МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ МОРСКОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
 (СПБГМТУ)

| ФАКУЛЬТЕТ ЦИФРОВЫХ ПРОМЫШЛЕННЫХ ТЕХНОЛОГИЙ  КАФЕДРА КИБЕРФИЗИЧЕСКИХ СИСТЕМ |
| --- |

ЛАБОРАТОРНАЯ РАБОТА №5

ВАРИАНТ 8

«РАБОТА С ДАТАСЕТОМ»

|  | |
| --- | --- |
|  | Выполнил  студент группы 20221 | | |
|  | Лаптев Иван Александрович | | |
|  | Проверила | | |
|  | Кайнова Татьяна Денисовна | | |
| Санкт-Петербург  2024 год | | |

**Оглавление**

[Цели работы……………………………………………………………………………………... 3](#_30j0zll)

[Результат работы………………………………………………………………………………… 4](#_1fob9te)

Листинг кода……[………………………………………………………………………………... 7](#_3znysh7)

# **ЦЕЛИ РАБОТЫ**

В ходе выполнения лабораторной работы необходимо написать программу, которая выводит пять вариантов написания номера варианта по списку группы. Вывод необходимо осуществлять цифрами из набора MNIST.

Также необходимо разобраться в приведённом коде и заполнить комментарии, поясняющие работу соответствующего блока кода везде, где встречается символ «#».

# **РЕЗУЛЬТАТЫ РАБОТЫ**

Вариант работы 20, а значит нам нужно вывести цифру 0. Обращаемся к датасету MNIST и получаем следующие идентификаторы для цифры 0: 1, 21, 34, 37, 51

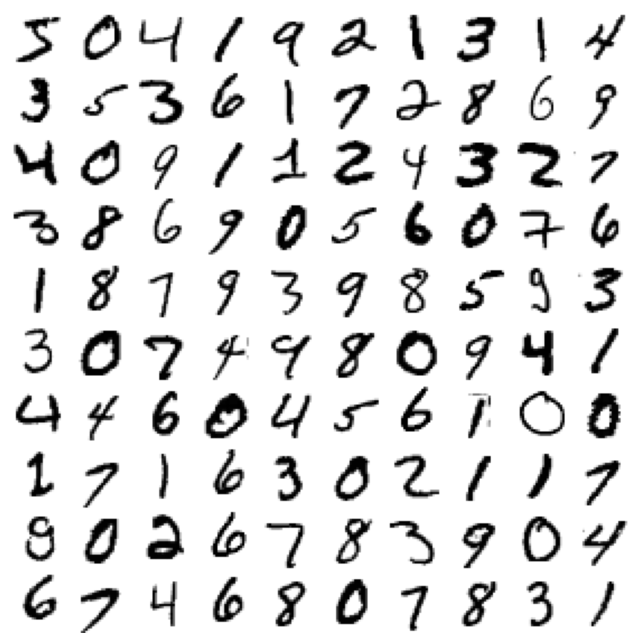


Рисунок 1 – Датасет MNIST

Полученные изображения:

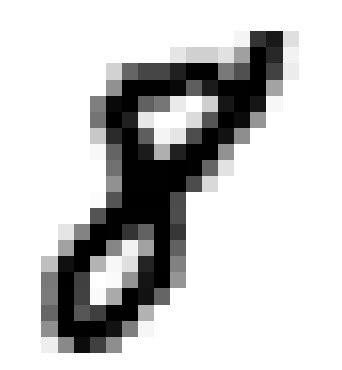


Рисунок 2 – Первое изображение

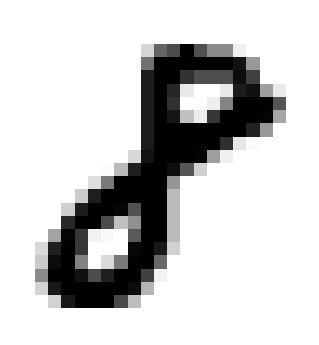


Рисунок 3 – Второе изображение

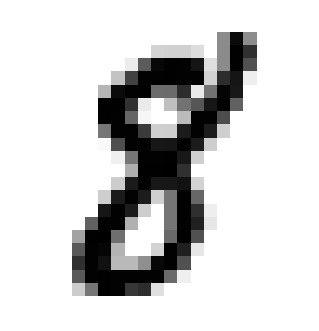


Рисунок 4 – Третье изображение

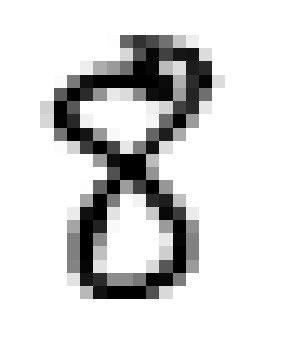


Рисунок 5 – Четвёртое изображение

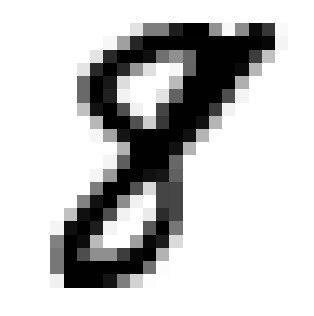


Рисунок 6 – Пятое изображение

# **ЛИСТИНГ КОДА**

Листинг 1 – Код с комментариями

import sys # Импортируем модуль sys для работы с параметрами системы

assert sys.version\_info >= (3, 5) # Проверяем, что версия Python не ниже 3.5

IS\_COLAB = "google.colab" in sys.modules # Проверяем, запущен ли код в Google Colab

IS\_KAGGLE = "kaggle\_secrets" in sys.modules # Проверяем, запущен ли код на платформе Kaggle

import sklearn # Импортируем библиотеку Scikit-learn

assert sklearn.\_\_version\_\_ >= "0.20" # Проверяем, что версия scikit-learn не ниже 0.20

import numpy as np # Импортируем библиотеку NumPy для работы с массивами

import os # Импортируем библиотеку os для работы с операционной системой

np.random.seed(42) # Устанавливаем фиксированное начальное значение для генератора случайных чисел (для воспроизводимости)

import matplotlib as mpl # Импортируем библиотеку Matplotlib для визуализации данных

import matplotlib.pyplot as plt # Импортируем модуль для рисования графиков

mpl.rc('axes', labelsize=14) # Настройка размеров шрифтов меток на осях графиков

mpl.rc('xtick', labelsize=12) # Настройка размера шрифта для меток на оси X

mpl.rc('ytick', labelsize=12) # Настройка размера шрифта для меток на оси Y

PROJECT\_ROOT\_DIR = "." # Устанавливаем текущую директорию как корень проекта

CHAPTER\_ID = "classification" # Идентификатор текущей главы проекта

IMAGES\_PATH = os.path.join(PROJECT\_ROOT\_DIR, "images", CHAPTER\_ID) # Путь для сохранения изображений

os.makedirs(IMAGES\_PATH, exist\_ok=True) # Создаем директорию для изображений, если она не существует

def save\_fig(fig\_id, tight\_layout=True, fig\_extension="png", resolution=300): # Функция для сохранения изображения

path = os.path.join(IMAGES\_PATH, fig\_id + "." + fig\_extension) # Формируем полный путь к файлу изображения

print("Saving figure", fig\_id) # Выводим сообщение о сохранении изображения

if tight\_layout: # Если tight\_layout=True, применяем его к графику

plt.tight\_layout()

plt.savefig(path, format=fig\_extension, dpi=resolution) # Сохраняем график в файл

from sklearn.datasets import fetch\_openml # Импортируем функцию для загрузки данных с OpenML

mnist = fetch\_openml('mnist\_784', version=1, as\_frame=False) # Загружаем датасет MNIST

mnist.keys() # Выводим ключи, доступные в загруженном датасете

X, y = mnist["data"], mnist["target"] # Разделяем данные на признаки (X) и целевую переменную (y)

