National Research Tomsk State University

Классификаторы

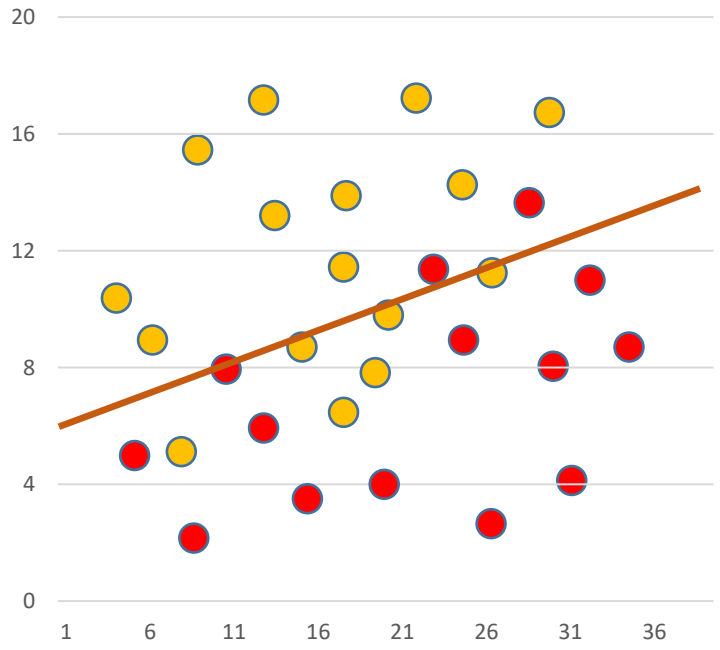
Сергей В. Аксёнов,

к.т.н., доцент кафедры теоретических основ информатики,

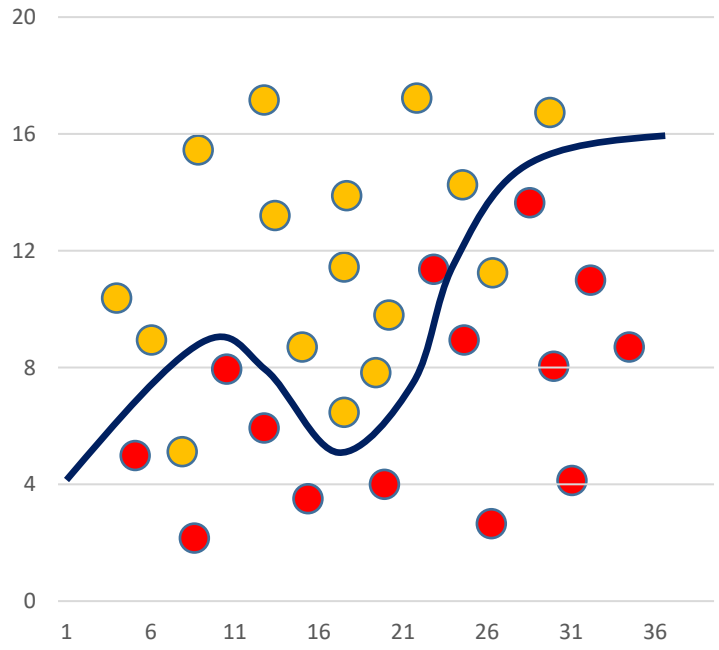
Томский государственный университет

Томск-2023

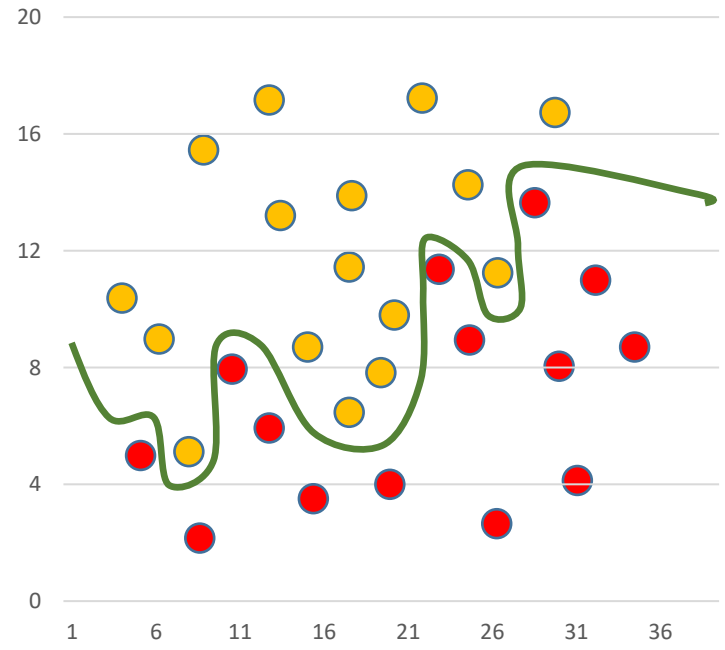
Плохое и хорошее обучение (Классификация)



