

重庆科技学院学生实验报告

课程名称	机器学习		实验项目名称	线性回归	
开课学院及实验室		智能技术与工程学院		实验日期	9月25号
学生姓名	刘志贵	学号	2023440221	专业班级	智科 2023-02
指导教师	易军		实验成绩		

一、实验目的和要求

1. 理解简单线性回归和多元线性回归的基本原理与数学模型。
2. 掌握线性回归模型的构建、训练、评估方法。
3. 学会使用 Python 及相关库 (numpy、pandas、matplotlib、scikit-learn) 实现线性回归模型。
4. 能够分析实验结果，解释模型参数 (系数、截距) 的意义及评估指标 (如 MSE、 R^2) 的含义。

二、实验内容

1. 生成一组简单线性数据和一组多元线性数据。
2. 划分训练与测试集，用来训练简单线性模型和多元线性模型。
3. 在测试集上计算 MSE, R^2 , 并打印结果。
4. 对比估计参数和真实参数，分析偏差来源。

三、实验步骤与结果

步骤 1: 本次实验采用的是模拟数据用于实现简单线性回归和多元线性回归案例。

步骤 2: 简单线性回归 (一元):

先是生成了满足 $y=2x+3+\epsilon$ 的样本 (其中 ϵ 为随机噪声, 模拟真实数据的波动):

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

#生成特征x(100个样本, 范围0-10)
x = np.linspace(0, 10, 100).reshape(-1, 1)

print(x.shape)
```

[58] ✓ 0.0s

... (100, 1)

图 1 数据生成图

生成的数据范围是 0-10，并且生成了 100 组数据，然后进行生成标签和添加随机种子并且可视化的操作，如下图：

```
▷ ▾  
# 生成标签（加入误差-噪声）  
np.random.seed(42) #随机种子  
y = 2 * x + 3 + np.random.normal(0,1, size = x.shape) #噪声均值0，标准差1  
  
plt.rcParams['font.sans-serif'] = ['SimHei']  
plt.scatter(x, y, color = 'm', label = '样本数据')  
plt.xlabel('特征x')  
plt.ylabel('标签y')  
plt.title('简单线性回归分布')  
plt.legend()  
plt.show()  
[21] ✓ 0.0s
```

图 2 添加标签并可视化代码图

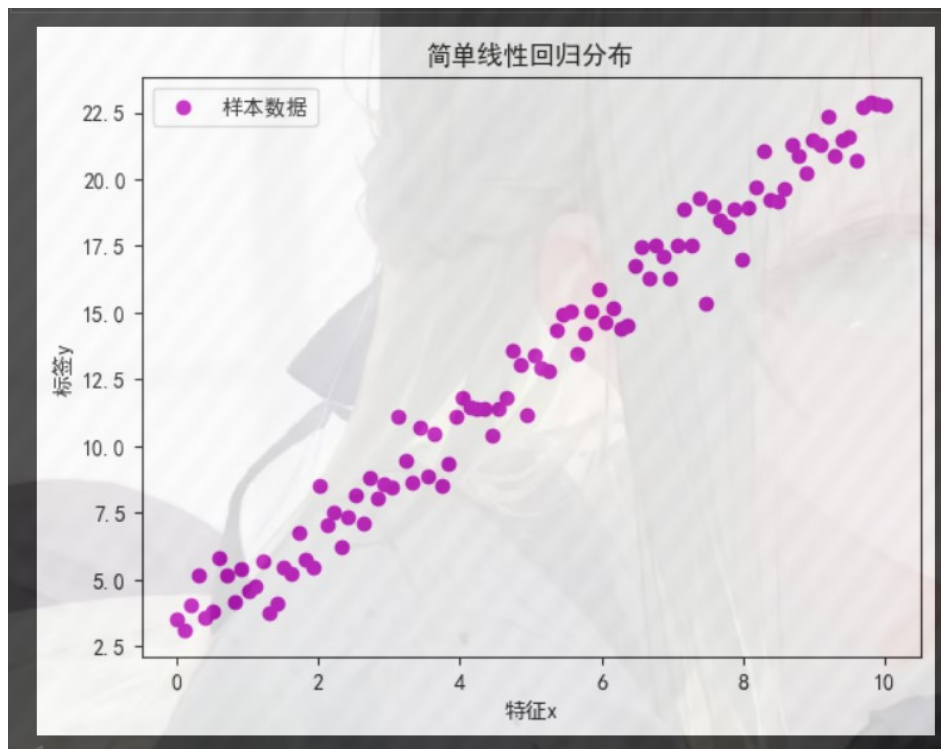


图 3 简单线性回归数据可视化图

数据生成完后就是模型构建与训练了，如下图是使用 scikit-learn 的 LinearRegression 类构建模型，并拟合数据代码图像：

