МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра САПР

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Автоматизация схемотехнического проектирования»

Тема: «КЛАССИФИКАТОР НА ОСНОВЕ ЛОГИСТИЧЕСКОЙ РЕГРЕССИИ С ГРАДИЕНТНЫМ СПУСКОМ»

Студент гр. 2301 _____ Комиссаров П.Е.

Преподаватель: _____ Боброва Ю.О.

Санкт-Петербург
2026

Цель работы:

Разработка модели классификатора на основе логистической регрессии, изучение его свойств и принципов работы, получение навыков программирования на Python и использования модуля scikit-learn

Основные теоретические положения

Логистическая регрессия — это модель **бинарной классификации** (два класса: 0 и 1).

Результат работы модели — не «класс», а **вероятность** принадлежности объекту классу 1. По этой вероятности затем принимается решение: если вероятность > 0.5 → класс 1, иначе → класс 0.

Для объекта $\mathbf{X} = (x_1, x_2, \ldots, x_n)$ выход модели вычисляется так:

Линейная комбинация (логит):

$$z = b_0 + b_1 x_1 + b_2 x_2 + ... + b_n x_n$$

Сигмоида (вероятность):

$$\hat{y} = \frac{1}{1 + e^{-z}}$$

• $\mathbf{b_0}$ — свободный член (intercept).

• $\mathbf{b_1 \ldots b_n}$ — коэффициенты при признаках (веса).

• **z** может быть любым числом; после подстановки в формулу $\hat{\mathbf{y}}$ всегда лежит в интервале (0, 1), то есть это вероятность.

Правило классификации: - если $\hat{\mathbf{y}} > \mathbf{0.5}$ → предсказываем класс **1**; - если $\hat{\mathbf{y}} \leq \mathbf{0.5}$ → предсказываем класс **0**.

Граница между классами задаётся уравнением **z = 0** (линейная гиперплоскость в пространстве признаков).

Связь с линейной регрессией

- **Линейная регрессия:** выход = z (линейная функция), используется для **регрессии** (предсказание числа).
- **Логистическая регрессия:** выход = σ(z) = $1/(1+e^{-z})$. Та же линейная комбинация z, но результат пропущен через **сигмоиду**, чтобы получить вероятность. Используется для **классификации**.

То есть «логистическая регрессия» — это по сути линейная модель + сигмоида на выходе.

Обучение с учителем и функция потерь

Модель относится к **обучению с учителем**: у нас есть размеченные данные (X, Y), и мы подбираем коэффициенты $\mathbf{b_0}$, $\mathbf{b_1}$, …, $\mathbf{b_n}$ так, чтобы минимизировать ошибку предсказания.

Функция потерь (cross-entropy / log loss) для бинарной классификации:

$$loss = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$$

где:

- **m** — число объектов в выборке;
- $\mathbf{y_i}$ — истинная метка (0 или 1);
- $\hat{\mathbf{y}}_\mathbf{i}$ — предсказанная вероятность класса 1 для объекта i.

Почему два слагаемых и логарифм:

- Для объекта класса 1 ($y_i=1$) второе слагаемое обнуляется, остаётся $-\log(\hat{y}_i)$. Чтобы loss был меньше, нужно, чтобы $\hat{y}_i$ было ближе к 1.

- Для объекта класса 0 ($y_i=0$) первое слагаемое обнуляется, остаётся $-\log(1-\hat{y}_i)$. Чтобы loss был меньше, нужно, чтобы $\hat{y}_i$ было ближе к 0.

Градиентный спуск

Градиентный спуск — метод обучения: мы итеративно обновляем веса в направлении, противоположном градиенту функции потерь, чтобы уменьшить loss.

- Вычисляем градиент loss по весам.
- Делаем шаг: веса = веса − шаг_обучения × градиент.
- Повторяем до сходимости.

Регуляризация

Регуляризация — штраф за слишком большие веса. Добавляется к функции потерь (например, L2: $\lambda\|w\|^2$).

Это:

- уменьшает переобучение;

- стабилизирует решение.

В LogisticRegression задаётся параметром C (обратная сила регуляризации: меньше C — сильнее регуляризация).

Чувствительность и специфичность

При интерпретации:

класс 0 = отсутствие признака (например, «здоров»),

класс 1 = наличие признака (например, «болен»).

- **Чувствительность (Sensitivity, TPR):** доля реально положительных, которых модель правильно отнесла к классу 1

$$\text{Чувствительность} = \frac{TP}{TP + FN}$$

- **Специфичность (Specificity, TNR):** доля реально отрицательных, которых модель правильно отнесла к классу 0

$$\text{Специфичность} = \frac{TN}{TN + FP}$$

где:

- **TP** — истинно положительные (истина 1, предсказание 1);
- **TN** — истинно отрицательные (истина 0, предсказание 0);
- **FP** — ложно положительные (истина 0, предсказание 1);
- **FN** — ложно отрицательные (истина 1, предсказание 0).