Недообучение



Хорошее обучение



Переобучение

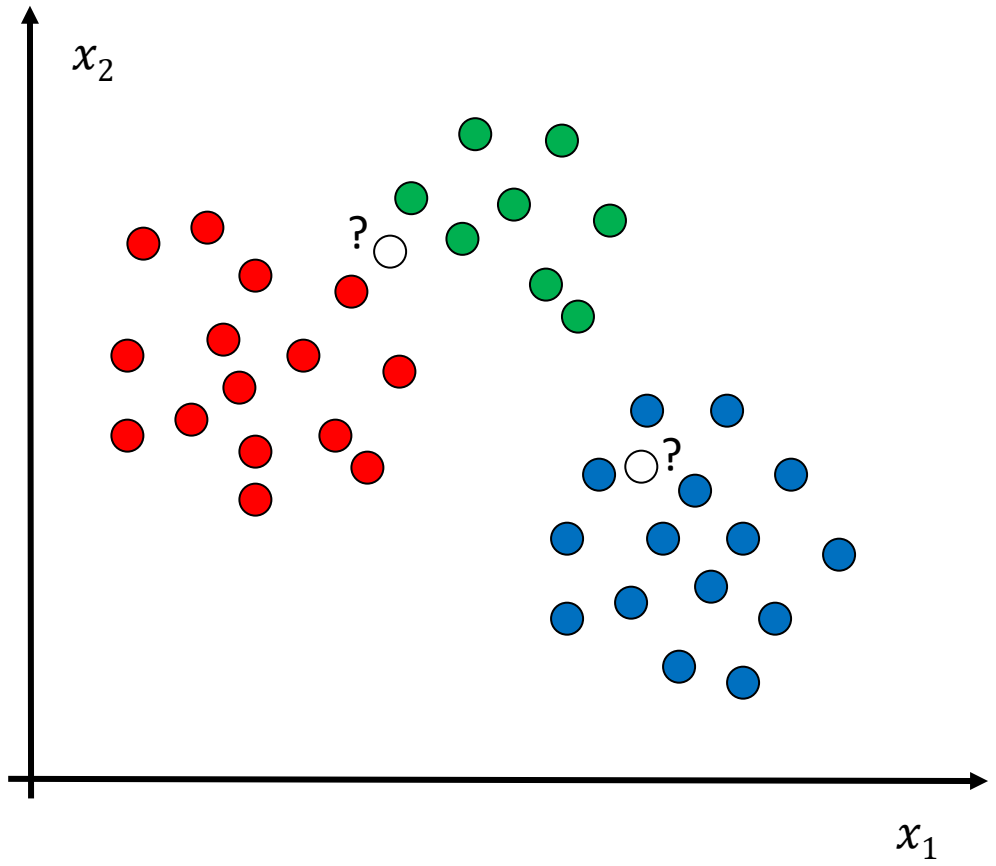
Класс A Класс B

Линейно разделимые и линейно неразделимые классы

К-ближайших соседей

Нахождение набора объектов, чьи признаки близки к тестирующему примеру.
К – число соседей = 1, 3, 5

Приведение признаков к одинаковой шкале.
Требование по хранению всей выборки.



