In [1]:
```
import IPython.display
IPython.display.display_latex(IPython.display.Latex(filename="../macros.tex"))
```

Логические алгоритмы

Логическая закономерность в задачах классификации — легко интерпретируемое правило (rule), выделяющее из обучающей выборки достаточно много объектов какого-то одного класса и практически не выделяющее объекты остальных классов. Логические закономерности являются элементарными «строительными блоками» для широкого класса логических алгоритмов классификации, называемых также алгоритмами индукции правил (rule induction).

Дано:

$\hat{X} \in \mathbb{R}^M$ - пространство объектов Y - множество классов

$(X, Y)^N$ - тренировочная выборка

Говорят: $\phi : \hat{X} \to \{0, 1\}$ выделяет объект x если $\phi(x) = 1$

Закономерностью называется предикат $\phi(x)$, удовлетворяющий требованиям интерпретируемости и информативности.

Предикат $\phi(x)$ называется «хорошо интерпретируемым» или правилом, если он описывается простой формулой, понятной экспертам в данной прикладной области. Строгого формального определения интерпретируемости не существует.(например использует не больше k признаков).

Информативный предикат интуитивно это если выделяет максимально положительных объектов и минимально отрицательных.(ассоциииируем предикат с каким то классом \phi_{yt}(x))

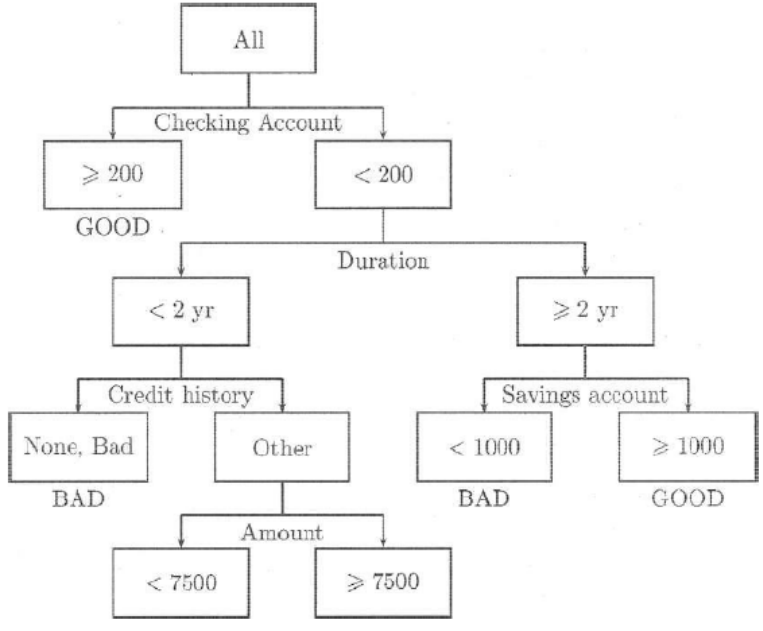
К набору закономерностей, образующих логический классификатор, дополнительно предъявляется требование взаимодополняемости (различности).

$$\alpha(x) = argmax_{y \in Y} \sum_{t=1}^{T_y} w_{yt} \phi_{yt}(x)$$

w_{yt} - неотрицательные веса.

Взаимодополняемостьозначает что для любого объекта должен быть ϕ_{yt} выделяющий этот объект.

Решающие деревья



1) $\forall v \in V_{\text{внутр}} \rightarrow$ предикат $\beta_v : \hat{X} \rightarrow \{0, 1\}$
2) $\forall v \in V_{\text{лист}} \rightarrow$ имя класса $c_v \in Y$

логические правила:

("checking account" < 200) AND ("Duration" < 2) AND ("credit history" = Bad) $\Rightarrow$ BAD

("checking account" < 200) AND ("Duration" $\geq$ 2) AND ("savings account" < 1000) $\Rightarrow$ BAD

как строить дерево

ПРОЦЕДУРА LearnID3 ($U \subseteq X^l$)