Точность (accuracy):

$$\text{Точность} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{\text{число верных предсказаний}}{\text{всего объектов}}.$$

Принцип работы классификатора на основе логистической регрессии

Основные этапы работы модели

Вычисление линейной комбинации признаков (логита)

Для каждого объекта с признаками $X = (x_1, x_2, \ldots, x_n)$ рассчитывается взвешенная сумма: $z = b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + \ldots + b_n \cdot x_n$

Преобразование логита в вероятность с помощью сигмоиды Полученное значение z пропускается через сигмоидную. Результат $\hat{y}$ лежит в интервале [0; 1] и интерпретируется как вероятность принадлежности объекта к классу 1.

Принятие решения о классе Применяется пороговое правило (по умолчанию порог = 0,5):

если $\hat{y} \geq 0{,}5 \rightarrow$ предсказывается класс 1

если $\hat{y} < 0{,}5 \rightarrow$ предсказывается класс 0

Граница между классами в пространстве признаков — это гиперплоскость, задаваемая уравнением $z = 0$.

Обучение модели (подбор весов)

Обучение заключается в подборе таких весов b_0, b_1, …, b_n, чтобы предсказанные вероятности максимально соответствовали истинным меткам классов на обучающей выборке.

Процесс обучения включает следующие ключевые шаги:

Инициализация весов (обычно малыми значениями или нулями).

Для каждого объекта (или группы объектов) **вычисляется текущее предсказание** $\hat{y}$.

Рассчитывается ошибка предсказания с помощью **функции потерь**

К функции потерь может д**обавляться регуляризационный член**, который штрафует слишком большие значения весов и предотвращает переобучение.

Вычисляется градиент функции потерь по всем весам — направление, в котором нужно их изменить, чтобы ошибка уменьшилась.

Веса корректируются маленьким шагом в направлении, противоположном градиенту (метод градиентного спуска или его модификации, например SAGA).

Шаги повторяются итеративно до достижения сходимости или исчерпания максимального числа итераций.

В результате обучения получаются оптимальные веса, которые затем используются для предсказания на новых данных.

Ход выполнения работы

1. Создан скрипт, в начале импортированы модули numpy, matplotlib, pathlib и модель LogisticRegression из sklearn.linear_model.

2. С использованием функций лабораторной работы №1 (norm_dataset и nonlinear_dataset_5 из DataGenerator) созданы переменные для трёх экспериментов: Xtrain, Ytrain и Xtest, Ytest.

X — объекты двух классов (нормальное распределение для выборок А и Б, нелинейная форма для выборки В),

Y — метки класса. Разбиение на обучение и тест — 70% / 30% с перемешиванием.

3. Модель обучена на обучающей выборке методом fit(). Для воспроизводимости задан random_state равным номеру варианта (Nvar = 5).

4. Выбран оптимизационный алгоритм SAGA:

clf = LogisticRegression(random_state=Nvar, solver='saga').fit(Xtrain, Ytrain).

5. Получены предсказания для новых данных: predict() — метки классов 0/1; predict_proba() — вероятности принадлежности каждому классу (два столбца).

Для гистограмм используется вероятность класса 1 (второй столбец).

6. Точность оценена методом score():

acc_train = clf.score(Xtrain, Ytrain),

acc_test = clf.score(Xtest, Ytest).

7. Точность посчитана вручную:

acc_test = sum(Pred_test == Ytest) / len(Ytest).

8. Визуализированы результаты: построены гистограммы распределения вероятности принадлежности классу 1 на выходе классификатора для обучающей и тестовой выборок

Самостоятельное задание.

Рассчитаны чувствительность и специфичность по формулам TP/(TP+FN) и TN/(TN+FP) без использования методов библиотеки (функция sensitivity_specificity).

Изменены параметры генерируемых данных для выборки Б (сближены средние, увеличены СКО) — получено более плотное пересечение классов и ухудшение метрик.

На выборке В (нелинейно разделимые данные из nonlinear_dataset_5) оценена эффективность классификатора — логистическая регрессия как линейный метод даёт худшие результаты.

