f_v

8: **для всех** $k \in D_v$

$$U_k = \{x \in U : f_v(x) = k\}, S_v(k) := TreeGrowing(U_k)$$

9: **вернуть** v

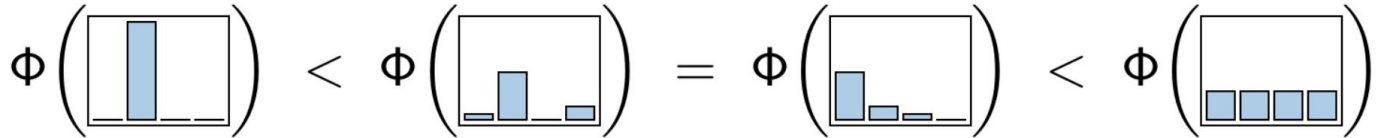
Мажоритарное правило: $Major(U) := \arg \max P(y|U)$

Мера неопределённости распределения

Частотная оценка вероятности класса y в вершине $v \in V_{\text{внутр}}$:

$$p_y \equiv P(y|U) = \frac{1}{|U|} \sum_{x_i \in U} [y_i = y]$$

$\Phi(U)$ — мера неопределённости (impurity) распределения p_y :



1. минимальна, когда $p_y \in \{0, 1\}$,
2. максимальна, когда $p_y = \frac{1}{|Y|}$ для всех $y \in Y$
3. симметрична: не зависит от перенумерации классов

$$\Phi(U) = E\mathcal{L}(p_y) = \sum_{y \in Y} p_y \mathcal{L}(p_y) = \frac{1}{|U|} \sum_{x_i \in U} \mathcal{L}(P(y_i|U)) \rightarrow \min,$$

где $\mathcal{L}(p)$ убывает и $\mathcal{L}(1) = 0$, например: $-\log p, 1 - p, 1 - p^2$

Критерий ветвления

Неопределённость распределений $P(y_i|U_k)$ после ветвления вершины v по признаку f и разбиения $U = \bigsqcup_{k \in D_v} U_k$:

$$\begin{aligned} \Phi(U_1, \dots, U_{|D_v|}) &= \frac{1}{|U|} \sum_{x_i \in U} \mathcal{L}(P(y_i|U_{f(x_i)})) = \\ &= \frac{1}{|U|} \sum_{k \in D_v} \sum_{x_i \in U_k} \mathcal{L}(P(y_i|U_k)) = \sum_{k \in D_v} \frac{|U_k|}{|U|} \Phi(U_k) \end{aligned}$$

Выигрыш от ветвления вершины v :

$$\text{Gain}(f, U) = \Phi(U) - \Phi(U_1, \dots, U_{|D_v|}) = \Phi(U) - \sum_{k \in D_v} \frac{|U_k|}{|U|} \Phi(U_k) \rightarrow \max_{f \in F}$$

Критерий Джини и энтропийный критерий

Два класса, $Y = \{0, 1\}$, $P(y|U) = \begin{cases} q, & y = 1 \\ 1 - q, & y = 0 \end{cases}$

- Если $\mathcal{L}(p) = -\log_2 p$, то

$\Phi(U) = -q \log_2 q - (1 - q) \log_2 (1 - q)$ — энтропия выборки

- Если $\mathcal{L}(p) = 2(1 - p)$, то

$\Phi(U) = 4q(1 - q)$ — неопределённость Джини (Gini impurity)

Обработка пропущенных значений

На стадии обучения

- $f_v(x_i)$ не определено $\Rightarrow x_i$ исключается из U для $\text{Gain}(f_v, U)$
- $q_{vk} = \frac{|U_k|}{|U|}$ — оценка вероятности k -й ветви, $v \in V_{\text{внутр}}$
- $P(y|x, v) = \frac{1}{|U|} \sum_{x_i \in U} [y_i = y]$ для всех $v \in V_{\text{лист}}$