```
▷ ▾  
# 模型构建与训练  
from sklearn.linear_model import LinearRegression  
from sklearn.model_selection import train_test_split  
  
#划分训练集(80%)和测试集(20%)  
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 42)  
print(x_train.shape, x_test.shape)  
[22] ✓ 0.0s  
... (80, 1) (20, 1)
```

图 4 模型构建与划分数据集代码图

将模拟数据进行划分，得到训练数据和测试数据，然后开始初始化训练模型，并且输出训练后得到的参数 w 和 b ，如下图：



```
# 初始化model
model = LinearRegression()
# 训练(拟合)
model.fit(x_train, y_train)

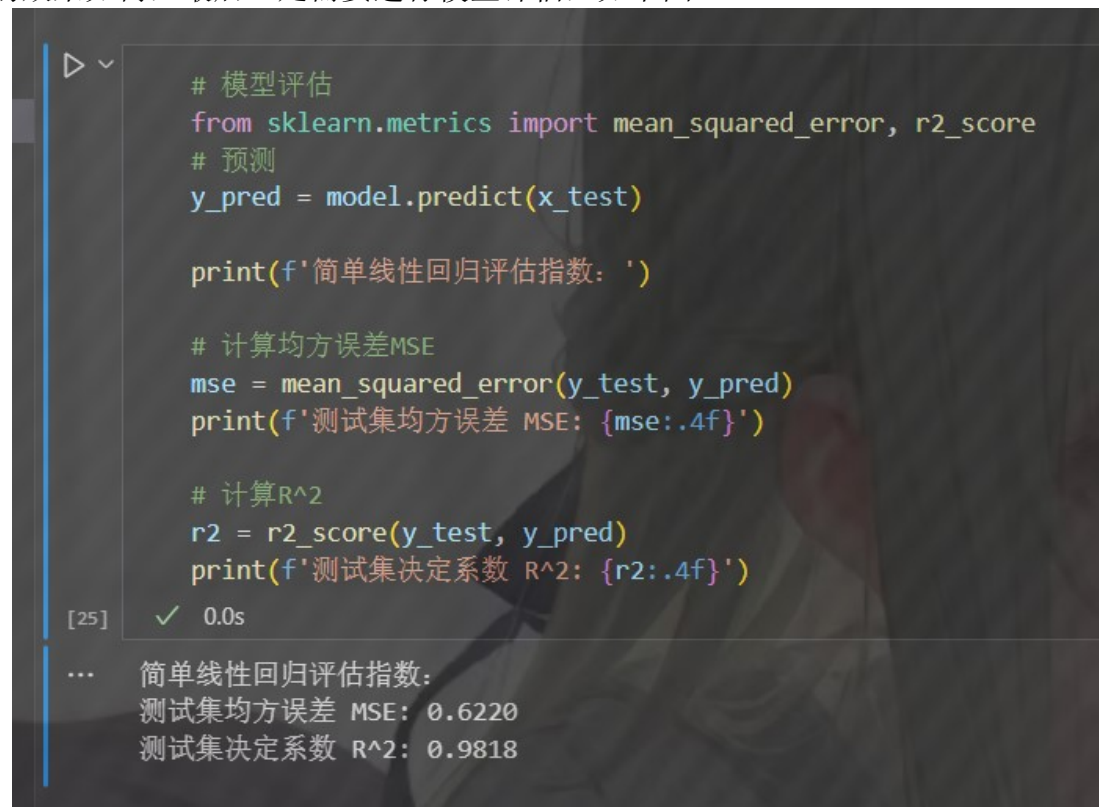
# 输出训练得到的参数
print(f'简单线性回归参数: ')
print(f'系数 w: {model.coef_[0][0]:.4f}') # 真实 w = 2
print(f'截距 b: {model.intercept_[0]:.4f}') # 真实 b = 3
```

[24] ✓ 0.0s

... 简单线性回归参数:
系数 w: 2.0121
截距 b: 2.8198

图 5 训练模型图

可以看到得到的参数虽然和真实参数有一点差距，但也是很接近的，为了看一下模型的效果如何，最后还是需要进行模型评估，如下图：



