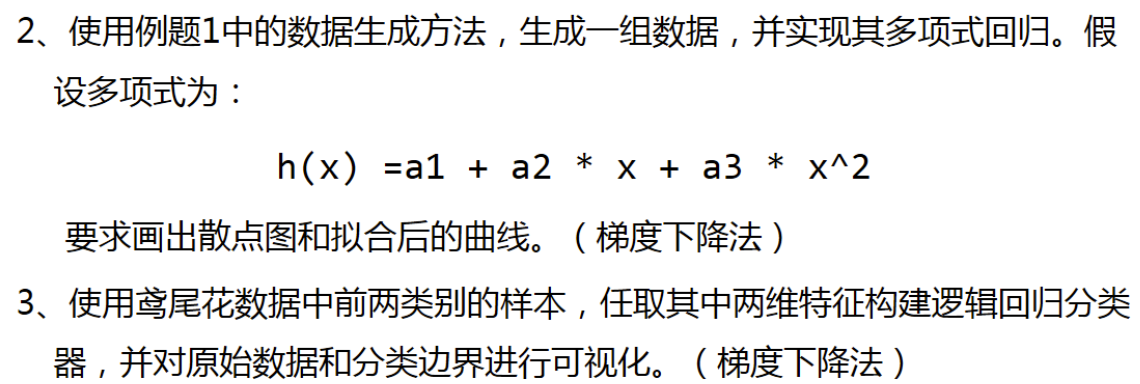
 

**智能信息系统综合实践**

**实验报告**

|  |  |
| --- | --- |
| **题 目：** | 回归分析 |
| **年 级：** | **2020级** |
| **专 业：** | **软件工程** |
| **姓 名：** | **庞晓宇** |

1. **题目（原题目）**



1. **解题步骤（思路+代码）**

1、在摩擦实验中，当电压为-285V时电机开启，之后的时间与速度关系如下： 时间（ms）,速度（m/s） 10,0 30,0.066 50,0.088 70,0.1539 90,0.1979 110,0.2419 130,0.2639 150,0.3079 170,0.3299 190,0.3738 210,0.3958 230,0.4398 250,0.4398 270,0.4618 290,0.4838 310,0.5498 330,0.5498 350,0.5718 370,0.5938 390,0.6158 410,0.6158 430,0.6377 450,0.6377 470,0.6597 490,0.6597

（1）编程实现拟合模型，对下列各函数模型进行曲线拟合，画出拟合图，求出函数表达式以及误差平方和，线性拟合，多项式二次拟合，多项式三次拟合，对数函数拟合，幂函数拟合。（最小二乘法）

In [ ]:

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

time **=** np**.**array([10, 30, 50, 70, 90, 110, 130, 150, 170, 190, 210, 230,

250, 270, 290, 310, 330, 350, 370, 390, 410, 430, 450, 470, 490])

speed **=** np**.**array([0, 0.066, 0.088, 0.1539, 0.1979, 0.2419, 0.2639, 0.3079, 0.3299, 0.3738, 0.3958, 0.4398,

0.4398, 0.4618, 0.4838, 0.5498, 0.5498, 0.5718, 0.5938, 0.6158, 0.6158, 0.6377, 0.6377, 0.6597, 0.6597])

In [ ]:

A **=** np**.**vstack([time, np**.**ones(len(time))])**.**T

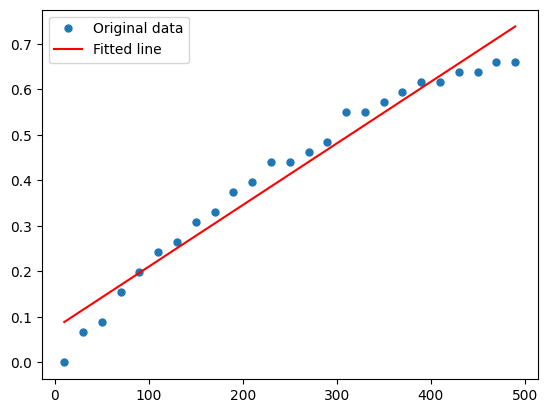
m, c **=** np**.**linalg**.**lstsq(A, speed, rcond**=None**)[0]

plt**.**plot(time, speed, 'o', label**=**'Original data', markersize**=**5)

plt**.**plot(time, m**\***time **+** c, 'r', label**=**'Fitted line')

plt**.**legend()

plt**.**show()