Гистограммы распределения вероятности и подписи к рисункам

По оси X — вероятность, по оси Y — число объектов. Два цвета: класс 1 (истинный) и класс 0 (истинный)

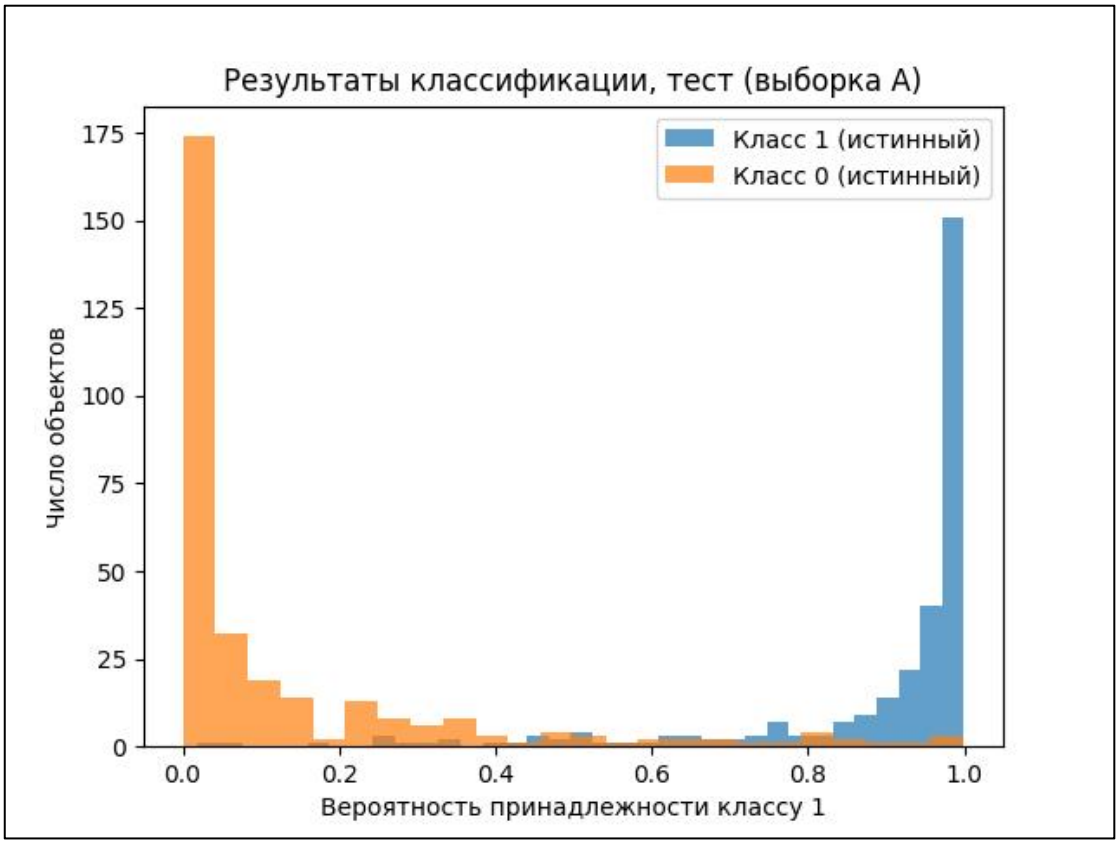

Рисунок 1 — Гистограмма распределения вероятности принадлежности классу 1, тестовая выборка A.