На стадии классификации

- $f_v(x)$ определено \Rightarrow из дочерней $s = S_v(f_v(x))$ взять $P(y|x, v) = P(y|x, s)$
- $f_v(x)$ не определено \Rightarrow пропорциональное распределение:

$$P(y|x, v) = \sum_{k \in D_v} q_{vk} P(y|x, S_v(k))$$

- Окончательное решение — наиболее вероятный класс:

$$a(x) = \arg \max_{y \in Y} P(y|x, v_0)$$

Жадная нисходящая стратегия: достоинства и недостатки

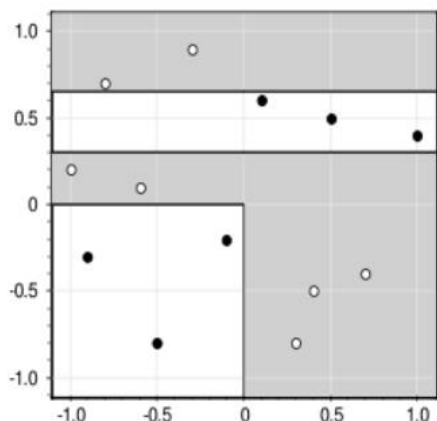
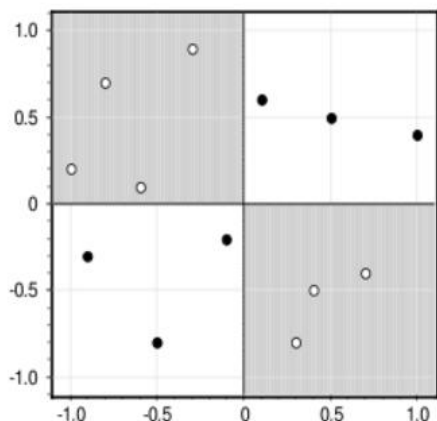
Достоинства:

- Интерпретируемость и простота классификации
- Гибкость: можно варьировать множество F
- Допустимы разнотипные данные и данные с пропусками
- Трудоёмкость линейна по длине выборки $O(|F|h\ell)$
- Не бывает отказов от классификации

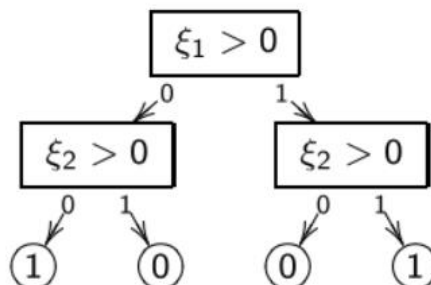
Недостатки:

- Жадная стратегия переусложняет структуру дерева, и, как следствие, сильно переобучается
- Фрагментация выборки: чем дальше v от корня, тем меньше статистическая надёжность выбора f_v, y_v
- Высокая чувствительность к шуму, к составу выборки, к критерию информативности

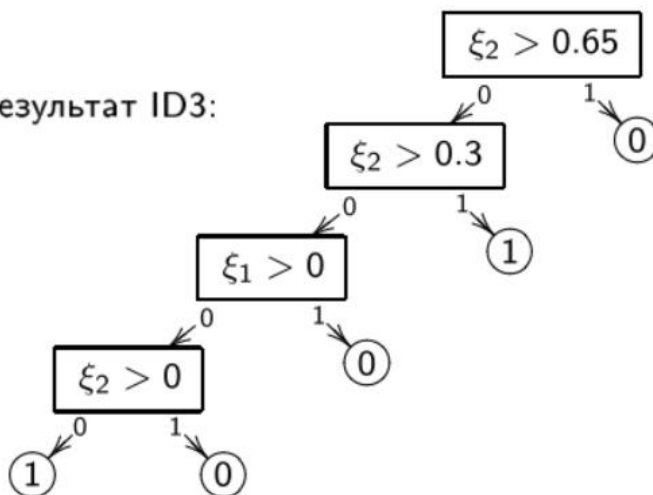
Жадная стратегия переусложняет структуру дерева



Оптимальное дерево для задачи XOR:



Результат ID3:



Усечение дерева (pruning)

X^q — независимая контрольная выборка, $q \approx 0.5\ell$

1: для всех $v \in V_{\text{внутр}}$

2: $X_v^q :=$ подмножество объектов X^q , дошедших до v

3: если $X_v^q = \emptyset$ то

4: вернуть новый лист $v, y_v := \text{Major}(U)$

5: число ошибок при классификации X_v^q разными способами:

$\text{Err}(v)$ — поддеревом, растущим из вершины v

$\text{Err}_k(v)$ — дочерним поддеревом $S_v(k), k \in D_v$

$\text{Err}_c(v)$ — классом $c \in Y$

6: в зависимости от того, какое из них минимально:

