# Алгоритм классификации «Дерево решений»



Отправив своего ассистента в очередную экспедицию в Африку для исследования поведения шимпанзе, профессор Буковски осознал, что без помощника ему трудно справляться со своими обязанностями. И он решил подыскать себе ассистентку. Профессор нашел на сайте по трудоустройству несколько заинтересовавших его резюме. Для каждой кандидатки он выделил следующие параметры: коэффициент IQ, количество научных публикаций, наличие высшего образования, соотношение рост/вес. Эти данные профессор Буковски занес в таблицу, которая приведена ниже.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № кандидатки | Коэффициент IQ | Научных публикаций | Высшее образование | Соотношение рост/вес | Подходит? |
| шкала | числовая | категориальная | категориальная | числовая |  |
| 1 | 110 | 1 | да | 2.8 | 2 |
| 2 | 95 | 2 | да | 2.2 | 2 |
| 3 | 135 | 1 | да | 2.9 | 2 |
| 4 | 115 | 1 | нет | 2.0 | 2 |
| 5 | 100 | 2 | нет | 2.9 | 1 |
| 6 | 90 | 1 | нет | 3.5 | 2 |
| 7 | 75 | 1 | да | 3.1 | 2 |
| 8 | 85 | 2 | да | 3.1 | 1 |
| 9 | 65 | 0 | да | 2.1 | 1 |
| 10 | 70 | 1 | нет | 3.0 | 1 |

В столбце «Подходит» значение 2 означает, что ассистентка может быть принята на работу, 1 – не может быть принята.

Основной скрипт, с которого начитается работа – run.py. В нем задается исходное обучающее множество, которое приведено в таблице выше. Функция decision\_tree запускает рекурсивную процедуру построения дерева решений на основании алгоритма ID3. В нее передается обучающее множество (переменные X и Y), тип шкалы по каждому признаку (числовая - 0, категориальная – 1), исходный уровень дерева (изначально ноль – начинаем с нулевого корневого узла). Далее в скрипте run.py приведен код запуска классификатора, реализующего логику дерева решений, а в конце скрипта код (пока закомментированный), который нужен для классификации себя на роль ассистентки профессора ☺.

Скрипт decision\_tree.py реализует функционал построения дерева решений. Скрипт classify.py – будет содержать код классификатора, реализованного на основании дерева решений.

## Порядок выполнения работы

Открыть для редактирования скрипт decision\_tree.py. Задать в нем условие выхода из рекурсии. Условием выхода является то, что все примеры множества X принадлежат одному классу, то есть, вектор Y будет содержать только одно уникальное значение. Для определения количества уникальных значений воспользуйтесь конструкцией len(np.unique(Y)).

Найдите в этом же файле функцию Info (ее прототип: def Info(set)) и реализуйте код вычисления информационной энтропии множества T. Результат запишите в переменную info, которая будет возвращаться из функции. Для вычисления информационной энтропии используйте формулу:

Здесь T – множество, для которого считается энтропия (оно передается в функцию как параметр). P(i) – вероятность i-го класса в множестве T. Например, если множество T = [A A B B B], то p(A) = 2/5, а p(B) = 3/5. При вычислении энтропии следует принять, что логарифм от нуля равен нулю.

Найти в функции decision\_tree код вычисления информационного выигрыша для i-го категориального признака. Воспользоваться для расчета выигрыша формулой:

Значение уже посчитано и сохранено в переменной info. Вам необходимо вычислить , а затем и . Формула для вычисления :

Здесь n – количество подмножеств, на которое разбивается исходное множество. Например, множество T = [A A B C C C] состоит из трех подмножеств. – количество элементов для i-го подмножества. Например, для A оно равно 2, для C – 3. – количество элементов в множестве T (в данном примере - 6). – энтропия i-го подмножества, считается по формуле, приведенной выше. Обратите внимание, что энтропия считается по столбцу Y.

На этом реализация функции decision\_tree завершена. Нужно запустить скрипт в файле run.py. Он должен вывести на экран дерево, построенное по заданной обучающей выборке. Кроме того, он выведет строчку classification fail... :(, которая означает, что классификатор на основе дерева решений пока не работает.

Чтобы заставить его работать, откройте для редактирования файл classify.py. С помощью обычных условий if-elif-else запишите логику полученного дерева решений в этом файле.

Снова запустите run.py. Если все сделано правильно, вы увидите строку classification success!, которая означает, что описанный вами классификатор точно классифицирует все примеры из обучающей выборки (что естественно, ведь дерево по ней и строилось).

В заключение, раскомментируйте строчки в нижней части скрипта run.py, запустите скрипт и проверьте себя на роль ассистентки профессора Буковски ☺. Продемонстрируйте результаты.