Домашнее задание №3: «Одеревенеть от страха»

Дедлайн 1 (20 баллов): 23 марта, 23:59
Дедлайн 2 (10 баллов): 30 марта, 23:59

Домашнее задание нужно написать на Python и сдать в виде одного файла. Правило именования файла: `name_surname_3.[py | ipnb]`. Например, если вас зовут Иван Петров, то имя файла должно быть: `ivan_petrov_3.py` или `ivan_petrov_3.ipnb`.

До Хэллоуина осталось всего полгода, самое время научиться отличать чудищ друг от друга. По ссылке [1] находится датасет, содержащий информацию, которая поможет нам научиться отличать призраков, гоблинов и гулей друг от друга. Значения колонок указаны в заголовке файла, в качестве меток классов будет использоваться последняя колонка.

1 Реализуйте класс `Node` для хранения узла в дереве принятия решений. Класс должен хранить ссылки на свои поддеревья в переменных `false_branch` и `true_branch`, а также предикат по которому происходит деление на поддеревья. Hint: предикат удобно хранить в виде номера признака, по которому происходит деление выборки, и его значения.

2 В качестве критерия информативности в этой задаче мы будем использовать энтропийный критерий. Реализуйте функцию `entropy`, вычисляющую информационный выигрыш для некоторого подмножества объектов.

3 Реализуйте рекурсивный алгоритм построения дерева решения в виде класса `DecisionTree`. Структура класса приведена ниже:

```
class DecisionTree:
    def build(self, X, y, score=entropy):
        # рекурсивный алгоритм построения дерева
        return self

    def predict(self, x):
        ...
```

[1] `https://gist.github.com/ktisha/c2d540df52be497c89ceaf27169b2bab`

Метод `build` должен:

- Оценить информативность по всем возможным признакам с помощью функции `score`
- Выбрать наилучшее с точки зрения информативности разбиение
- Для наилучшего разбиения рекурсивно построить правое и левое поддеревья.

4 Реализуйте метод `predict`, принимающий объект `x` и возвращающий метку класса.

5 Для визуализации понадобится библиотека pillow [2]. Реализуйте методы `getwidth` и `getdepth`. Дополните функцию `drawnode` для визуализации дерева.

```
def drawtree(tree, path='tree.jpg'):
    w = getwidth(tree) * 100
    h = getdepth(tree) * 100

    img = Image.new('RGB', (w, h), (255, 255, 255))
    draw = ImageDraw.Draw(img)

    drawnode(draw, tree, w / 2, 20)
    img.save(path, 'JPEG')

def drawnode(draw, tree, x, y):
    if isinstance(tree, Node):
        shift = 100
        width1 = getwidth(tree.false_branch) * shift
        width2 = getwidth(tree.true_branch) * shift
        left = x - (width1 + width2) / 2
        right = x + (width1 + width2) / 2

        # получите текстовое представление предиката для текущего узла
        predicate = ...

        draw.text((x - 20, y - 10), predicate, (0, 0, 0))
        draw.line((x, y, left + width1 / 2, y + shift), fill=(255, 0, 0))
        draw.line((x, y, right - width2 / 2, y + shift), fill=(255, 0, 0))
        drawnode(draw, tree.false_branch, left + width1 / 2, y + shift)
        drawnode(draw, tree.true_branch, right - width2 / 2, y + shift)
    else:
        draw.text((x - 20, y), tree, (0, 0, 0))
```

6 Какие предикаты влияют на классификацию объекта как класс ”Goblin”?

[2] http://pillow.readthedocs.io/en/3.1.x/reference/ImageDraw.html