Лабораторная работа 5

Работа с датасетом

Фрагмент кода 5.1 демонстрирует, как может быть организовано чтение файлов в массивы NumPy и как вывести на экран размеры массивов.

Фрагмент кода 5.1 — Чтение набора данных MNIST и вывод его параметров

```
import idx2numpy
TRAIN_IMAGE_FILENAME = '../data/mnist/train-images.idx3-ubyte'
TRAIN_LABEL_FILENAME = '../data/mnist/train-labels.idx1-ubyte'
TEST_IMAGE_FILENAME = '../data/mnist/t10k-images.idx3-ubyte'
TEST_LABEL_FILENAME = '../data/mnist/t10k-labels.idx1-ubyte'
# Чтение файлов
train_images = idx2numpy.convert_from_file(TRAIN_IMAGE_FILENAME)
train_labels = idx2numpy.convert_from_file(TRAIN_LABEL_FILENAME)
test_images = idx2numpy.convert_from_file(TEST_IMAGE_FILENAME)
test_labels = idx2numpy.convert_from_file(TEST_LABEL_FILENAME)
# Печать размеров массивов
print('dimensions of train_images: ', train_images.shape)
print('dimensions of train_labels: ', train_labels.shape)
print('dimensions of test_images: ', test_images.shape)
print('dimensions of test_images: ', test_labels.shape)
```

Фрагмент кода 4.2 выводит первую обучающую метку и шаблон изображения.

Фрагмент кода 4.2 — Вывод одного из обучающих примеров

```
# Печатаем один обучающий пример
print('метка первого обучающего примера: ', train_labels[0])
print('---начало паттерна первого обучающего примера---')
for line in train_images[0]:
      for num in line:
            if num > 0:
                  print('*', end = ' ')
      else:
            print(' ', end = ' ')
    print('')
print('--- начало паттерна первого обучающего примера ---')
```

В результате выполнения программы получаем следующее:

```
метка для первого тренировочного примера: 5
---начало образца для первого обучающего примера---
                            ************
                        ****************
                       ****************
                       ************
                       *********
                        *****
                          ****
                          ****
                               ******
                                ******
                                 ******
                                      *****
                                        ****
                                      *******
                                   ********
                                 *********
                             **********
                          **********
                         *********
                        ********
---конец образца для первого обучающего примера---
```

1. *Напишите программу, которая выводит пять вариантов написания номера вашего варианта по списку группы. Вывод необходимо осуществлять цифрами из набора MNIST.*

Теперь импортируем несколько общих модулей, убедимся, что MatplotLib отображает цифры в строке, и подготовим функцию для сохранения рисунков. Также проверяем, установлен ли Python 3.5 или более поздней версии, а также Scikit-Learn ≥0.20.

Фрагмент кода 5.3 — Применение Scikit-Learn

```
# Требуется версия Python ≥3.5
import sys
assert sys.version_info >= (3, 5)

# Проверяем, запущен ли данный документ в Colab или Kaggle?
IS_COLAB = "google.colab" in sys.modules
IS_KAGGLE = "kaggle_secrets" in sys.modules

# Требуется версия Scikit-Learn ≥0.20
import sklearn
assert sklearn.__version__ >= "0.20"

# импортируем общие библиотеки
import numpy as np
import os

# для стабильного запуска примера более одного раза
np.random.seed(42)

# Рисуем симпатичные рисунки
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)

# Указываем куда сохранять рисунки
PROJECT_ROOT_DIR = "."
CHAPTER_ID = "classification"
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR, "images", CHAPTER_ID)
os.makedirs(IMAGES_PATH, exist_ok=True)

def    save_fig(fig_id,    tight_layout=True,    fig_extension="png",
resolution=300):
    path = os.path.join(IMAGES_PATH, fig_id + "." + fig_extension)
    print("Saving figure", fig_id)
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)
```

Теперь поработаем с MNIST. Введите в ячейки нового документа следующий код:

```
#
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784', version=1, as_frame=False)
mnist.keys()

#
X, y = mnist["data"], mnist["target"]
X.shape

#
y.shape

#
28 * 28

#
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt

#
some_digit = X[0]
some_digit_image = some_digit.reshape(28, 28)
plt.imshow(some_digit_image, cmap=mpl.cm.binary)
plt.axis("off")

#
save_fig("some_digit_plot")
plt.show()

#
y[0]

#
y = y.astype(np.uint8)

#
def plot_digit(data):
    image = data.reshape(28, 28)
    plt.imshow(image, cmap = mpl.cm.binary,
               interpolation="nearest")
    plt.axis("off")


# Выводим много цифр…
def plot_digits(instances, images_per_row=10, **options):
    size = 28
    images_per_row = min(len(instances), images_per_row)

    # Эквивалентно n_rows = ceil(len(instances) / images_per_row)
    n_rows = (len(instances) - 1) // images_per_row + 1

    # Добавляем пустое изображение, чтобы заполнить конец таблицы
    n_empty = n_rows * images_per_row - len(instances)
    padded_instances = np.concatenate([instances, np.zeros((n_empty,
size * size))], axis=0)

    # Изменяем массив так, чтобы он представлял сетку изображений 28×28
    image_grid = padded_instances.reshape((n_rows, images_per_row,
```

```
size, size))

    # Комбинируем оси 0 и 2 (вертик ось сетки и гориз ось
    # и оси 1 и 3 (гориз оси). Сначала перемещаем оси так,
    # чтобы они стояли друг за другом. Используем transpose()
    # Только после этого можно изменить форму массива
    big_image = image_grid.transpose(0, 2, 1, 3).reshape(n_rows * size,
images_per_row * size)

    # Теперь мы получили большую картинку, которую не стыдно показать:
    plt.imshow(big_image, cmap = mpl.cm.binary, **options)
    plt.axis("off")

#
plt.figure(figsize=(9,9))
example_images = X[:100]
plot_digits(example_images, images_per_row=10)
save_fig("more_digits_plot")
plt.show()
```

2. *Разберитесь в приведённом коде. В отчёте заполните комментарии, поясняющие работу соответствующего блока кода везде, где встречается символ «#».*
3. *Программу, которая выводит пять вариантов написания номера вашего варианта по списку группы цифрами из набора MNIST, реализуйте теперь модифицировав Фрагмент кода 5.3 и далее.*