Выполнил:

студент группы УВП-412

Рогов К.Д.

**Лабораторная работа № 4**

**Логистическая регрессия**

**Задание**

На основе данных файла ex2data1.txt решить задачу о нахождении вероятности неисправности двигателя в зависимости от его вибрации и неравномерности вращения.

Улучшить модель путём добавления нелинейного признака

**Описание кода**

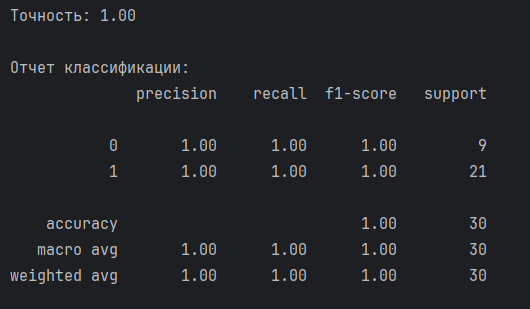
Данный код решает задачу бинарной классификации данных с использованием логистической регрессии и полиномиальных признаков для захвата нелинейных зависимостей между признаками.

1. Загрузка и подготовка данных:
   * Данные загружаются из файла ex2data1.txt, содержащего три колонки: Vibration (вибрация), Rotation (ротация) и Label (метка класса).
   * Данные делятся на признаки X (вибрация и ротация) и целевую переменную y (класс 0 или 1).
2. Визуализация исходных данных:
   * Используется scatter plot, чтобы отобразить точки данных двух классов на графике.
   * Класс 0 представлен красными кругами, а класс 1 – синими крестиками.
3. Подготовка данных для обучения:
   * Данные разделяются на обучающую (70%) и тестовую (30%) выборки.
   * Полиномиальные признаки второй степени добавляются с помощью PolynomialFeatures, чтобы учесть нелинейные зависимости между переменными.
4. Обучение модели логистической регрессии:
   * Логистическая регрессия обучается на полиномиальных признаках с максимальным числом итераций 1000 для гарантии сходимости.
5. Оценка модели:
   * На тестовой выборке производится предсказание и оценивается точность с использованием метрик accuracy\_score и classification\_report.
   * Точность модели составляет 1.00 (100%).
6. Визуализация границы принятия решений:
   * Создается сетка точек в диапазоне значений признаков.
   * Полиномиальные признаки сетки преобразуются, и для каждой точки предсказывается класс.
   * Граница решений отображается как разделение цветных областей на графике:
     + Область класса 0 (голубой цвет).
     + Область класса 1 (коричневый цвет).

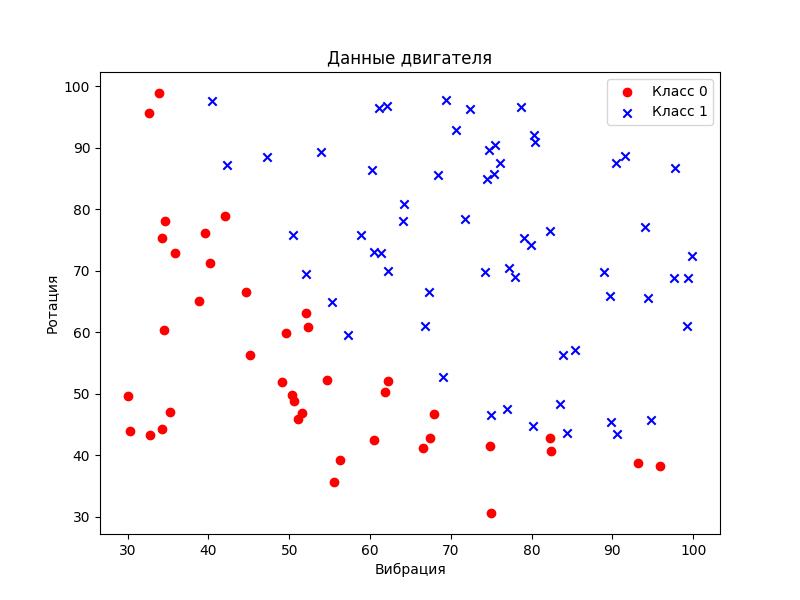
**Анализ результатов**

1. Точность модели:
   * Модель показывает идеальную точность на тестовых данных (1.00), что может указывать на:
     + Отличное разделение классов в исходных данных.
     + Наличие нелинейных зависимостей, которые эффективно учитываются полиномиальными признаками второй степени.

