**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

**отчет**

**по лабораторной работе №1**

**по дисциплине «Автоматизация схемотехнического проектирования»**

**на тему «Генерация модельных наборов данных»**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 1301 |  | Семейкин С.А. |
|  |  | Гальченко М.А. |
| Преподаватель |  | Боброва Ю.О. |

Санкт-Петербург

2025

**Цель:**

Получение навыков работы с numpy-массивами и написание функций на языке Python на примере генерации массивов произвольно распределенных данных.

**Ход работы:**

1. Создадим переменные, распределение по нормальному закону с незначительно различными средними и дисперсиями.
2. Создадим переменные, соответствующие классам.
3. Визуализируем результаты, чтобы оценить пересекаемость классов.
4. В первую очередь построим гистограммы для каждого признака и скаттерограммы для каждой пары признаков.

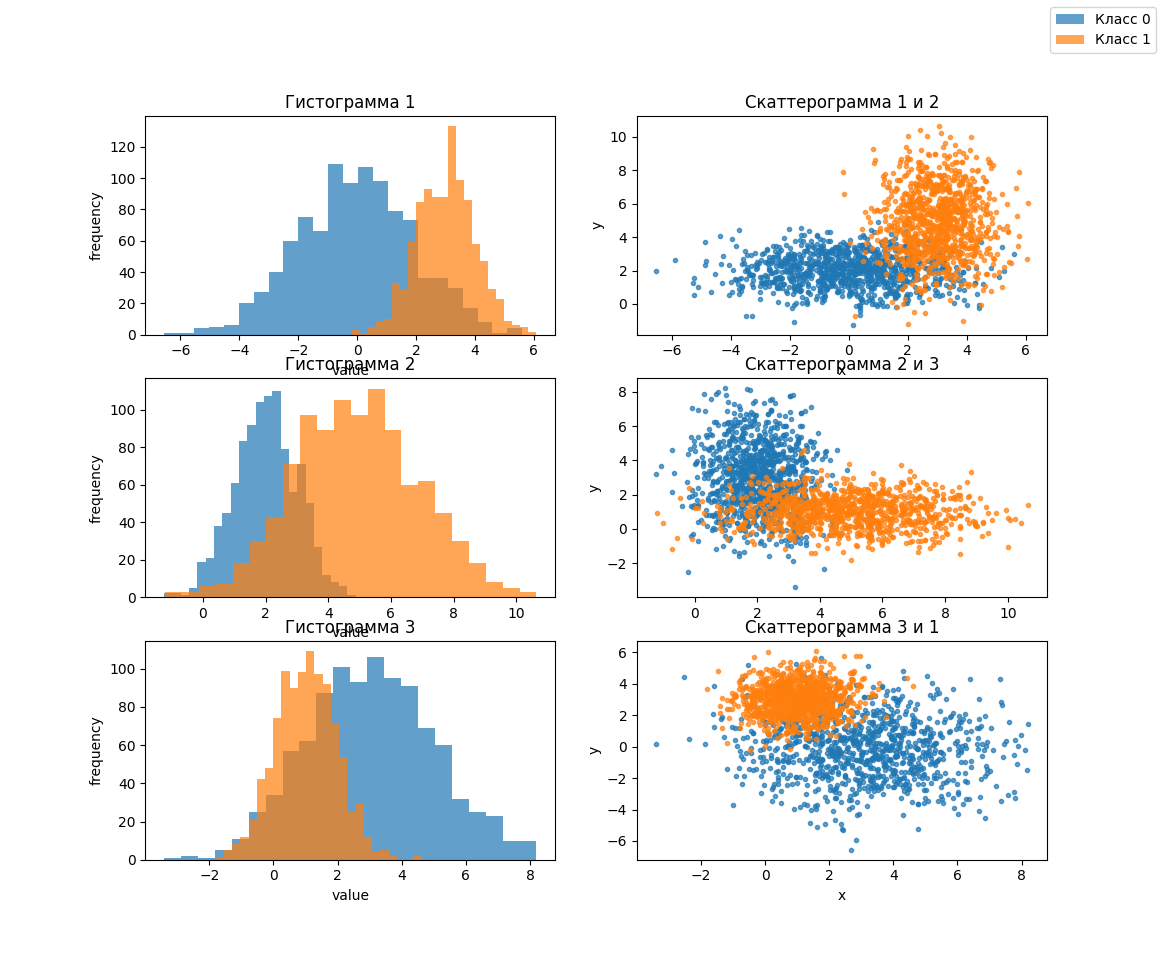


Рисунок Гистограммы и скаттерограммы признаков

1. Создадим метод класса DataGenerator nonlinear\_dataset\_N, который генерирует двумерный массив данных, распределенный в пространстве в виде заданных фигур на рисунке 2.



Рисунок Необходимый вид распределения

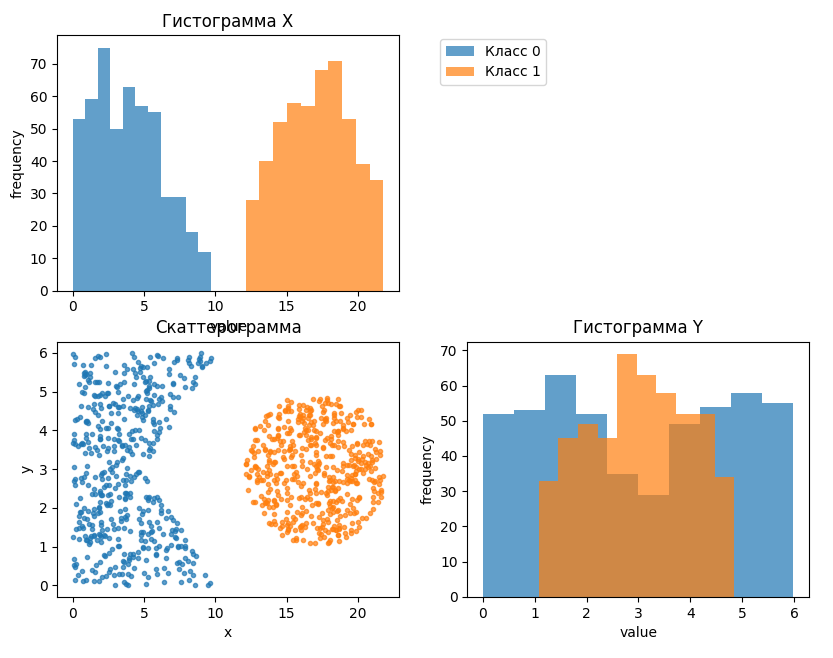


