МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТк

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №1

Специальность ИИ26(з)

Выполнил
Т.В.
Данилюк,
студент группы ИИ26

Проверила
К.В. Андренко,
ст. преп. кафедры ИИТ,
«___k_____2026 г.

Брест 2026

Цель работы: Изучить принципы бинарной классификации и реализовать однослойную нейронную сеть (персептрон) для решения задачи классификации с использованием пороговой функции активации, а также исследовать процесс обучения модели с применением среднеквадратичной ошибки (MSE).

Код программы:

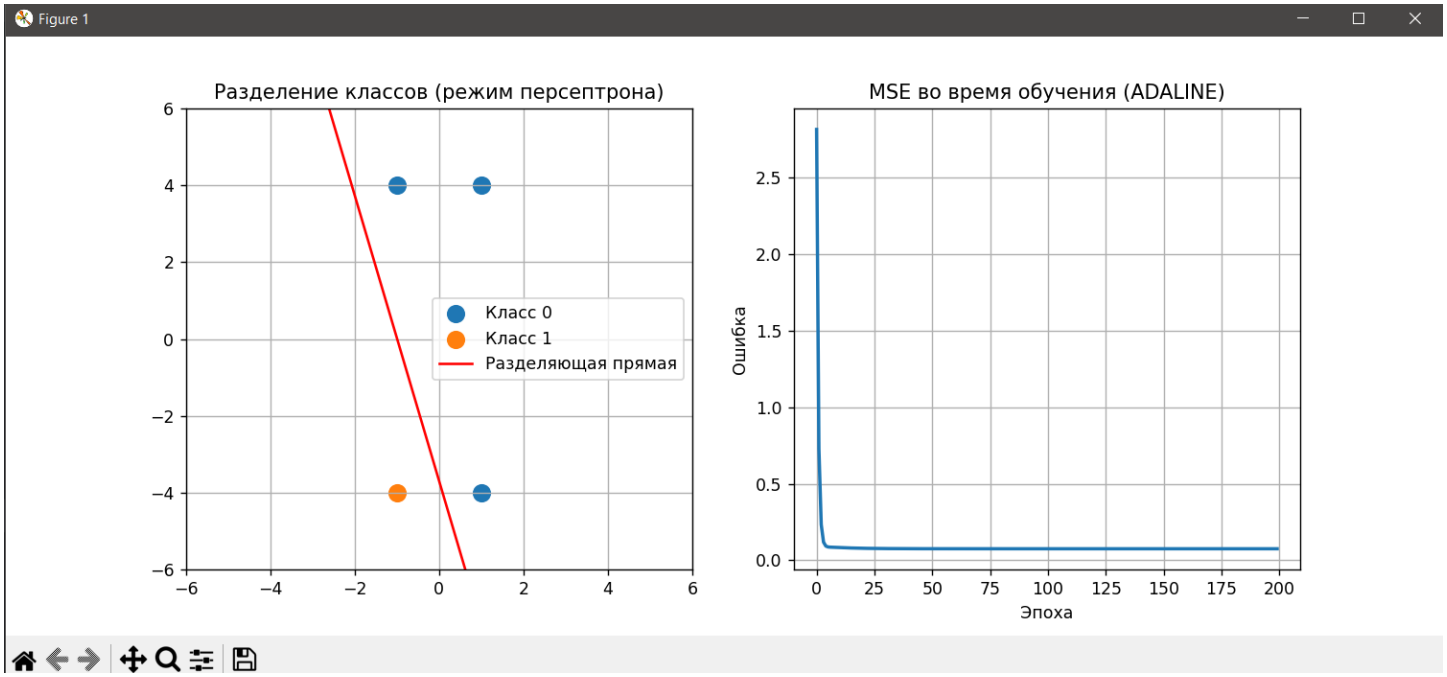
Содержимое в файле main.py

```
import numpy as np
import matplotlib.pyplot as plt


class PerceptronADALINE:
    def __init__(self, input_size, learning_rate=0.01):
        np.random.seed(42)
        self.w = np.random.uniform(-0.5, 0.5, input_size)
        self.w0 = np.random.uniform(-0.5, 0.5)
        self.lr = learning_rate

    # ===== ЛИНЕЙНЫЙ ВЫХОД (ADALINE) =====
    def net(self, X):
        return np.dot(X, self.w) + self.w0

    # ===== ПОРОГОВАЯ ФУНКЦИЯ (ПЕРСЕПТРОН) =====
    def step(self, S):
        return np.where(S >= 0.5, 1, 0)

    # ===== ЭТАП 1: ОБУЧЕНИЕ ADALINE =====
    def train_adaline(self, X, target, epochs=200):
        mse_history = []

        for epoch in range(epochs):
            errors = []

            for i in range(len(X)):
                S = self.net(X[i])          # линейный выход
                delta = target[i] - S       # ошибка ДО активации

                # дельта-правило ADALINE
                self.w += self.lr * delta * X[i]
                self.w0 += self.lr * delta

                errors.append(delta ** 2)

            mse_history.append(np.mean(errors))

        return mse_history

    # ===== ЭТАП 2: ОБУЧЕНИЕ ПЕРСЕПТРОНА =====
    def train_perceptron(self, X, target, epochs=50):
        for epoch in range(epochs):
            for i in range(len(X)):
                S = self.net(X[i])
                y = self.step(S)            # пороговый выход
```

```
            delta = target[i] - y      # ошибка ПОСЛЕ активации

            # правило персептрона
            self.w += self.lr * delta * X[i]