- если все объекты из U лежат в одном классе $c \in Y$
  - то **вернуть** новый лист $v, c_v = c$
- найти предикат с максимальной информативностью:

$$\beta = argmax_{\beta \in B} I(\beta, U)$$

- разбить выборку на две части $U = U_0 \cup U_1$ по предикату β:

$$U_0 = \{x \in U : \beta(x) = 0\}$$
$$U_1 = \{x \in U : \beta(x) = 1\}$$

- если $U_0 = \emptyset$ или $U_1 = \emptyset$
  - **вернуть** новый лист $v, c_v =$ Мажоритарный класс(U)
- создать новую внутреннюю вершину $v : \beta_v = \beta$
  построить левое поддерево: $L_v = LearnID3(U0)$
  построить правое поддерево: $R_v = LearnID3(U1)$
- **вернуть** v

Мы хотим получить "лучшее"(максимально информативное) разбиение на каждом шаге

Энтропия(Шенонна)

$$S = -\sum_{i=1}^{C} p_i log_2 p_i$$

p_i - вероятность принадлежать i-му классу(вероятности нахождения системы в i-ом состоянии)

В листе:

- класс 1: 9 объектов
- класс 2: 11 объектов

$p_1 = \frac{9}{20}$

$p_2 = \frac{11}{20}$

$$S_0 = -\frac{9}{20} log_2\left(\frac{9}{20}\right) - \frac{11}{20} log_2\left(\frac{11}{20}\right) \approx 1$$

$P = f_j > c_k$ -предикат

$P = true$:

- class 1: 8 objects
- class 2: 5 objects

$P = false$:

- class 1: 1 objects
- class 2: 6 objects

$$S_1 = -\frac{8}{13} log_2 \left(\frac{8}{13} \right) - \frac{5}{13} log_2 \left(\frac{5}{13} \right) \approx 0.96$$

$$S_2 = -\frac{1}{7} log_2 \left(\frac{1}{7} \right) - \frac{6}{7} log_2 \left(\frac{6}{7} \right) \approx 0.6$$

Прирост информации(information gain), IG:

$$IG(P) = S_0 - \sum_{i=1}^{q} \frac{C_i}{C} S_i$$

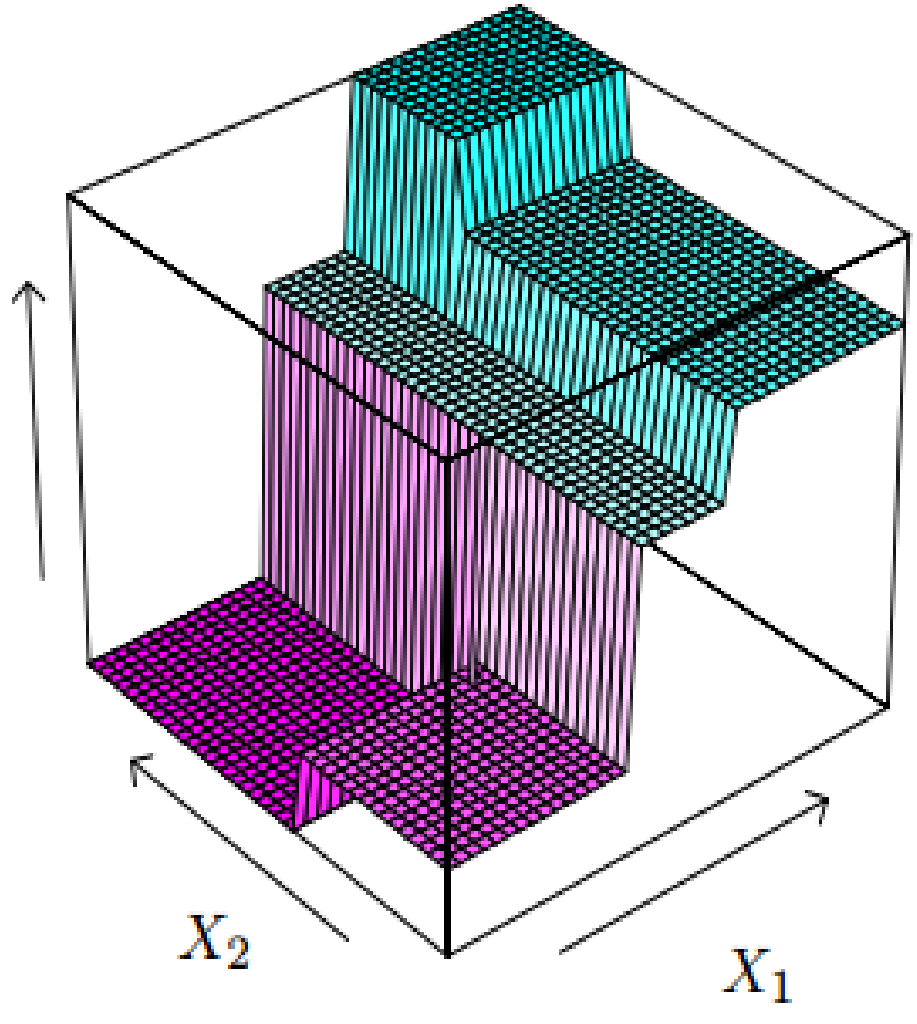
C_i - количество объектов в i-ом листе

$$IG(f_j > c_k) = S_0 - \frac{13}{20} * S_1 - \frac{7}{20} * S_2 \approx 0.16$$

выбор предиката в дереве:

$$argmax_P(IG(P))$$

$$argmax_{j,k}(IG(f_j > k))$$


X_2
X_1

другие реализации

Неопределенность Джини(Gini impurity)

$$G = 1 - \sum_k (p_k)^2$$

Критерий разделения донсокого

$$D(P) = \left| \{ (x_i, y_i) \ : \ y_i \neq y_j \ AND \ P(x_i) \neq P(x_j) \} \right|$$

Числовые признаки:

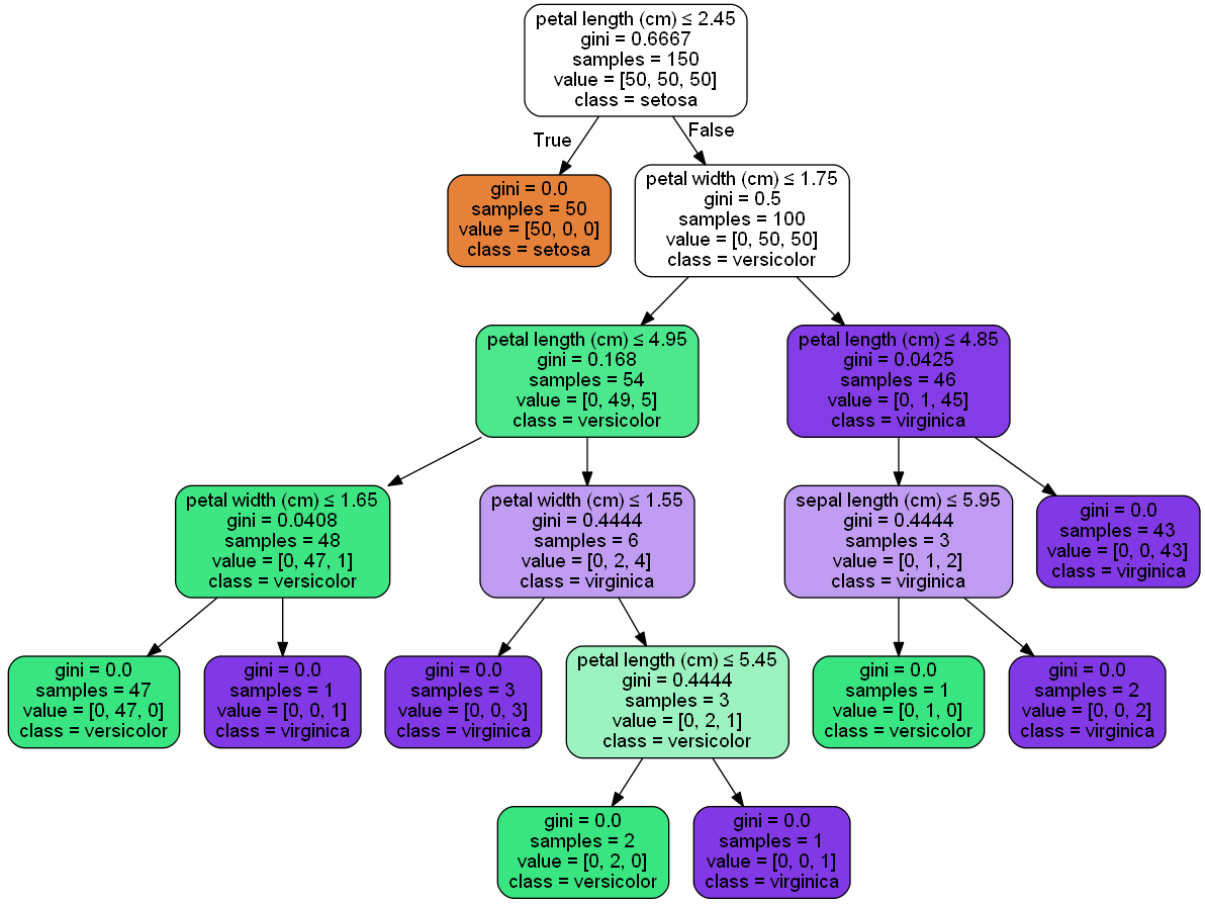
- сортируем значения $\rightarrow$ получаем сегменты
- выбираем число из сегмента