Бинарный линейный классификатор

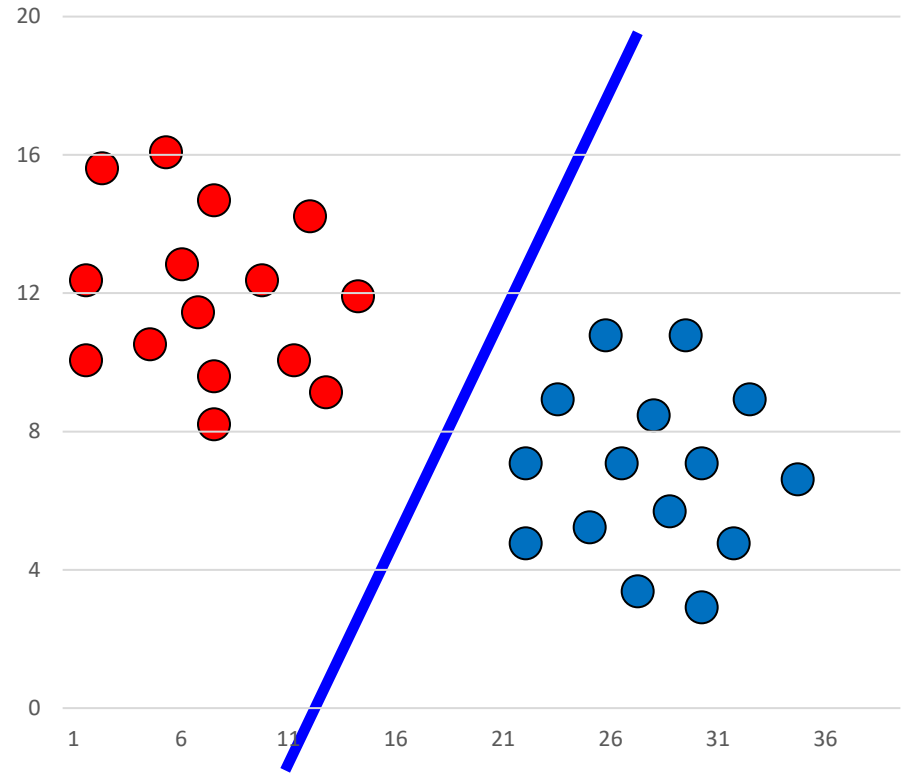
Результат обучения: входной вектор относится либо к положительному ($\hat{y}$=+1), либо отрицательному ($\hat{y}$=-1) классу

Вектор признаков:

$$x = (x_1, x_2, x_3, \dots, x_N)$$

Выход модели:

$$\hat{y} = \hat{y}(x, w) = sign\left(w_0 + \sum_i^N w_i x_i\right) = sign(w^T x)$$



Метод опорных векторов (Support Vector Machine)

Результат обучения: максимизация зазора (расстояния между разделяющей гиперплоскостью и самыми близкими к этой плоскости тренировочными образцами

«Положительная» гиперплоскость: $w^T x = +1$

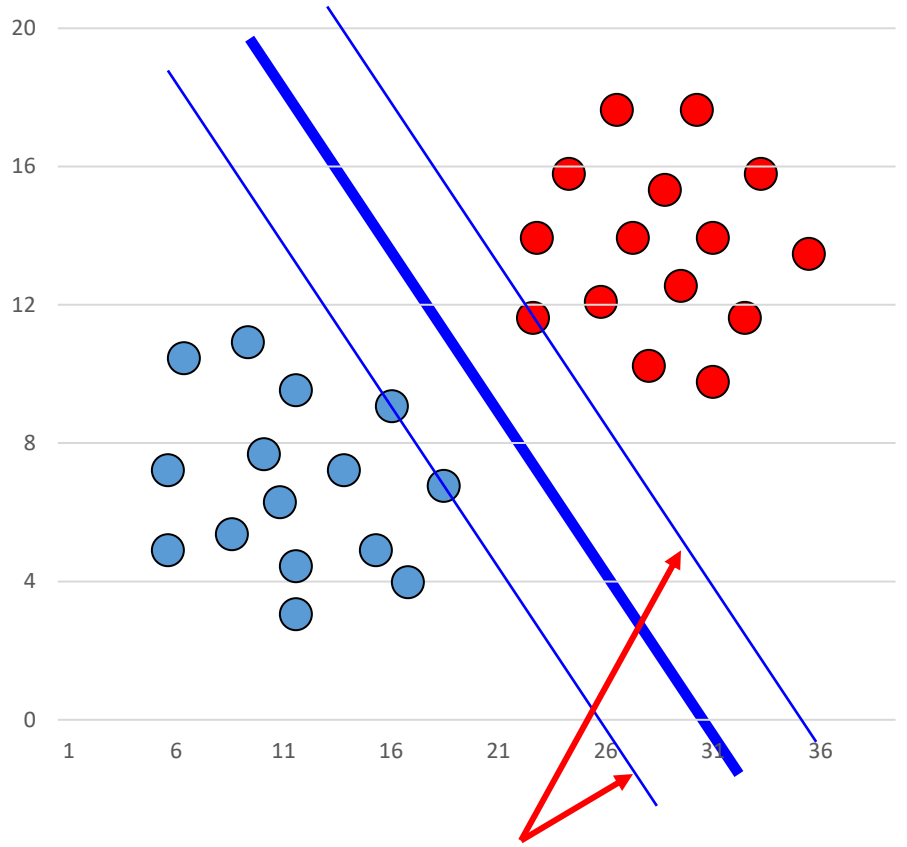
«Отрицательная» гиперплоскость: $w^T x = -1$