Рисунок Гистограммы и скаттерограмма заданного распределения

**Вывод:**

В ходе лабораторной работы были получены навыки работы с numpy‑массивами и написания функций на языке Python на примере генерации массивов произвольно распределенных данных.

Благодаря изучению numpy-массивов была получена возможность эффективно обрабатывать и манипулировать данными. Кроме того, реализация функций для генерации массивов с произвольным распределением данных позволила углубить понимание работы с различными статистическими распределениями и их влиянием на создаваемые данные.

**Листинг:**

data\_generator.py

import numpy as np  
from typing import Any  
from matplotlib.path import Path  
  
  
def is\_point\_in\_pentagon(x, y, rect\_x, rect\_y, rect\_width, rect\_height):  
 *# Определяем вершины пятиугольника* bottom\_left = (rect\_x, rect\_y)  
 bottom\_right = (rect\_x + rect\_width, rect\_y)  
 top\_right = (rect\_x + rect\_width, rect\_y + rect\_height)  
 top\_left = (rect\_x, rect\_y + rect\_height)  
 center\_right = (rect\_x + rect\_width / 2, rect\_y + rect\_height / 2)  
  
 pentagon\_vertices = [bottom\_left, bottom\_right, center\_right, top\_right, top\_left, bottom\_left]  
  
 pentagon\_path = Path(np.array(pentagon\_vertices))  
  