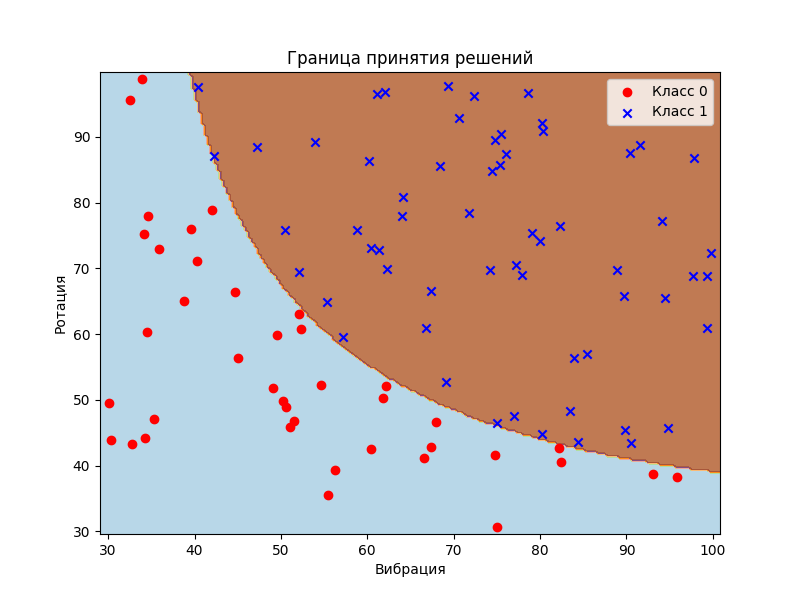
Отчет классификации подтверждает, что модель безошибочно классифицирует оба класса



1. Графики:
   * График исходных данных (categories.png):
     + Красные точки (класс 0) и синие крестики (класс 1) хорошо разделены.
     + Можно заметить, что граница между классами нелинейна, что объясняет необходимость использования полиномиальных признаков.



* + График границы принятия решений (libs\_decision\_boundary.png):
    - Граница решений имеет криволинейную форму, что подтверждает способность модели захватывать нелинейные зависимости.
    - Голубая область соответствует классу 0, а коричневая – классу 1.
    - Все точки из обучающего набора находятся в правильной области, что объясняет 100%-ную точность модели.



**Заключение**

Код успешно решает задачу бинарной классификации, используя логистическую регрессию с нелинейными признаками. Добавление полиномиальных признаков позволяет модели строить нелинейную границу принятия решений, что видно на итоговом графике.

Модель достигает идеальной точности на тестовой выборке. Однако такой результат также может указывать на переобучение, особенно если объем данных небольшой. Для проверки устойчивости модели можно использовать кросс-валидацию или добавить регуляризацию.

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import PolynomialFeatures  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import accuracy\_score, classification\_report  
  
# Загрузка данных из файла  
file\_path = 'ex2data1.txt' # Укажите путь к вашему файлу  
data = pd.read\_csv(file\_path, header=None, names=["Vibration", "Rotation", "Label"])  
  
# Разделение данных на признаки (X) и целевую переменную (y)  
X = data[["Vibration", "Rotation"]].values  
y = data["Label"].values  
  
# Визуализация исходных данных  
plt.figure(figsize=(8, 6))  
# Цикл для отображения каждой категории данных разными цветами и маркерами  
for label, color, marker in zip([0, 1], ['red', 'blue'], ['o', 'x']):  
 subset = data[data["Label"] == label]  
 plt.scatter(subset["Vibration"], subset["Rotation"], label=f"Класс {label}", c=color, marker=marker)  
plt.xlabel("Вибрация")  
plt.ylabel("Ротация")  
plt.legend()  
plt.title("Данные двигателя")  
plt.savefig("categories.png") # Сохранение графика  
  
# Разделение данных на обучающую и тестовую выборки  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)  
  
# Добавление полиномиальных признаков степени 2 для захвата нелинейных зависимостей  
poly = PolynomialFeatures(degree=2, include\_bias=False)  
X\_train\_poly = poly.fit\_transform(X\_train)  
X\_test\_poly = poly.transform(X\_test)  
  
# Обучение модели логистической регрессии с увеличенным количеством итераций для сходимости  
model = LogisticRegression(max\_iter=1000)  
model.fit(X\_train\_poly, y\_train)  
  
# Предсказание меток на тестовой выборке  
y\_pred = model.predict(X\_test\_poly)  
# Оценка точности модели  
accuracy = accuracy\_score(y\_test, y\_pred)  
  
print(f"Точность: {accuracy:.2f}")  
print("\nОтчет классификации:")  
print(classification\_report(y\_test, y\_pred))  
  
# Визуализация границы принятия решений  
# Определение диапазона значений для построения сетки  
x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1  
y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1  
# Создание сетки точек  
xx, yy = np.meshgrid(np.linspace(x\_min, x\_max, 200),  
 np.linspace(y\_min, y\_max, 200))  
# Преобразование сетки с помощью полиномиальных признаков  
Z = model.predict(poly.transform(np.c\_[xx.ravel(), yy.ravel()]))  
Z = Z.reshape(xx.shape)  
  
plt.figure(figsize=(8, 6))  
# Отображение области, соответствующей каждому классу  
plt.contourf(xx, yy, Z, alpha=0.8, cmap=plt.cm.Paired)  
# Повторное отображение исходных данных поверх границы решений  
for label, color, marker in zip([0, 1], ['red', 'blue'], ['o', 'x']):  
 subset = data[data["Label"] == label]  
 plt.scatter(subset["Vibration"], subset["Rotation"], label=f"Класс {label}", c=color, marker=marker)  
plt.xlabel("Вибрация")  
plt.ylabel("Ротация")  
plt.legend()  
plt.title("Граница принятия решений")  
plt.savefig("libs\_decision\_boundary.png") # Сохранение графика границы решений