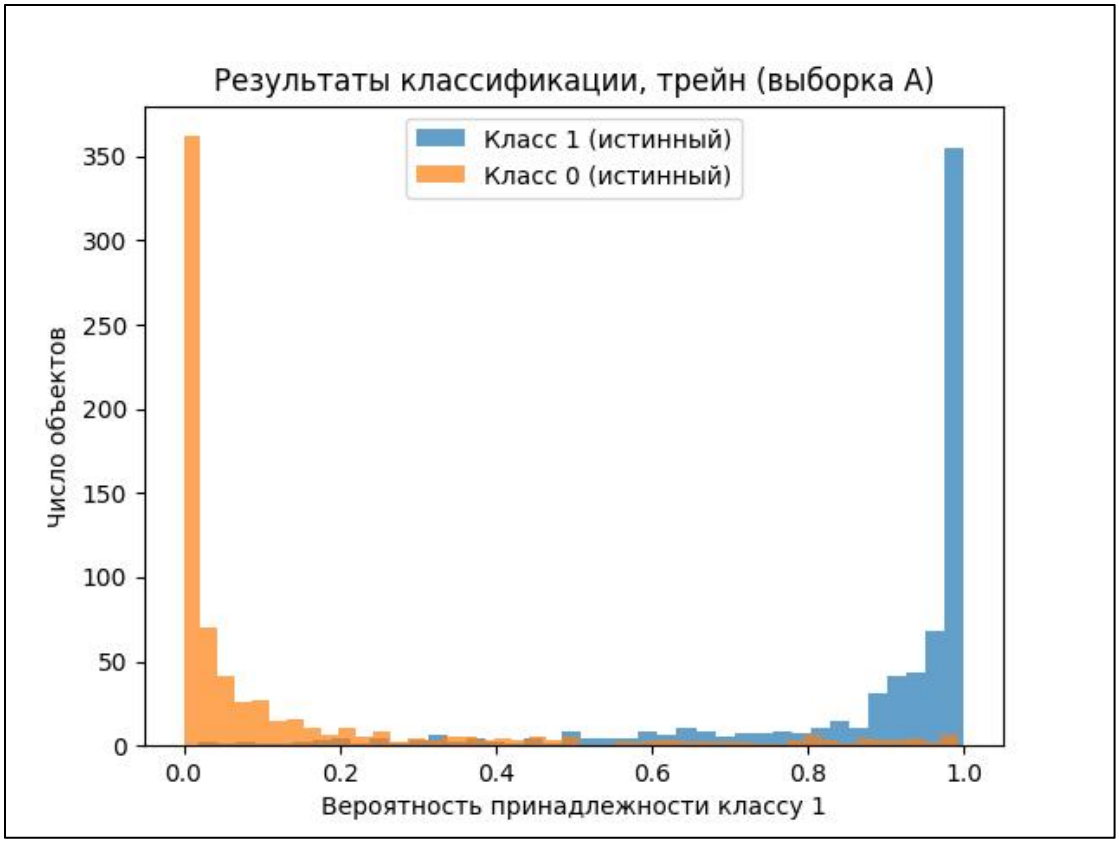

Рисунок 2 — Гистограмма, обучающая выборка A (хорошо разделимые).

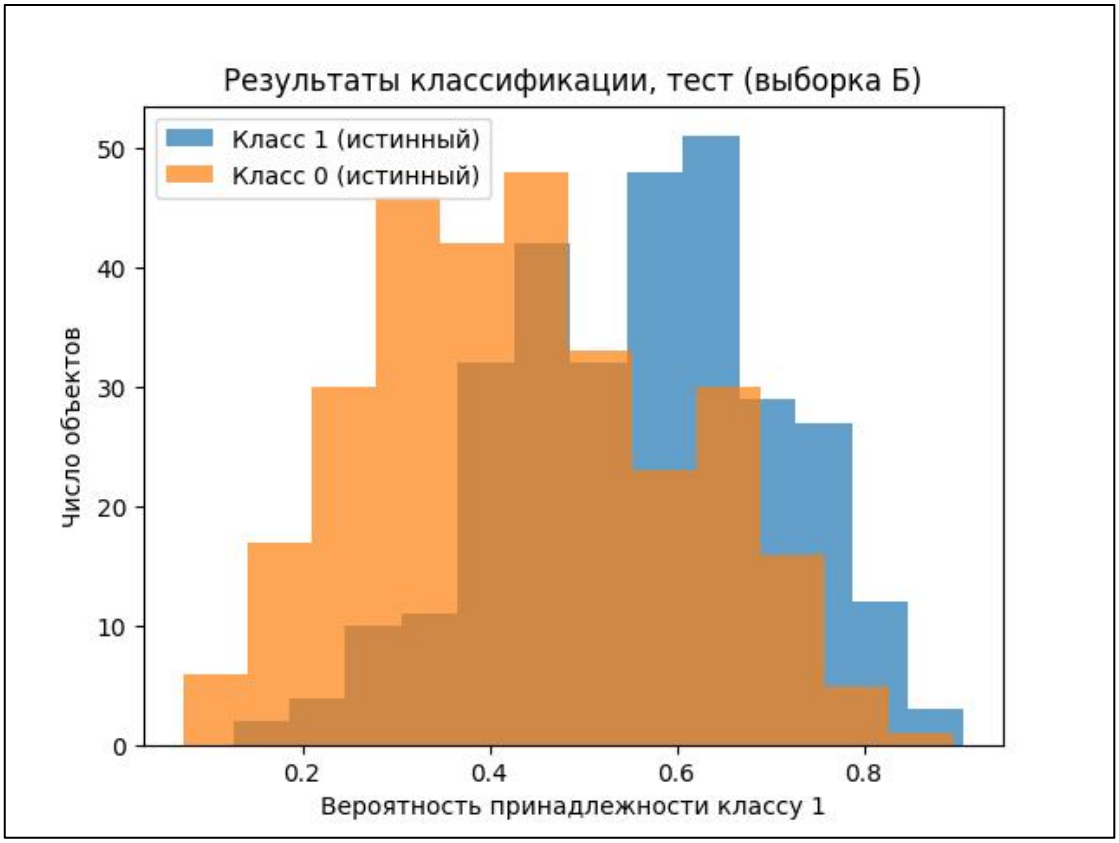

Рисунок 3 — Гистограмма, тестовая выборка Б (плохо разделимые нормальные данные).