In [ ]:

fit\_func **=** f"y = {m:.4f}x + {c:.4f}"

print("函数表达式:", fit\_func)

error **=** np**.**linalg**.**lstsq(A, speed, rcond**=None**)[1][0]

print("误差平方和:", error)

函数表达式: y = 0.0014x + 0.0753

误差平方和: 0.039290752176922975

In [ ]:

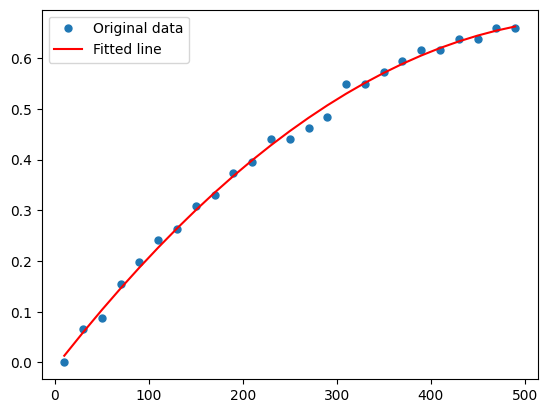
p **=** np**.**polyfit(time, speed, 2)

plt**.**plot(time, speed, 'o', label**=**'Original data', markersize**=**5)

plt**.**plot(time, np**.**polyval(p, time), 'r', label**=**'Fitted line')

plt**.**legend()

plt**.**show()



In [ ]:

fit\_func **=** f"y = {p[0]:.4f}x^2 + {p[1]:.4f}x + {p[2]:.4f}"

print("函数表达式:", fit\_func)

error **=** np**.**sum((np**.**polyval(p, time) **-** speed) **\*\*** 2)

print("误差平方和:", error)

函数表达式: y = -0.0000x^2 + 0.0024x + -0.0103

误差平方和: 0.003017346513415091

In [ ]:

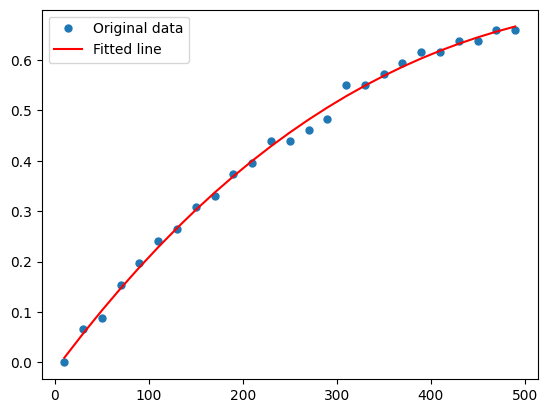
p **=** np**.**polyfit(time, speed, 3)

plt**.**plot(time, speed, 'o', label**=**'Original data', markersize**=**5)

plt**.**plot(time, np**.**polyval(p, time), 'r', label**=**'Fitted line')

plt**.**legend()

plt**.**show()



In [ ]:

fit\_func **=** f"y = {p[0]:.4f}x^3 + {p[1]:.4f}x^2 + {p[2]:.4f}x + {p[3]:.4f}"

print("函数表达式:", fit\_func)

error **=** np**.**sum((np**.**polyval(p, time) **-** speed) **\*\*** 2)

print("误差平方和:", error)

函数表达式: y = 0.0000x^3 + -0.0000x^2 + 0.0025x + -0.0155

误差平方和: 0.0029210730215668322

In [ ]:

*# 过滤掉速度为0的数据点*

non\_zero\_mask **=** speed **!=** 0

time\_non\_zero **=** time[non\_zero\_mask]

speed\_non\_zero **=** speed[non\_zero\_mask]

*# 取对数*

log\_time **=** np**.**log(time\_non\_zero)

log\_speed **=** np**.**log(speed\_non\_zero)

*# 线性拟合*

p **=** np**.**polyfit(log\_time, log\_speed, 1)