Граница решения: $w^T x = 0$

Целевая функция SVM: $\frac{2}{\|w\|} \rightarrow max$

При ограничениях: $w_0 + w^T x^i \geq +1, \text{если } y^i = 1$

$w_0 + w^T x^i < -1, \text{если } y^i = -1$

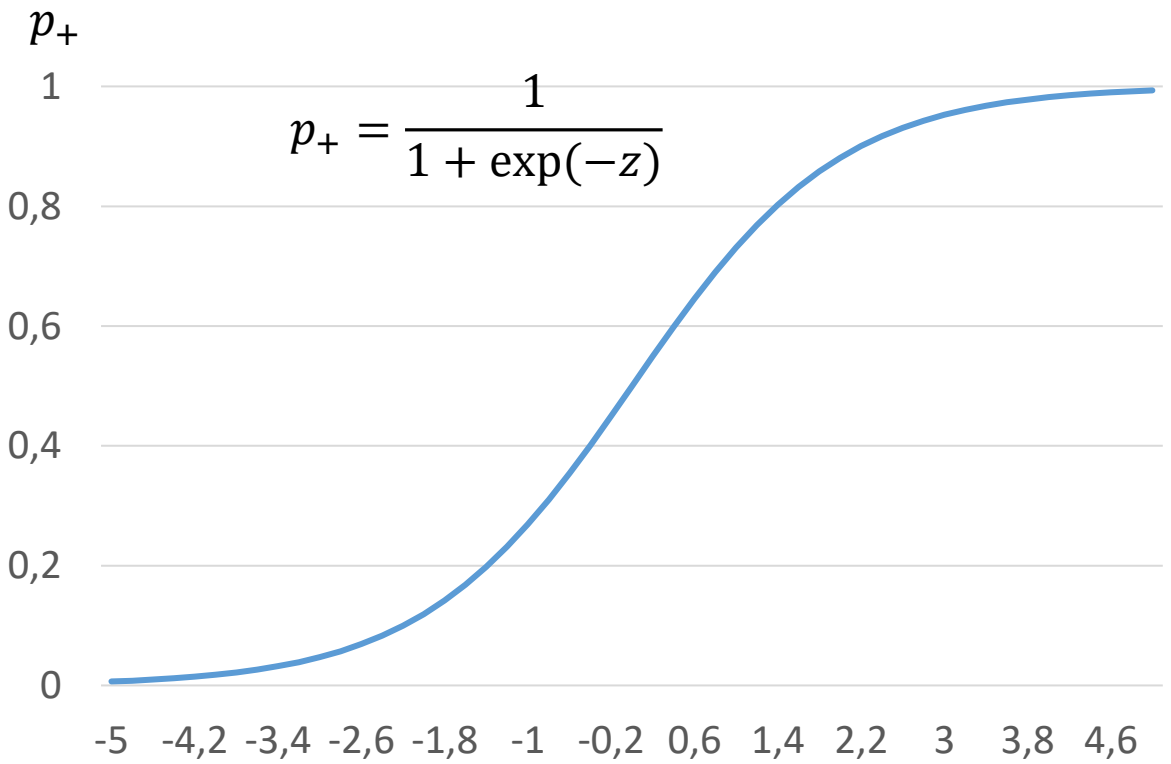


Опорные вектора

Логистическая регрессия

Прогнозируют вероятность p_+ отнесения примера x r к классу +1

$z=\sum_i^N w_i x_i$



Функция стоимости

$$J(w)=\sum_i \frac{1}{2}\left(\phi(z^{(i)})-y^{(i)}\right)^2$$

Функция правдоподобия

$$L(w)=P(y\,|\,x;w)=\prod_{i=1}^{n} P\left(y^{(i)}\,|\,x^{(i)};w\right)=\prod_{i=1}^{n}\left(\phi(z^{(i)})\right)^{y^{(i)}}\left(1-\phi(z^{(i)})\right)^{1-y^{(i)}}$$