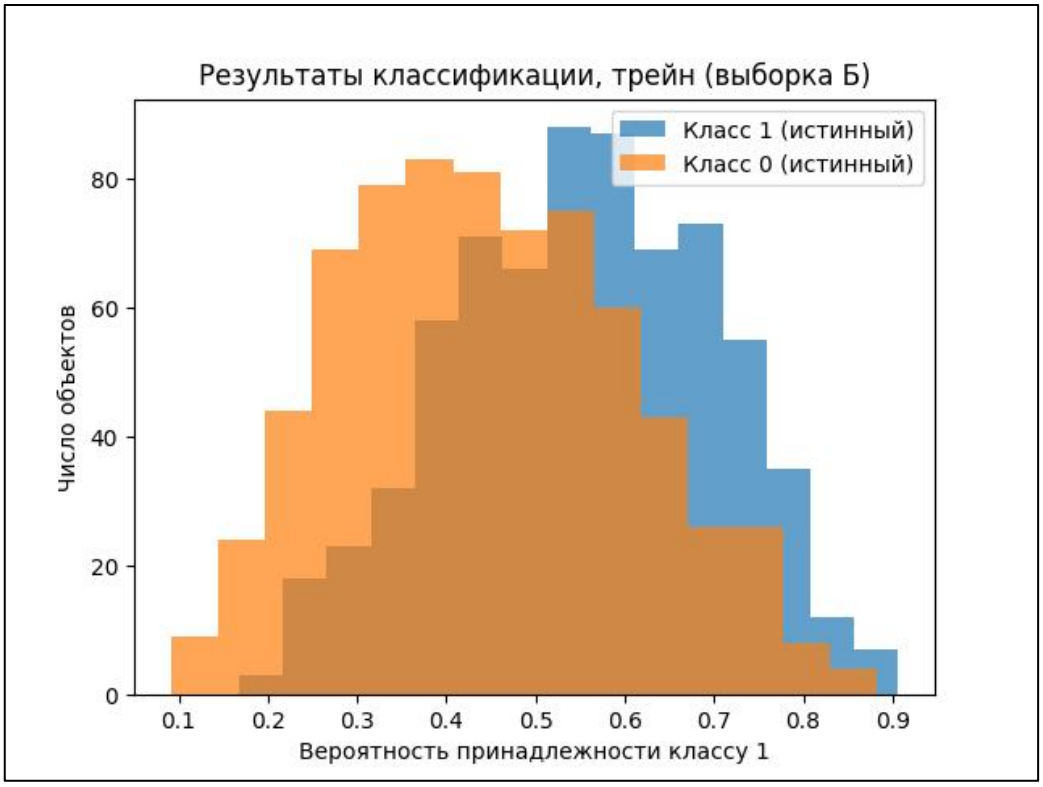

Рисунок 4 — Гистограмма, обучающая выборка Б (плохо разделимые).