X.shape # Получаем размерность матрицы данных

y.shape # Получаем размерность массива меток

28 \* 28 # Размер изображения (28x28 пикселей)

import matplotlib as mpl # Импортируем Matplotlib для рисования

import matplotlib.pyplot as plt # Импортируем функции для работы с графиками

some\_digit = X[17] # Выбираем 17-й элемент из данных

some\_digit\_image = some\_digit.reshape(28, 28) # Преобразуем его в изображение размером 28x28

plt.imshow(some\_digit\_image, cmap=mpl.cm.binary) # Отображаем изображение

plt.axis("off") # Отключаем оси

save\_fig("some\_digit\_plot") # Сохраняем изображение

plt.show() # Показываем изображение

y[0] # Получаем метку первого элемента

some\_digit = X[31] # Выбираем 31-й элемент из данных

some\_digit\_image = some\_digit.reshape(28, 28) # Преобразуем в изображение

plt.imshow(some\_digit\_image, cmap=mpl.cm.binary) # Отображаем изображение

plt.axis("off") # Отключаем оси

plt.show() # Показываем изображение

y[0] # Получаем метку первого элемента

some\_digit = X[41] # Выбираем 41-й элемент из данных

some\_digit\_image = some\_digit.reshape(28, 28) # Преобразуем в изображение

plt.imshow(some\_digit\_image, cmap=mpl.cm.binary) # Отображаем изображение

plt.axis("off") # Отключаем оси

plt.show() # Показываем изображение

y[0] # Получаем метку первого элемента

some\_digit = X[46] # Выбираем 46-й элемент из данных

some\_digit\_image = some\_digit.reshape(28, 28) # Преобразуем в изображение

plt.imshow(some\_digit\_image, cmap=mpl.cm.binary) # Отображаем изображение

plt.axis("off") # Отключаем оси

plt.show() # Показываем изображение

y[0] # Получаем метку первого элемента

some\_digit = X[55] # Выбираем 55-й элемент из данных

some\_digit\_image = some\_digit.reshape(28, 28) # Преобразуем в изображение

plt.imshow(some\_digit\_image, cmap=mpl.cm.binary) # Отображаем изображение

plt.axis("off") # Отключаем оси

plt.show() # Показываем изображение

y[0] # Получаем метку первого элемента

y = y.astype(np.uint8) # Преобразуем метки в тип данных uint8

def plot\_digit(data): # Функция для отображения одной цифры

image = data.reshape(28, 28) # Преобразуем данные в изображение 28x28

plt.imshow(image, cmap=mpl.cm.binary, interpolation="nearest") # Отображаем изображение

plt.axis("off") # Отключаем оси

def plot\_digits(instances, images\_per\_row=10, \*\*options): # Функция для отображения нескольких цифр

size = 28 # Размер изображений (28x28)

images\_per\_row = min(len(instances), images\_per\_row) # Ограничиваем количество изображений в ряду

n\_rows = (len(instances) - 1) // images\_per\_row + 1 # Вычисляем количество рядов для изображений

n\_empty = n\_rows \* images\_per\_row - len(instances) # Вычисляем, сколько пустых изображений нужно добавить

padded\_instances = np.concatenate([instances, np.zeros((n\_empty, size \* size))], axis=0) # Добавляем пустые изображения

image\_grid = padded\_instances.reshape((n\_rows, images\_per\_row, size, size)) # Формируем сетку изображений

big\_image = image\_grid.transpose(0, 2, 1, 3).reshape(n\_rows \* size, images\_per\_row \* size) # Переставляем оси и получаем большое изображение

plt.imshow(big\_image, cmap=mpl.cm.binary, \*\*options) # Отображаем сетку изображений

plt.axis("off") # Отключаем оси

plt.figure(figsize=(9,9)) # Создаем фигуру для отображения

example\_images = X[:100] # Выбираем первые 100 изображений

plot\_digits(example\_images, images\_per\_row=10) # Отображаем изображения в виде сетки

save\_fig("more\_digits\_plot") # Сохраняем изображение

plt.show() # Показываем изображение