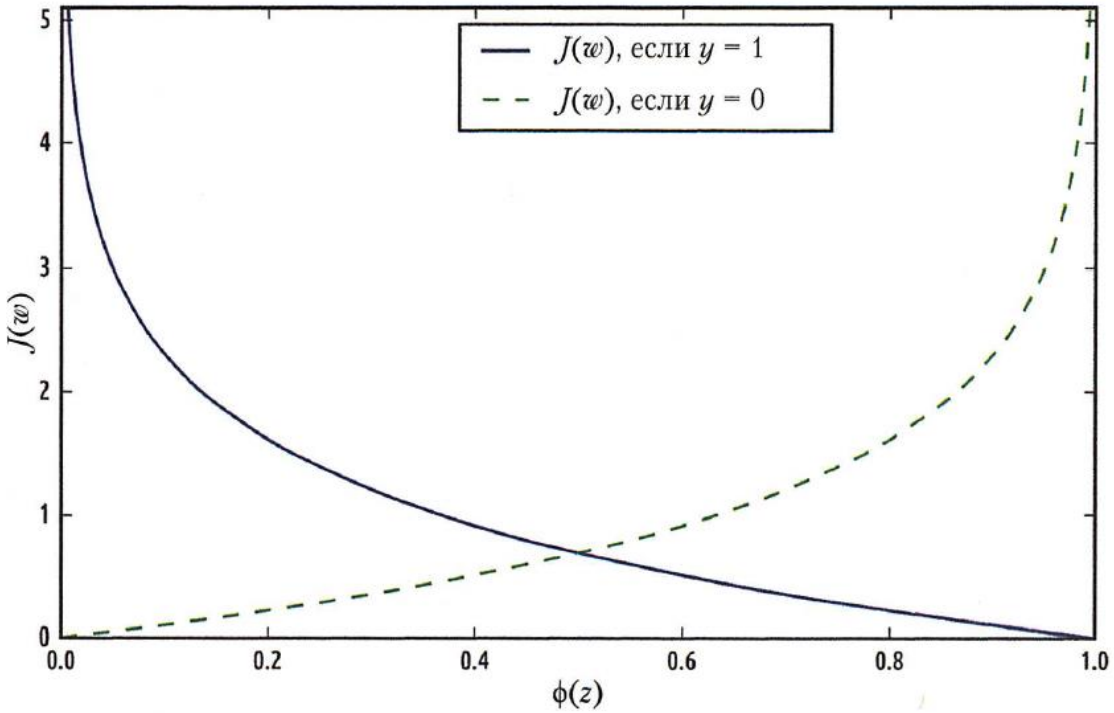
Логарифмическая функция правдоподобия

$$l(w)=\log L(w)=\sum_{i=1}^{n}\left[y^{(i)}\log\left(\phi(z^{(i)})\right)+\left(1-y^{(i)}\right)\log\left(1-\phi(z^{(i)})\right)\right]$$

Регуляризация в логистической регрессии

Функция, использующаяся для поиска параметров модели

$$J(w)=\sum_{i=1}^{n}\left[-y^{(i)}\log\left(\phi(z^{(i)})\right)-\left(1-y^{(i)}\right)\log\left(1-\phi(z^{(i)})\right)\right]$$



Регуляция переобучения выполняется при помощи регуляризации. Наложение штрафов на экстремальные значения параметров.

$$\frac{\lambda}{2}\|w\|^2=\frac{\lambda}{2}\sum_{j=1}^{m}w_j^2$$ L2 - регуляризация

Новая функция, учитывающая штрафы

$$J(w)=\sum_{i=1}^{n}\left[-y^{(i)}\log(\phi(z^{(i)}))-(1-y^{(i)})\log(1-\phi(z^{(i)}))\right]+\frac{\lambda}{2}\|w\|^2$$