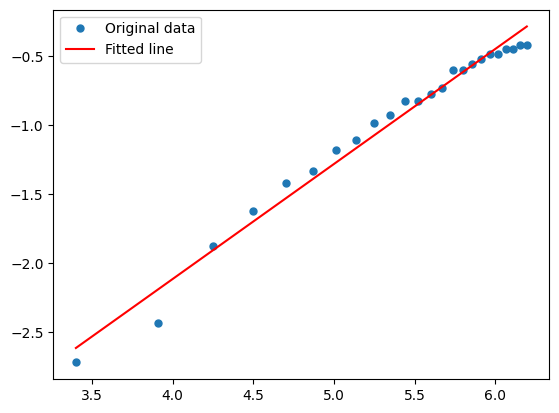
*# 绘制拟合图*

plt**.**plot(log\_time, log\_speed, 'o', label**=**'Original data', markersize**=**5)

plt**.**plot(log\_time, np**.**polyval(p, log\_time), 'r', label**=**'Fitted line')

plt**.**legend()

plt**.**show()



In [ ]:

*# 计算误差平方和和函数表达式*

fit\_func **=** f"y = {np**.**exp(p[1]):.4f}x^{p[0]:.4f}"

print("函数表达式:", fit\_func)

error **=** np**.**sum((np**.**exp(np**.**polyval(p, log\_time)) **-** speed\_non\_zero) **\*\*** 2)

print("误差平方和:", error)

函数表达式: y = 0.0043x^0.8339

误差平方和: 0.027266453311032852

（2）分析各种曲线的拟合结果与误差，选择最佳模型，画出速度与时间的拟合曲线及200ms时的加速度。

In [ ]:

*# 对比可以看出 函数表达式: y = 0.0000x^3 + -0.0000x^2 + 0.0025x + -0.0155*

*# 误差平方和: 0.0029210730215668322 是最佳的模型*

*# 定义三次多项式*

poly **=** np**.**poly1d([0.0000, **-**0.0000, 0.0025, **-**0.0155])

*# 定义时间范围*

time **=** np**.**arange(0, 0.5, 0.01)

*# 计算速度和加速度*

speed **=** poly(time)

acceleration **=** np**.**gradient(speed, time)

*# 绘制速度-时间曲线*

plt**.**plot(time, speed, 'r-', label**=**'speed')

plt**.**xlabel('time')

plt**.**ylabel('speed')

*# 绘制加速度-时间曲线*

plt**.**plot(time, acceleration, 'b-', label**=**'accelerated speed')

plt**.**xlabel('time')

plt**.**ylabel('accelerated speed')

*# 标记200ms时的速度和加速度*

plt**.**plot([0.2], [poly(0.2)], 'go')

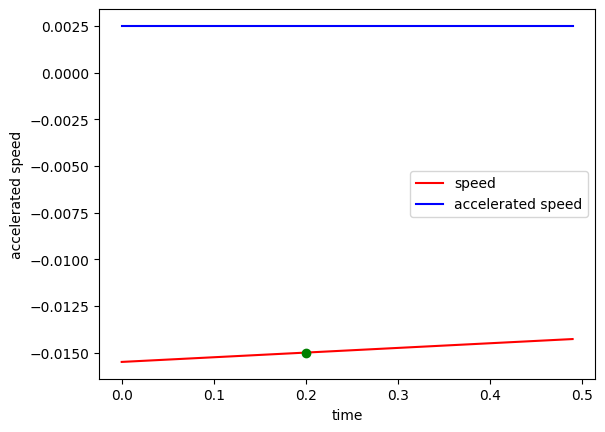
plt**.**plot([0.2], [np**.**gradient(poly(0.2), 0.01)], 'yo')

*# 添加图例*

plt**.**legend()

*# 显示图形*

plt**.**show()



2、使用例题1中的数据生成方法，生成一组数据，并实现其多项式回归。假设多项式为： h(x)=a1+a2∗x+a3∗x2ℎ(�)=�1+�2∗�+�3∗�2 要求画出散点图和拟合后的曲线。（梯度下降法）

In [ ]:

*# 生成数据*

np**.**random**.**seed(0)

n **=** 100

x **=** np**.**linspace(**-**1, 1, n)

y **=** 2 **+** 3 **\*** x **+** 4 **\*** x**\*\***2 **+** np**.**random**.**randn(n)**\***0.5

In [ ]:

*# 定义代价函数和梯度下降算法*

**def** compute\_cost(X, y, theta):

m **=** len(y)

J **=** 1**/**(2**\***m)**\***np**.**sum((X**.**dot(theta)**-**y)**\*\***2)

**return** J

**def** gradient\_descent(X, y, theta, alpha, num\_iters):

m **=** len(y)