 return pentagon\_path.contains\_point((x, y)) *# , pentagon\_vertices*def is\_point\_in\_ellipse(x, y, center\_x, center\_y, width, height):  
 norm\_x = (x - center\_x) / (width / 2)  
 norm\_y = (y - center\_y) / (height / 2)  
 return norm\_x \*\* 2 + norm\_y \*\* 2 <= 1  
  
  
class DataGenerator:  
 @staticmethod  
 def norm\_dataset(mu: tuple[list[int], list[int]],  
 sigma: tuple[list[int], list[int]],  
 N: int  
 ) -> \  
 tuple[np.ndarray[np.floating[np.float64], Any],  
 np.ndarray[np.floating[np.float64], Any],  
 np.ndarray[np.floating[np.float64], Any],  
 np.ndarray[np.floating[np.float64], Any]]:  
 mu0 = mu[0]  
 mu1 = mu[1]  
 sigma0 = sigma[0]  
 sigma1 = sigma[1]  
  
 col = len(mu0)  
 class0 = np.random.normal(mu0[0], sigma0[0], [N, 1]) *# инициализируем первый столбец* class1 = np.random.normal(mu1[0], sigma1[0], [N, 1])  
 for i in range(1, col):  
 v0 = np.random.normal(mu0[i], sigma0[i], [N, 1])  
 class0 = np.hstack((class0, v0))  
 v1 = np.random.normal(mu1[i], sigma1[i], [N, 1])  
 class1 = np.hstack((class1, v1))  
 X = np.vstack((class0, class1))  
 *# print(X)  
 # print(X.size)* Y0 = np.zeros((N, 1), dtype=bool)  
 Y1 = np.ones((N, 1), dtype=bool)  
 Y = np.vstack((Y0, Y1)).ravel()  
  
 rng = np.random.default\_rng()  
 arr = np.arange(2 \* N) *# индексы для перемешивания [ 0, 1 ,2 .... 1999]* rng.shuffle(arr)  
 X = X[arr]  
 Y = Y[arr]  
  
 return X, Y, class0, class1  
  
 @staticmethod  
 def nonlinear\_dataset\_N(num\_points):  
 *"""  
 Генерирует два массива точек, распределенных в пределах пятиугольника и эллипса.  
 """* pentagon\_params = (0, 0, 10, 6) *# x, y, width, height* ellipse\_params = (17, 3, 10, 4) *# center\_x, center\_y, width, height* pentagon\_points = []  
 ellipse\_points = []  
  
 x\_min, x\_max = min(pentagon\_params[0], ellipse\_params[0] - ellipse\_params[2] / 2), max(  
 pentagon\_params[0] + pentagon\_params[2], ellipse\_params[0] + ellipse\_params[2] / 2)  
 y\_min, y\_max = min(pentagon\_params[1], ellipse\_params[1] - ellipse\_params[3] / 2), max(  
 pentagon\_params[1] + pentagon\_params[3], ellipse\_params[1] + ellipse\_params[3] / 2)  
  
 while len(pentagon\_points) < num\_points or len(ellipse\_points) < num\_points:  
 x = np.random.uniform(x\_min, x\_max)  
 y = np.random.uniform(y\_min, y\_max)  
  
 if len(pentagon\_points) < num\_points and is\_point\_in\_pentagon(x, y, \*pentagon\_params):  
 pentagon\_points.append((x, y))  
  
 if len(ellipse\_points) < num\_points and is\_point\_in\_ellipse(x, y, \*ellipse\_params):  
 ellipse\_points.append((x, y))  
 class0 = np.array(pentagon\_points)  
 class1 = np.array(ellipse\_points)  
 X = np.vstack((class0,class1))  
 Y0 = np.zeros((num\_points, 1), dtype=bool)  
 Y1 = np.ones((num\_points, 1), dtype=bool)  
 Y = np.vstack((Y0, Y1)).ravel()  
  
 rng = np.random.default\_rng()  
 arr = np.arange(num\_points)  
 rng.shuffle(arr)  
 X = X[arr]  
 Y = Y[arr]  
 return X, Y, class0, class1