Дерево решений

Разбиение данных на подмножества, приводящему к самому большому приросту информации (получению однородных регионов решения)

Функция прироста информации:

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

Меры неоднородности:

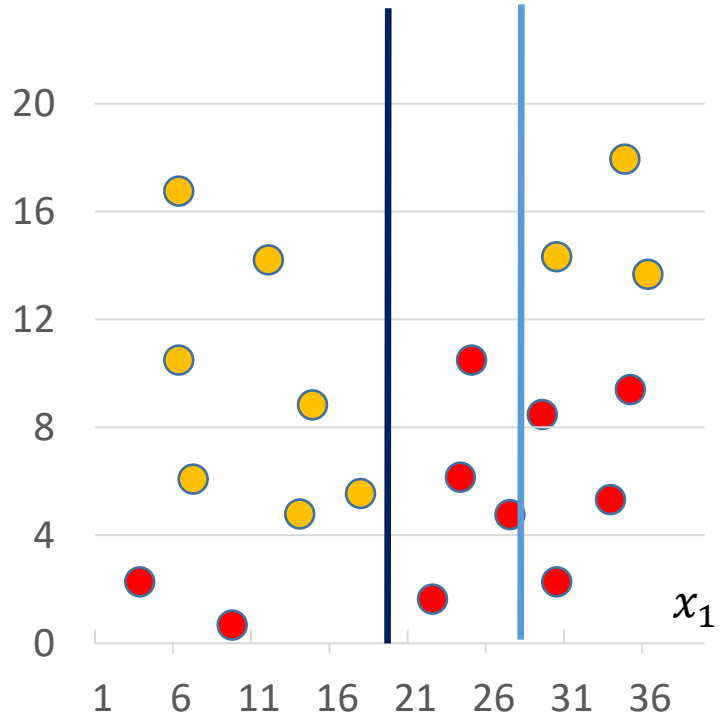
Энтропия: $I_G(t) = 1 - \sum_{i=1}^{c} p(i|t)^2$

Мера неопределенности Джини: $I_H(t) = -\sum_{i=1}^{c} p(i|t) log_2 p(i|t)$

Ошибка классификации: $I_E(t) = 1 - \max(p(i|t))$

$p(i|t)$ -доля образцов, принадлежащая классу i для узла t

Построение деревьев решений. Пример-1



$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

В качестве критерия взята ошибка классификации:

$$I_E(t) = 1 - \max(p(i|t))$$

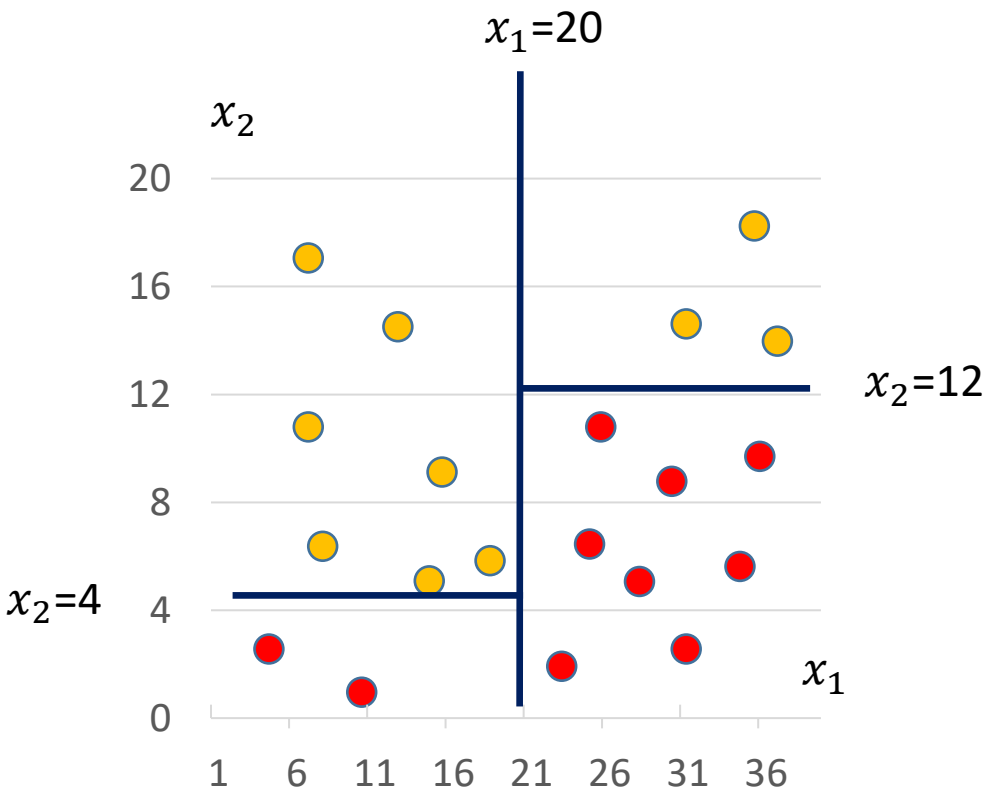
Неоднородность корневого узла:

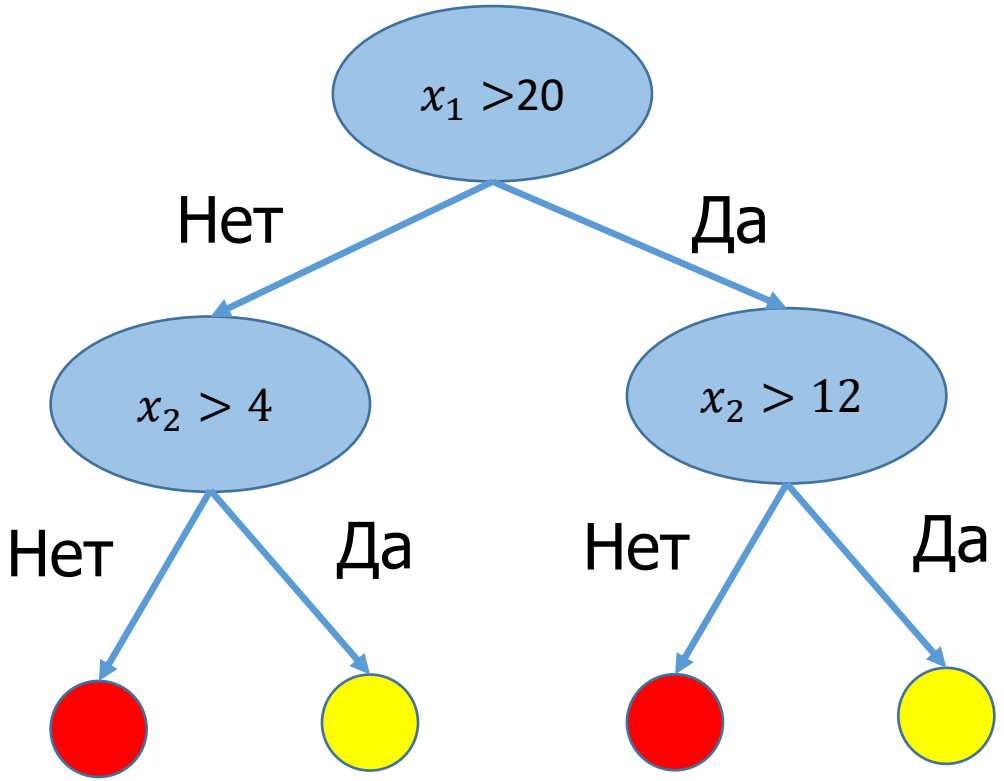
$$I(D_0) = 1 - \max\left(\frac{10}{20}, \frac{10}{20}\right) = 1 - 0.5 = 0.5$$

Для расщепления $x_1 = 20$: $IG(D_0, x_1 = 20) = 0.5 - \frac{9}{20}\left(1 - \frac{7}{9}\right) - \frac{11}{20}\left(1 - \frac{8}{11}\right) = 0.25$

Для расщепления $x_1 = 28$: $IG(D_0, x_1 = 28) = 0.5 - \frac{13}{20}\left(1 - \frac{7}{13}\right) - \frac{7}{20}\left(1 - \frac{4}{7}\right) = 0.05$

Построение деревьев решений. Пример-2





Примеры классов из Scikit-learn. Параметры

class sklearn.tree.DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False)

class sklearn.linear_model.LogisticRegression(penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None)

Хорошая модель должна использовать не параметры алгоритмов по умолчанию, а исследование результатов алгоритмов с разными параметрами!!!