# Решающие деревья

Важный класс методов машинного обучения, которые базируются на решающих деревьях, предназначен в основном для решения задач классификации, но, тем не менее, есть возможность приспособления для решения задач регрессии.

Но первоначально решающие деревья были придуманы как попытка формализовать тот способ мышления, который используют люди при принятии решений. Например, это хорошо иллюстрируется логикой работы врача, когда он говорит с пациентом и задает один за другим уточняющие вопросы, и буквально за 4–5 вопросов он, имея ответы пациента, либо ставит диагноз, либо дает ему какие-то советы. Но если вообразить себе всю ту информацию, которую использует врач при таких опросах, то это не 3–4 пункта, а огромное разветвленное дерево, и каждый ответ пациента отправляет врача в очередную веточку этого дерева. Таким образом, рассматривая диалог врача с пациентом, мы имеем только один путь от корня дерева к листовой вершине. Такая аналогия всей информации, которой пользуется врач, и то, как можно было бы автоматически принимать решения, приводит к конструкции решающих деревьев.

В общем случае задача обучения выглядит так: у нас есть обучающая выборка, объекты, заданные $n$ признаками, и каждому объекту соответствует какой-то правильный ответ. Наша задача — построить алгоритм классификации, который был бы способен классифицировать новые объекты. При этом мы этот алгоритм классификации будем строить в виде дерева, то есть в виде какой-то вот последовательности принимаемых решений.

Начнем с определения бинарного решающего дерева:

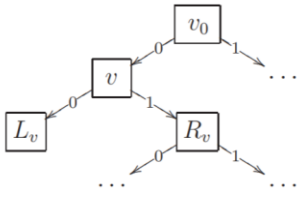
**Опр.** Бинарное решающее дерево — это алгоритм классификации, задающийся бинарным деревом, в котором:

1) каждой внутренней вершине $v \in V$ приписан предикат $\beta_v : X \to \{0,1\}$,
2) каждой терминальной вершине $v \in V$ приписано имя класса $c_v \in Y$.

При классификации объекта $x \in X$ мы проходим по дереву путь от корня до некоторого листа, в соответствии с алгоритмом:

**Алгоритм** Классификация объекта $x \in X$ бинарным решающим деревом

```
1: v := v_0;
2: пока вершина v внутренняя
3:     если β_v(x) = 1 то
4:         v := R_v;   (переход вправо)
5:     иначе
6:         v := L_v;   (переход влево)
7: вернуть c_v.
```



Имеется объект $x$ и дерево, следует пропускать этот объект через имеющееся дерево, начиная с корня. И в каждой внутренней вершине проверяется значение предиката на данном объекте. Если это значение равно 1, то объект переходит в правую дочернюю вершину, а в противном случае он переходит в левую дочернюю вершину. Так происходит до тех пор, пока объект не упрется в какую-то листовую вершину, в которой сидит метка класса, и тогда решающее дерево относит этот объект к этому классу.

Решающее дерево делит все пространство на непересекающиеся области. Логические закономерности часто характеризуются тем, что каждое правило выделяет какую-то область в пространстве объектов, в которой находятся объекты либо только одного какого-то класса, либо преимущественно объекты одного класса. То есть еще один способ представлять решающее дерево – это представить его в виде покрывающего набора конъюнкций. Почему это конъюнкция? Потому что когда объект спускается от корня дерева к листу, все внутренние вершины, которые он прошел, то можно взять предикаты, находящиеся в этих внутренних вершинах, из них образовать конъюнкцию, и это будет то самое правило.

**Задание на лабораторную работу**

Данная работа носит творческий характер и призвана показать, насколько студент подготовлен к реальному применению полученных знаний на практике. Как известно, в реальной работе никаких вводных данных не предоставляется, тем не менее, мы слегка пренебрегли данным правилом и предоставили теорию и предпочтительный метод для применения.

В приложенном файле (heart_data.csv) располагаются реальные данные по сердечной заболеваемости, собранные различными медицинскими учреждениями. Каждый человек представлен 13-ю характеристиками и полем goal, которое показывает наличие болезни сердца, поле принимает значение 0 или 1 (0 – нет болезни, 1 - есть). Символ ‘?’ в каком-либо поле означает, что для конкретного человека отсутствуют данные в этом поле (либо не производились замеры, либо не записывались в базу).

Требуется имеющиеся данные разбить на обучающую и тестовую выборки в процентном соотношении 70 к 30. После чего по обучающей выборке необходимо построить решающее дерево. Для построения дерева можно пользоваться любыми существующими средствами. Кроме того, для построения дерева необходимо будет решить задачу выделения информативных решающих правил относительно имеющихся числовых признаков.

Разрешается использовать уже реализованные решающие деревья из известных библиотек (например, scikit-learn для Python), либо реализовывать алгоритм построения дерева самостоятельно (все необходимые алгоритмы представлены в теории по ссылке).

В качестве результата работы необходимо сделать не менее 10 случайных разбиений исходных данных на обучающую и тестовую выборки, для каждой построить дерево и протестировать, после чего построить таблицу, в которой указать процент правильно классифицированных данных. Полученную таблицу необходимо включить в отчёт по лабораторной работе.

В отчёте следует отразить следующие изменяемые параметры: глубина дерева и количество деревьев для каждого тестируемого случая.

## Реализация в Scikit-Learn

В библиотеке scikit-learn решающие деревья реализованы в классах sklearn.tree.DecisionTreeClassifier (для классификации) и sklearn.tree.DecisionTreeRegressor (для регрессии). Обучение модели производится с помощью функции fit.

Пример использования:

```
import numpy as np
from sklearn.tree import DecisionTreeClassifier
X = np.array([[1, 2], [3, 4], [5, 6]])
y = np.array([0, 1, 0])
clf = DecisionTreeClassifier()
clf.fit(X, y)
```

## Материалы

Подробнее о деревьях и их построении

Подробнее про решающие деревья в sklearn