object	f1
1	22
2	36
3	19
4	36
5	54

$f1 : 19, 22, 36, 54 \Rightarrow c = \{0, 20, 30, 40, 60\}$

In [1]:
```
from sklearn.datasets import load_iris
from sklearn import tree
import graphviz
iris = load_iris()
clf = tree.DecisionTreeClassifier()
clf = clf.fit(iris.data, iris.target)
dot_data = tree.export_graphviz(clf, out_file=None,
                         feature_names=iris.feature_names,
                         class_names=iris.target_names,
                         filled=True, rounded=True,
                         special_characters=True)
graph = graphviz.Source(dot_data, format='png')
graph.render("images/iris")
```

Out[1]:
```
'images/iris.png'
```


petal length (cm) ≤ 2.45
gini = 0.6667
samples = 150
value = [50, 50, 50]
class = setosa
True
False
gini = 0.0
samples = 50
value = [50, 0, 0]
class = setosa
petal width (cm) ≤ 1.75
gini = 0.5
samples = 100
value = [0, 50, 50]
class = versicolor
petal length (cm) ≤ 4.95
gini = 0.168
samples = 54
value = [0, 49, 5]
class = versicolor
petal length (cm) ≤ 4.85
gini = 0.0425
samples = 46
value = [0, 1, 45]
class = virginica
petal width (cm) ≤ 1.65
gini = 0.0408
samples = 48
value = [0, 47, 1]
class = versicolor
petal width (cm) ≤ 1.55
gini = 0.4444
samples = 6
value = [0, 2, 4]
class = virginica
sepal length (cm) ≤ 5.95
gini = 0.4444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 43
value = [0, 0, 43]
class = virginica
gini = 0.0
samples = 47
value = [0, 47, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
gini = 0.0
samples = 3
value = [0, 0, 3]
class = virginica
petal length (cm) ≤ 5.45
gini = 0.4444
samples = 3
value = [0, 2, 1]
class = versicolor
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica

Регуляризация

- задаем глубину
- задаем минимальное количество элментов в листе

Прунинг - вначале строим полное дерево, потом "подрезаем" до оптимального с точки зрения достижения максимальной обучающей способности.

sklearn.tree.DecisionTreeClassifier main params:

- max_depth
- min_samples_leaf
- max_features

Важность признака x_i для предсказания y можно считать:

- по доле наблюдений, охватываемых правилом с участием x_i в дереве
- по тому, насколько правила с x_i улучшили разделимость классов с учетом доли охвата правил с участием x

на сколько часто признак встречался в дереве, как высоко он находился.

Деревья

- интерпретируемы
- просты
- встроенный отбор признаков
- быстрая классификация
- работает одновременно с дискретными и непрерывными признаками
- прогноз инвариантен к монотонным преобразованиям признаков (не нужна нормализация)
- разнообразные данные и пропуски не проблема

- дают возможность оценить важность признаков

- проблемы с шумами

- переобучаются
- если разделяющая кривая не параллельна осям координат, то может потребоваться много вершин
- для новых наблюдений требуется полная перестройка всего дерева