сохранить поддерево v

заменить поддерево v дочерним $S_v(k)$

заменить поддерево v листом, $y_v := \arg \min_{c \in Y} \text{Err}_c(v)$

CART: деревья регрессии и классификации

Обобщение на случай регрессии: $Y = \mathbb{R}, y_v \in \mathbb{R}$

$$C(a) = \sum_{i=1}^{\ell} (a(x_i) - y_i)^2 \rightarrow \min_a$$

Пусть U — множество объектов x_i , дошедших до вершины v

Мера неопределённости — среднеквадратичная ошибка

$$\Phi(U) = \min_{y \in Y} \frac{1}{|U|} \sum_{x_i \in U} (y - y_i)^2$$

Значение y_v в терминальной вершине v — МНК-решение:

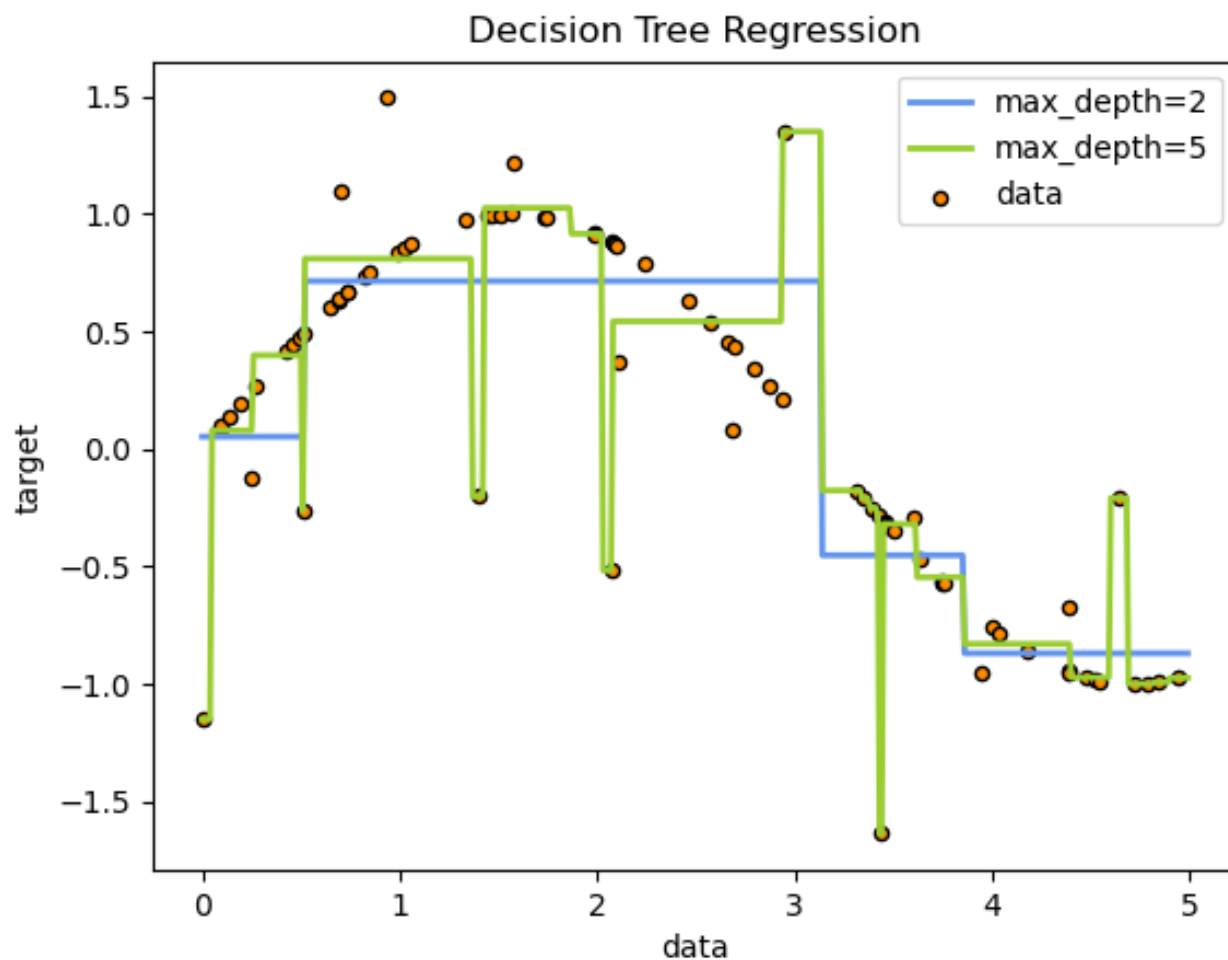
$$y_v = \frac{1}{|U|} \sum_{x_i \in U} y_i$$

Дерево регрессии $a(x)$ — это кусочно-постоянная функция.