J\_history **=** np**.**zeros(num\_iters)

**for** i **in** range(num\_iters):

theta **=** theta **-** alpha**/**m**\***X**.**T**.**dot(X**.**dot(theta)**-**y)

J\_history[i] **=** compute\_cost(X, y, theta)

**return** theta, J\_history

In [ ]:

*# 预处理数据*

X **=** np**.**vstack((np**.**ones(n), x, x**\*\***2))**.**T

*# 初始化参数*

theta **=** np**.**zeros(3)

*# 梯度下降*

alpha **=** 0.1

num\_iters **=** 1000

theta, J\_history **=** gradient\_descent(X, y, theta, alpha, num\_iters)

In [ ]:

*# 绘制散点图*

plt**.**scatter(x, y)

*# 绘制拟合曲线*

xx **=** np**.**linspace(**-**1, 1, 100)

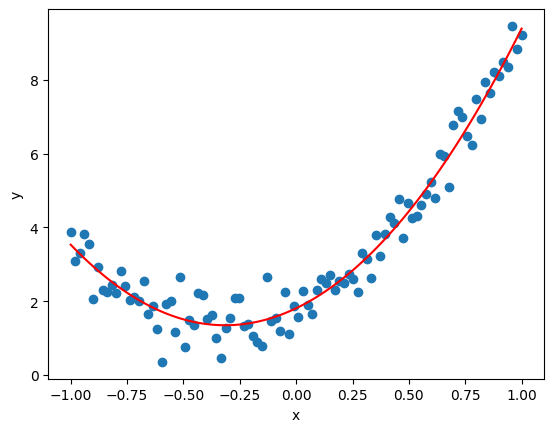
yy **=** theta[0] **+** theta[1]**\***xx **+** theta[2]**\***xx**\*\***2

plt**.**plot(xx, yy, 'r')

plt**.**xlabel('x')

plt**.**ylabel('y')

plt**.**show()



3、使用鸢尾花数据中前两类别的样本，任取其中两维特征构建逻辑回归分类器，并对原始数据和分类边界进行可视化。（梯度下降法）

In [ ]:

**from** sklearn.datasets **import** load\_iris

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

In [ ]:

*# 加载数据集*

iris **=** load\_iris()

*# 取出前两类别的样本并把标签变为-1和1*

X **=** iris**.**data[:100, :2]

y **=** iris**.**target[:100]

y[y**==**0] **=** **-**1

In [ ]:

*# 定义代价函数和梯度下降算法*

**def** sigmoid(z):

**return** 1 **/** (1 **+** np**.**exp(**-**z))

**def** compute\_cost(X, y, theta):

m **=** len(y)

h **=** sigmoid(X**.**dot(theta))

J **=** **-**1 **/** m **\*** (y**.**T**.**dot(np**.**log(h)) **+** (1 **-** y)**.**T**.**dot(np**.**log(1 **-** h)))

**return** J

**def** gradient\_descent(X, y, theta, alpha, num\_iters):

m **=** len(y)

J\_history **=** np**.**zeros(num\_iters)

**for** i **in** range(num\_iters):

theta **=** theta **-** alpha **/** m **\*** X**.**T**.**dot(sigmoid(X**.**dot(theta)) **-** y)

J\_history[i] **=** compute\_cost(X, y, theta)

**return** theta, J\_history

In [ ]:

*# 初始化参数*

m, n **=** X**.**shape

X **=** np**.**hstack((np**.**ones((m, 1)), X))

theta **=** np**.**zeros(n **+** 1)

*# 梯度下降*

alpha **=** 0.1

num\_iters **=** 1000

theta, J\_history **=** gradient\_descent(X, y, theta, alpha, num\_iters)

In [ ]:

*# 绘制原始数据*

plt**.**scatter(X[y **==** **-**1, 1], X[y **==** **-**1, 2], label**=**'class 0')

plt**.**scatter(X[y **==** 1, 1], X[y **==** 1, 2], label**=**'class 1')

*# 绘制分类边界*

xx **=** np**.**linspace(4, 7, 100)

yy **=** **-**(theta[0] **+** theta[1] **\*** xx) **/** theta[2]