norm\_dataset.py

import matplotlib.pyplot as plt  
from data\_generator import DataGenerator  
  
mu0 = [0, 2, 3]  
mu1 = [3, 5, 1]  
sigma0 = [2, 1, 2]  
sigma1 = [1, 2, 1]  
col = len(mu0)  
N = 1000  
X, Y, class0, class1 = DataGenerator.norm\_dataset((mu0, mu1), (sigma0, sigma1), N)  
  
train\_size = 0.7  
trainCount = round(train\_size \* N \* 2) *# \*2 потому что было 2 класса*Xtrain = X[0:trainCount]  
Xtest = X[trainCount:N \* 2 + 1]  
Ytrain = Y[0:trainCount]  
Ytest = Y[trainCount:N \* 2 + 1]  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 figure, axis = plt.subplots(col, 2)  
 for i in range(0, col):  
 axis[i, 0].set\_title(f"Гистограмма {i + 1}")  
 hist\_class0 = axis[i, 0].hist(class0[:, i], bins='auto', alpha=0.7,  
 label='Класс 0')  
 hist\_class1 = axis[i, 0].hist(class1[:, i], bins='auto', alpha=0.7, label='Класс 1')  
 axis[i, 0].set\_xlabel("value")  
 axis[i, 0].set\_ylabel("frequency")  
 axis[i, 1].set\_xlabel("x")  
 axis[i, 1].set\_ylabel("y")  
 if i != col - 1:  
 axis[i, 1].set\_title(f"Скаттерограмма {i + 1} и {i + 2}")  
 axis[i, 1].scatter(class0[:, i], class0[:, i + 1], marker=".", alpha=0.7, label='Класс 0')  
 axis[i, 1].scatter(class1[:, i], class1[:, i + 1], marker=".", alpha=0.7, label='Класс 1')  
 else:  
 axis[i, 1].set\_title(f"Скаттерограмма {i + 1} и {1}")  
  
 axis[i, 1].scatter(class0[:, i], class0[:, 0], marker=".", alpha=0.7, label='Класс 0')  
 axis[i, 1].scatter(class1[:, i], class1[:, 0], marker=".", alpha=0.7, label='Класс 1')  
  
 figure.legend(['Класс 0', 'Класс 1'])  
 plt.show()

custom\_dataset.py

import matplotlib.pyplot as plt  
from data\_generator import DataGenerator  
  
col = 2  
N = 500  
X, Y, class0, class1 = DataGenerator.nonlinear\_dataset\_N(N)  
figure, axis = plt.subplots(2, 2)  
  
axis[0, 0].set\_title(f"Гистограмма X")  
axis[0, 0].set\_xlabel("value")  
axis[0, 0].set\_ylabel("frequency")  
axis[0, 0].hist(class0[:, 0], bins='auto', alpha=0.7,  
 label='5 угольник') *# параметр alpha позволяет задать прозрачность цвета\*axis[0, 0].hist(class1[:, 0], bins='auto', alpha=0.7,  
 label='Овал') *# параметр alpha позволяет задать прозрачность цвета*axis[1, 1].set\_title(f"Гистограмма Y")  
axis[1, 1].set\_xlabel("value")  
axis[1, 1].set\_ylabel("frequency")  
axis[1, 1].hist(class0[:, 1], bins='auto', alpha=0.7,  
 label='5 угольник') *# параметр alpha позволяет задать прозрачность цвета*axis[1, 1].hist(class1[:, 1], bins='auto', alpha=0.7,  
 label='Овал') *# параметр alpha позволяет задать прозрачность цвета*axis[1, 0].set\_title(f"Скаттерограмма")  
axis[1, 0].set\_xlabel("x")  
axis[1, 0].set\_ylabel("y")  
axis[1, 0].scatter(class0[:, 0], class0[:, 1], marker=".", alpha=0.7, label='5 угольник')  
axis[1, 0].scatter(class1[:, 0], class1[:, 1], marker=".", alpha=0.7, label='Овал')  
  
figure.legend(['Класс 0', 'Класс 1'])  
plt.savefig("test\_jpg.png")  
plt.show()