            self.w0 += self.lr * delta

    # ===== ЭТАП 3: РАБОТА ПЕРСЕПТРОНА =====
    def predict(self, X):
        S = self.net(X)
        return self.step(S)


# ===== ДАННЫЕ =====
X = np.array([[1, 4],
              [-1, 4],
              [1, -4],
              [-1, -4]])

target = np.array([0, 0, 0, 1])

# ===== ОБУЧЕНИЕ =====
p = PerceptronADALINE(input_size=2, learning_rate=0.01)

print("Этап 1 — обучение ADALINE")
mse = p.train_adaline(X, target, epochs=200)

print("Этап 2 — дообучение Персептрона")
p.train_perceptron(X, target, epochs=50)

# ===== ГРАФИК MSE =====
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 2)
plt.plot(mse, linewidth=2)
plt.title("MSE во время обучения (ADALINE)")
plt.xlabel("Эпоха")
plt.ylabel("Ошибка")
plt.grid(True)

# ===== ВИЗУАЛИЗАЦИЯ РАЗДЕЛЕНИЯ =====
plt.subplot(1, 2, 1)

plt.scatter(X[:3, 0], X[:3, 1], s=100, label='Класс 0')
plt.scatter(X[3:, 0], X[3:, 1], s=100, label='Класс 1')

x_line = np.linspace(-6, 6, 200)
y_line = (0.5 - p.w0 - p.w[0] * x_line) / p.w[1]
plt.plot(x_line, y_line, 'r', label='Разделяющая прямая')

plt.xlim(-6, 6)
plt.ylim(-6, 6)
plt.grid(True)
plt.legend()
plt.title("Разделение классов (режим персептрона)")

plt.show()
```

```
# ===== РЕЖИМ ТЕСТИРОВАНИЯ =====
print("\nРежим работы персептрона")
while True:
    user_input = input("Введите x1 x2 или 'q': ")
    if user_input.lower() == 'q':
        break

    try:
        x1, x2 = map(float, user_input.split())
        pred = p.predict(np.array([x1, x2]))
        print(f"Класс: {pred}")
    except:
        print("Ошибка ввода")
```

Рисунки с результатами работы программы




```
Режим работы персептрона
Введите x1 x2 или 'q': 1 4
Класс: 0
Введите x1 x2 или 'q': -1 4
Класс: 0
Введите x1 x2 или 'q': 1 -4
Класс: 0
Введите x1 x2 или 'q': -1 -4
Класс: 1
Введите x1 x2 или 'q': q

Process finished with exit code 0
```

Вывод: Изучил принципы бинарной классификации и реализовал однослойную нейронную сеть (персептрон) для решения задачи классификации с использованием пороговой функции активации, а также исследовал процесс обучения модели с применением среднеквадратичной ошибки (MSE).