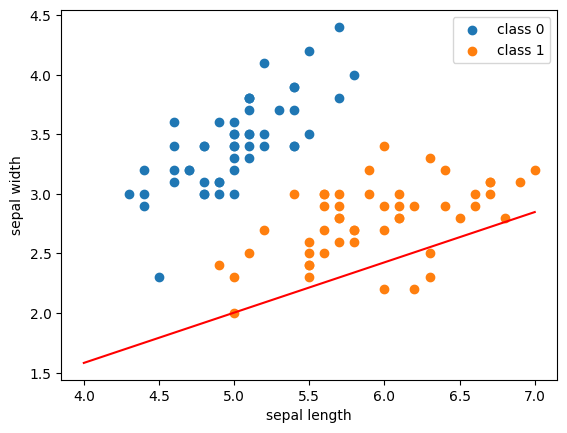
plt**.**plot(xx, yy, 'r')

plt**.**xlabel('sepal length')

plt**.**ylabel('sepal width')

plt**.**legend()

plt**.**show()



In [ ]:

**from** sklearn.datasets **import** load\_iris

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

*# 加载鸢尾花数据集*

iris **=** load\_iris()

*# 筛选出前两类别的样本*

X **=** iris**.**data[:100, :2]

y **=** iris**.**target[:100]

In [ ]:

*# 将数据集分为训练集和测试集，并使用逻辑回归分类器进行训练和预测。*

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.linear\_model **import** LogisticRegression

*# 将数据集分为训练集和测试集*

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(

X, y, test\_size**=**0.2, random\_state**=**42)

In [ ]:

*# 使用逻辑回归分类器进行训练和预测*

clf **=** LogisticRegression()

clf**.**fit(X\_train, y\_train)

y\_pred **=** clf**.**predict(X\_test)

In [ ]:

*# 使用 matplotlib 库将原始数据和分类边界进行可视化。*

*# 生成网格点坐标*

x\_min, x\_max **=** X[:, 0]**.**min() **-** 0.5, X[:, 0]**.**max() **+** 0.5

y\_min, y\_max **=** X[:, 1]**.**min() **-** 0.5, X[:, 1]**.**max() **+** 0.5

xx, yy **=** np**.**meshgrid(np**.**arange(x\_min, x\_max, 0.01),

np**.**arange(y\_min, y\_max, 0.01))

*# 绘制原始数据*

plt**.**scatter(X[:, 0], X[:, 1], c**=**y, cmap**=**plt**.**cm**.**Set1, edgecolor**=**'k')

plt**.**xlabel('Sepal length')

plt**.**ylabel('Sepal width')

*# 绘制分类边界*

Z **=** clf**.**predict(np**.**c\_[xx**.**ravel(), yy**.**ravel()])

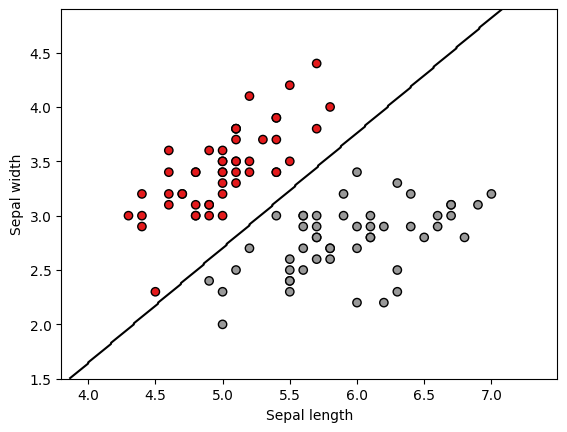
Z **=** Z**.**reshape(xx**.**shape)

plt**.**contour(xx, yy, Z, colors**=**'k', levels**=**[0], linestyles**=**['-'])

plt**.**show()

C:\Users\25810\AppData\Local\Temp\ipykernel\_9548\1569391782.py:17: UserWarning: No contour levels were found within the data range.

plt.contour(xx, yy, Z, colors='k', levels=[0], linestyles=['-'])



1. **总结（心得体会）**

回归分析是一种广泛应用于统计和机器学习领域的一种技术，可以用于分析变量之间的关系，建立模型并对因变量的值进行预测。在本次的实践中，我使用python语言中的matplotlib、numpy对一份数据使用多种函数进行了拟合，并绘制了拟合曲线计算了误差。最后基于鸢尾花数据集，使用sklearn中的逻辑回归模型对数据进行了分类。