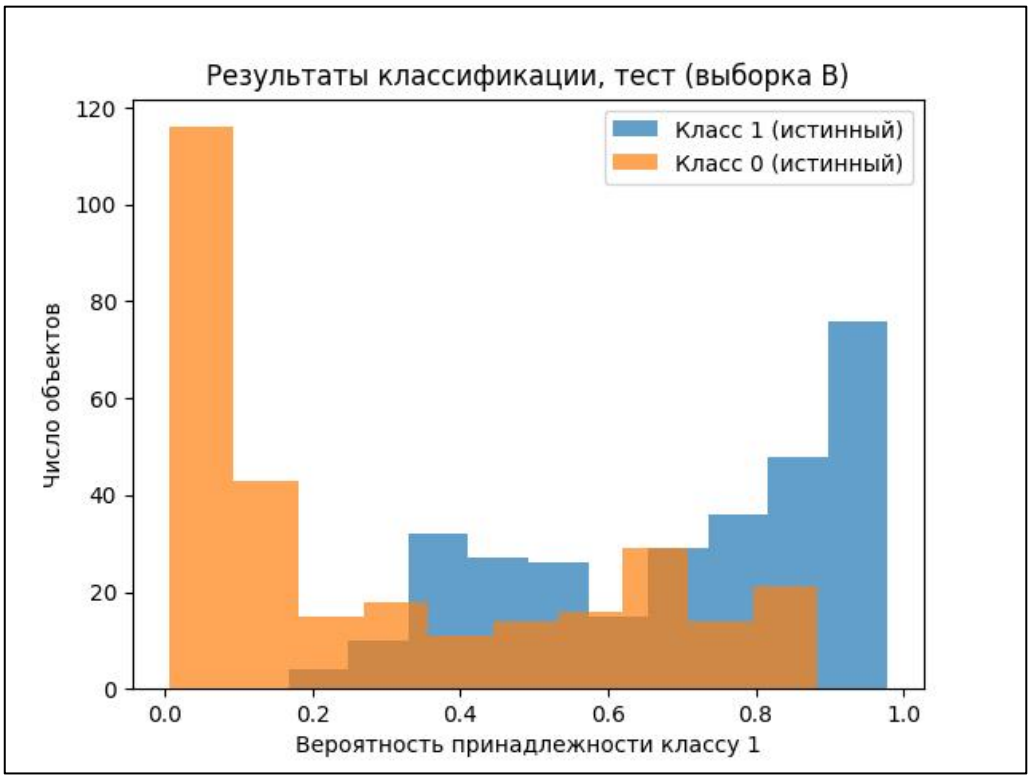

Рисунок 5 — Гистограмма, тестовая выборка В (нелинейно разделимые данные).

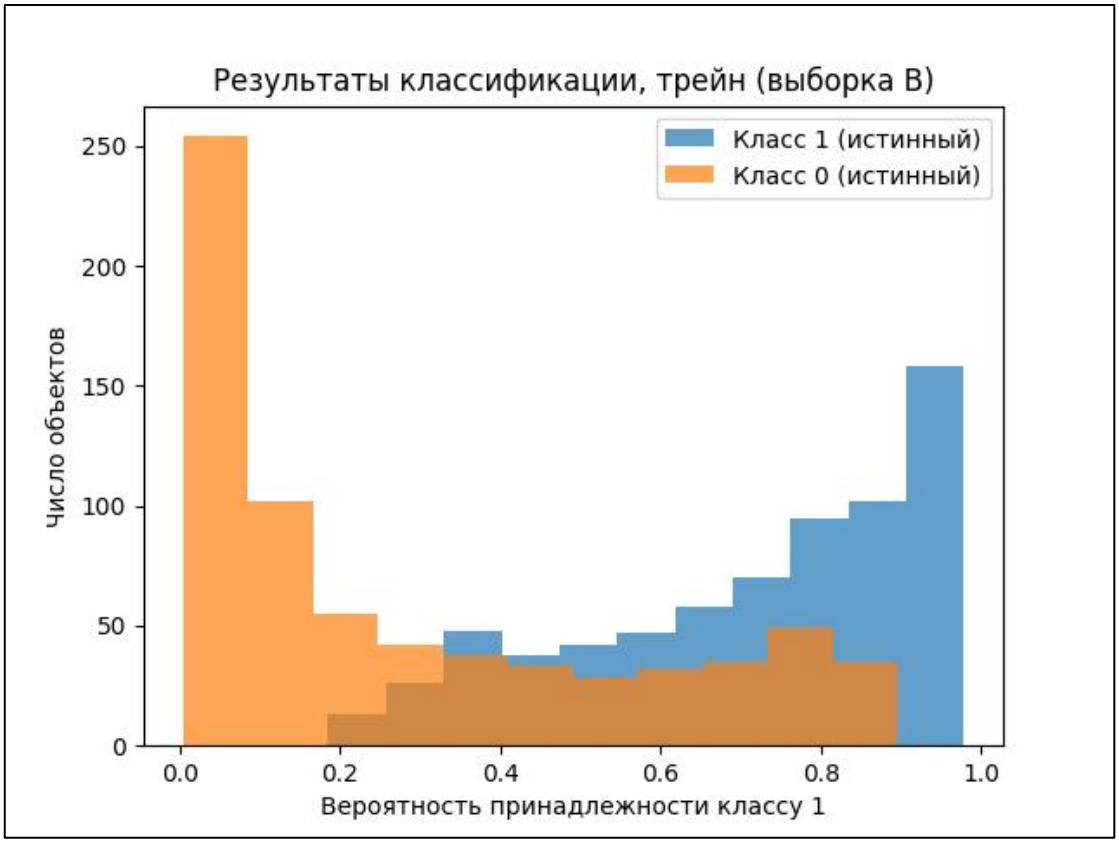

Рисунок 6 — Гистограмма, обучающая выборка B (нелинейно разделимые).

Выводы по гистограммам:

Выборка A (хорошо разделимые нормальные данные).

На тесте и на трейне класс 0 концентрируется у вероятности ≈ 0, класс 1 — у ≈ 1.

Пересечение в средней зоне небольшое.