Пример. Деревья регрессии различной глубины

Чем сложнее дерево (чем больше его глубина), тем выше влияние шумов в данных и выше риск переобучения.

https://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html (https://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html)



CART: критерий Minimal Cost-Complexity Pruning

Среднеквадратичная ошибка со штрафом за сложность дерева:

$$C_\alpha(a) = \sum_{i=1}^{\ell} (a(x_i) - y_i)^2 + \alpha |V_{\text{лист}}| \rightarrow \min_{\alpha}$$

При увеличении α дерево последовательно упрощается.

Причем последовательность вложенных деревьев единственна.

Из этой последовательности выбирается дерево с минимальной ошибкой на тестовой выборке (Hold-Out).

Для случая классификации используется аналогичная стратегия усечения, только с критерием Джини.

Вспомогательная задача бинаризации вещественного признака

Цель: сократить перебор предикатов вида $[\alpha \leq f(x) \leq \beta]$

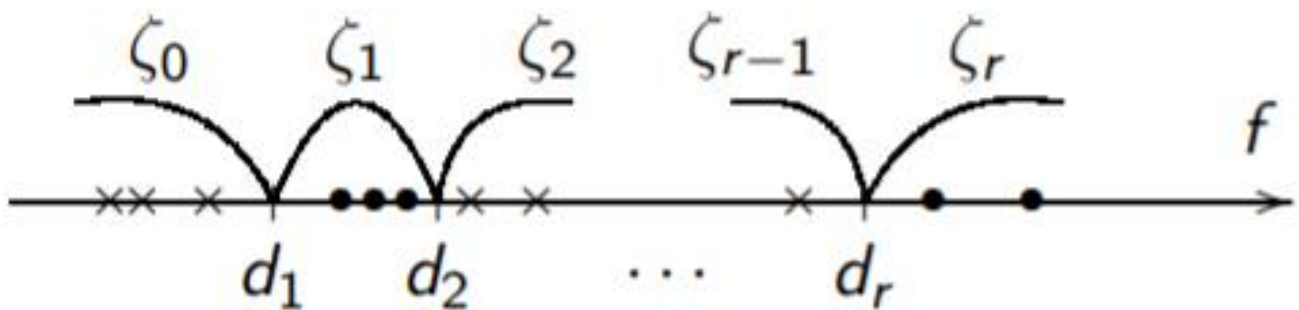
Дано: выборка значений вещественного признака $f(x_i), x_i \in X^l$

Найти: наилучшее (в каком-то смысле) разбиение области значений признака на относительно небольшое число зон:

$$\zeta_0(x) = [f(x) < d_1]$$

$$\zeta_s(x) = [d_s \leq f(x) < d_{s+1}], s = 1, \dots, r-1$$

$$\zeta_r(x) = [d_r \leq f(x)]$$



Способы разбиения области значений признака на зоны

- Жадная максимизация информативности путём слияний
- Разбиение на равномошные подвыборки
- Разбиение по равномерной сетке «удобных» значений
- Объединение нескольких разбиений — например, равномерное и медианное

Случайный лес (Random Forest)

Голосование деревьев классификации, $Y = \{-1, +1\}$

$$a(t) = \text{sign} \frac{1}{T} \sum_{t=1}^T b_t(x)$$

Голосование деревьев регрессии, $Y = \mathbb{R}$

$$a(t) = \text{sign} \frac{1}{T} \sum_{t=1}^T b_t(x)$$

- каждое дерево $b_t(x)$ обучается по случайной выборке с возвращениями
- в каждой вершине признак выбирается из случайного подмножества \sqrt{n} признаков ($\lfloor n/3 \rfloor$ для регрессии)
- признаки и пороги выбираются по критерию Джини
- усечений (pruning) нет

Резюме

- Логистическая регрессия — метод классификации, оценивающий условные вероятности классов $P(y|x)$
- Основные требования к логическим закономерностям:
 - интерпретируемость, информативность, различность
- Преимущества решающих деревьев:
 - интерпретируемость
 - допускаются разнотипные данные
 - возможность обхода пропусков
- Недостатки решающих деревьев:
 - переобучение
 - фрагментация
 - неустойчивость к шуму, составу выборки, критерию
- Способы устранения этих недостатков:
 - редукция
 - регуляризация
 - композиции (леса) деревьев