Классификатор уверенно разделяет классы, большинство объектов попадает в крайние столбцы (0–0.1 и 0.9–1.0).

На трейне разделение ещё чётче, что соответствует высокой точности на хорошо разделимых данных.

Выборка Б (плохо разделимые нормальные данные).

Распределения сильно перекрываются.

Класс 0 даёт пик примерно в 0.35–0.45, класс 1 — в 0.55–0.60 (на трейне) и около 0.6–0.7 (на тесте).

В интервале примерно 0.3–0.7 много объектов обоих классов — при пороге 0.5 неизбежны ложные положительные и ложные отрицательные.

Это согласуется с более плотным пересечением облаков при сближенных средних и больших СКО: метрики (точность, чувствительность, специфичность) ниже, чем на выборке А.

Выборка В (нелинейно разделимые данные).

Часть объектов хорошо отделяется: у класса 0 выраженный пик у 0, у класса 1 — у 1

При этом в средней зоне (примерно 0.2–0.9) есть заметный «хвост» обоих классов — зона неопределённости и ошибок.

Логистическая регрессия задаёт линейную границу, а классы нелинейно разделимы (один внутри другого), поэтому полного разделения нет; для таких данных уместны нелинейные методы или ядра.

Таблицы с результатами оценки качества классификации

Выборка А (нормальное распределение, хорошо разделимые)

	Число объектов	Точность, %	Чувствительность, %	Специфичность, %
Train	1400	92.36	93.04	91.69
Test	600	93.17	94.84	91.38

Выборка Б (нормальное распределение, плохо разделимые)

	Число объектов	Точность, %	Чувствительность, %	Специфичность, %
Train	1400	61.86	61.10	62.61
Test	600	68.33	66.99	69.73

Выборка В (нелинейно разделимые)

	Число объектов	Точность, %	Чувствительность, %	Специфичность, %
Train	1400	78.14	81.40	74.89
Test	600	77.50	79.73	75.25

Вывод

В работе реализован классификатор на основе логистической регрессии (scikit-learn) с оптимизатором SAGA. Проведены эксперименты на трёх типах данных: хорошо разделимые нормальные (А), плохо разделимые нормальные (Б), нелинейно разделимые (В).

На выборке А модель даёт высокие точность, чувствительность и специфичность; гистограммы вероятностей для классов 0 и 1 слабо пересекаются. На выборке Б из-за сближения

средних и больших СКО метрики снижаются. На выборке B логистическая регрессия как линейный метод оказывается неэффективной — разделяющая граница линейная, а классы нелинейно разделимы (один внутри другого); для таких данных нужны нелинейные методы или ядра.

Чувствительность и специфичность рассчитаны вручную по формулам. Визуализация распределения вероятностей на выходе классификатора позволяет наглядно оценить уверенность модели и области ошибок.

Код программы:

```
"""
Лабораторная работа 2
"""
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
from sklearn.linear_model import LogisticRegression

FIGURES_DIR = Path(__file__).resolve().parent / "figures"
FIGURES_DIR.mkdir(exist_ok=True)


    import sys
    _root = Path(__file__).resolve().parent.parent
    if str(_root) not in sys.path:
        sys.path.insert(0, str(_root))
    import lab1.DataGenerator as dg

Nvar = 5
N = 1000
random_state = 42


def sensitivity_specificity(Y_true, Y_pred):
    Y_true = np.asarray(Y_true).ravel()
    Y_pred = np.asarray(Y_pred).ravel()
    Y_true = (Y_true != 0).astype(int)
    Y_pred = (Y_pred != 0).astype(int)
    TP = np.sum((Y_true == 1) & (Y_pred == 1))
    TN = np.sum((Y_true == 0) & (Y_pred == 0))
    FP = np.sum((Y_true == 0) & (Y_pred == 1))
    FN = np.sum((Y_true == 1) & (Y_pred == 0))
    sens = TP / (TP + FN) if (TP + FN) > 0 else 0.0
    spec = TN / (TN + FP) if (TN + FP) > 0 else 0.0
    return sens, spec
```

```
def split_70_30(X, Y):
    """Разбиение на обучающую (70%) и тестовую (30%) выборки."""
    np.random.seed(random_state)
    idx = np.random.permutation(len(Y))
    X, Y = X[idx], Y[idx]
    train_count = round(0.7 * len(Y))
    return X[:train_count], Y[:train_count], X[train_count:],
Y[train_count:]


def run_experiment(Xtrain, Ytrain, Xtest, Ytest, suffix, title_suffix):
    """
    Обучает логистическую регрессию (SAGA), считает метрики,
    строит гистограммы вероятностей для train и test.
    """
    clf = LogisticRegression(random_state=Nvar, solver="saga").fit(Xtrain,
Ytrain)

    Pred_train = clf.predict(Xtrain)
    Pred_test = clf.predict(Xtest)
    Pred_train_proba = clf.predict_proba(Xtrain)
    Pred_test_proba = clf.predict_proba(Xtest)

    acc_train = clf.score(Xtrain, Ytrain)
    acc_test = clf.score(Xtest, Ytest)
    acc_test_manual = np.sum(Pred_test == Ytest) / len(Ytest)

    sens_train, spec_train = sensitivity_specificity(Ytrain, Pred_train)
    sens_test, spec_test = sensitivity_specificity(Ytest, Pred_test)

    # Тест
    plt.figure()
    plt.hist(Pred_test_proba[Ytest, 1], bins="auto", alpha=0.7,
label="Класс 1 (истинный)", color="C0")
    plt.hist(Pred_test_proba[~Ytest, 1], bins="auto", alpha=0.7,
label="Класс 0 (истинный)", color="C1")
    plt.xlabel("Вероятность принадлежности классу 1")
    plt.ylabel("Число объектов")
    plt.title(f"Результаты классификации, тест ({title_suffix})")
    plt.legend()
    plt.savefig(FIGURES_DIR / f"lab2_hist_test_{suffix}.png")
    plt.close()

    # Трейн
    plt.figure()
    plt.hist(Pred_train_proba[Ytrain, 1], bins="auto", alpha=0.7,
label="Класс 1 (истинный)", color="C0")
    plt.hist(Pred_train_proba[~Ytrain, 1], bins="auto", alpha=0.7,
label="Класс 0 (истинный)", color="C1")
    plt.xlabel("Вероятность принадлежности классу 1")
    plt.ylabel("Число объектов")
    plt.title(f"Результаты классификации, трейн ({title_suffix})")
    plt.legend()
    plt.savefig(FIGURES_DIR / f"lab2_hist_train_{suffix}.png")
    plt.close()

    return {
        "train": (len(Ytrain), acc_train, sens_train, spec_train),
```

```
        "test": (len(Ytest), acc_test, sens_test, spec_test),
    }


def print_table(results, dataset_name):
    """Выводит таблицу метрик для одной выборки (Train/Test)."""
    print(f"\n--- {dataset_name} ---")
    print(f"{'':8} {'Число объектов':>14} {'Точность, %':>12}
{'Чувствительность, %':>20} {'Специфичность, %':>18}")
    print("-" * 78)
    for part in ("train", "test"):
        label = "Train" if part == "train" else "Test"
        n, acc, sens, spec = results[part]
        print(f"{label:8} {n:>14} {acc*100:>11.2f}% {sens*100:>19.2f}%
{spec*100:>17.2f}%")


# ---------- Выборка А: хорошо разделимые нормальные данные ----------
mu0_A = [0, 2, 3]
mu1_A = [3, 5, 1]
sigma0_A = [2, 1, 2]
sigma1_A = [1, 2, 1]
X_A, Y_A, _, _ = dg.norm_dataset([mu0_A, mu1_A], [sigma0_A, sigma1_A], N)
Xtrain_A, Ytrain_A, Xtest_A, Ytest_A = split_70_30(X_A, Y_A)
results_A = run_experiment(Xtrain_A, Ytrain_A, Xtest_A, Ytest_A, "A",
"выборка А")
print_table(results_A, "Выборка А (хорошо разделимые)")

# ---------- Выборка Б: плохо разделимые нормальные данные ----------
mu0_B = [1, 3, 2]
mu1_B = [2, 4, 3]
sigma0_B = [2.5, 2, 2.5]
sigma1_B = [2, 2.5, 2]
X_B, Y_B, _, _ = dg.norm_dataset([mu0_B, mu1_B], [sigma0_B, sigma1_B], N)
Xtrain_B, Ytrain_B, Xtest_B, Ytest_B = split_70_30(X_B, Y_B)
results_B = run_experiment(Xtrain_B, Ytrain_B, Xtest_B, Ytest_B, "B",
"выборка Б")
print_table(results_B, "Выборка Б (плохо разделимые)")

# ---------- Выборка В: нелинейно разделимые данные ----------
X_C, Y_C, _, _ = dg.nonlinear_dataset_5(N)
Xtrain_C, Ytrain_C, Xtest_C, Ytest_C = split_70_30(X_C, Y_C)
results_C = run_experiment(Xtrain_C, Ytrain_C, Xtest_C, Ytest_C, "C",
"выборка В")
print_table(results_C, "Выборка В (нелинейно разделимые)")

print(f"\nГистограммы сохранены в папке: {FIGURES_DIR}")
```