```
# 模型评估
from sklearn.metrics import mean_squared_error, r2_score
# 预测
y_pred = model.predict(x_test)

print(f'简单线性回归评估指数: ')

# 计算均方误差MSE
mse = mean_squared_error(y_test, y_pred)
print(f'测试集均方误差 MSE: {mse:.4f}')

# 计算R^2
r2 = r2_score(y_test, y_pred)
print(f'测试集决定系数 R^2: {r2:.4f}')
```

[25] ✓ 0.0s

... 简单线性回归评估指数:
测试集均方误差 MSE: 0.6220
测试集决定系数 R^2: 0.9818

图 6 简单线性回归模型图

评估采用了 MSE 和 R^2 的方法，结果比较理想，然后将拟合结果可视化，如下图所示：

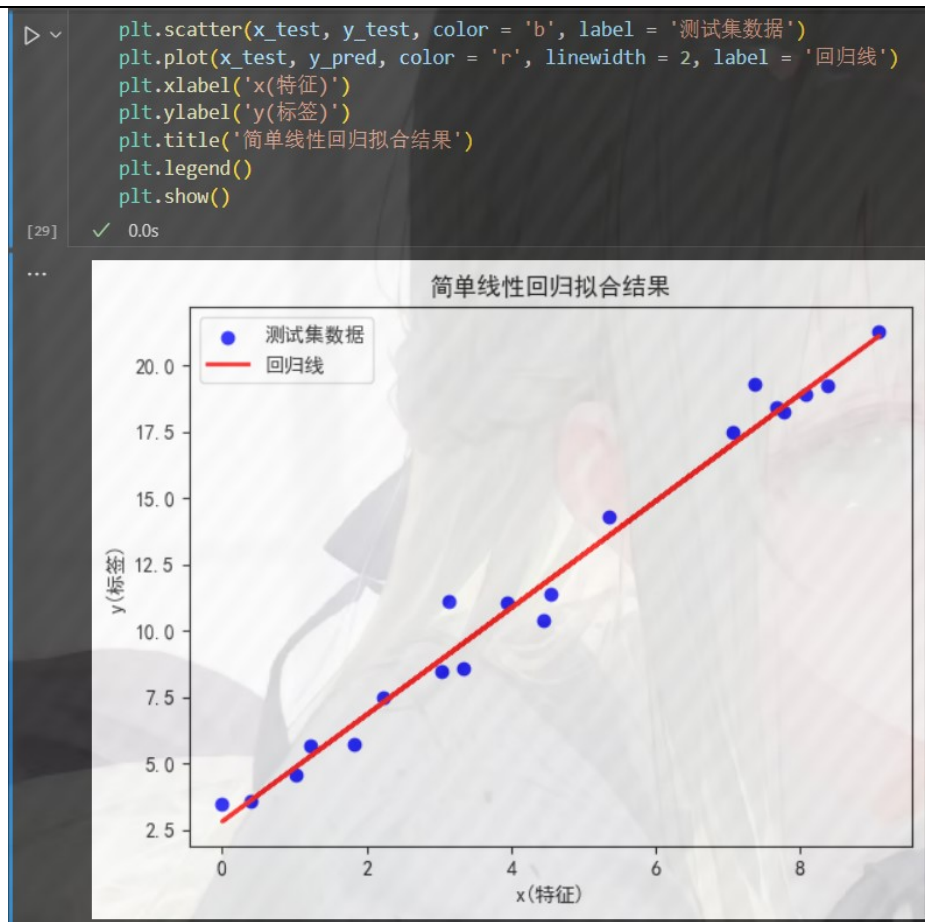


图 7 简单线性回归拟合结果图

通过图片能够清晰的感受到拟合效果良好。

步骤 3：多元线性回归

先是生成模型 $y = 1.5x_1 + 0.8x_2 - 2.3x_3 + 5 + \epsilon$ 的线性回归数据：

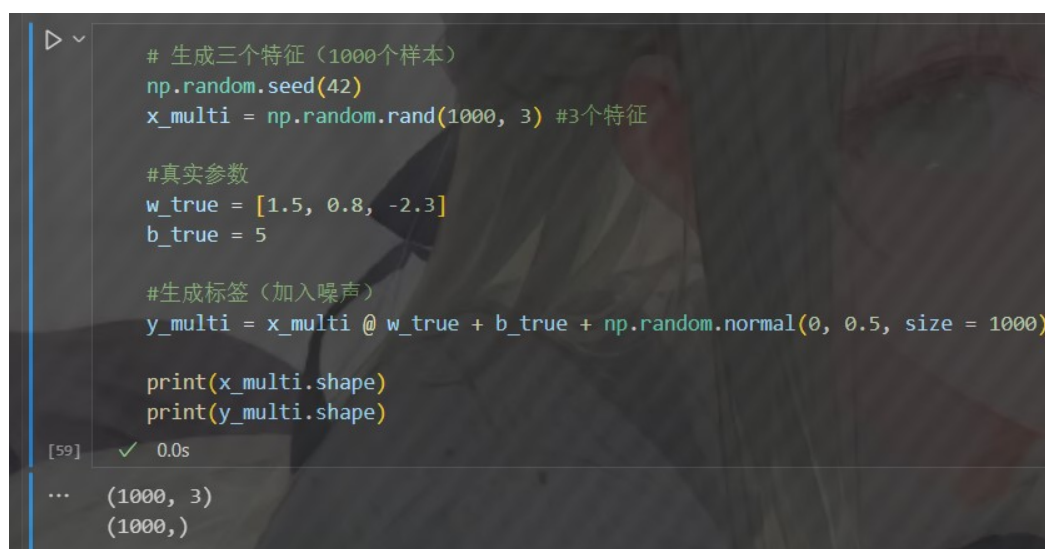


图 8 多元线性回归模型数据生成图

数据生成 3 组 1000 个数据，然后生成标签 y_{multi} ，然后利用模拟数据训练多元线性回归模型，最后打印训练出来的 w 和 b 参数，如下图：

```
x_train_m, x_test_m, y_train_m, y_test_m = train_test_split(x_multi, y_multi, test_size = 0.4, random_state = 42)

#初始化并训练多元线性回归模型
model_m = LinearRegression()
model_m.fit(x_train_m, y_train_m)

print(f'多元线性回归参数: ')
print(f'系数 w: {model_m.coef_} (真实值: {w_true})')
print(f'截距 b: {model_m.intercept_:.4f} (真实值: {b_true})')

[55] ✓ 0.0s
```

... 多元线性回归参数:
系数 w: [1.47549404 0.80480017 -2.41054863] (真实值: [1.5, 0.8, -2.3])
截距 b: 5.0689 (真实值: 5)

图 9 多元线性模型训练与参数打印图

可以看到训练后的模型打印的参数与真实参数同样有一点点差距，但并不能说明什么，最后进行模型评估看一下模型训练效果，如下图：

```
# 评估
y_pred_m = model_m.predict(x_test_m)

mse_m = mean_squared_error(y_test_m, y_pred_m)
r2_m = r2_score(y_test_m, y_pred_m)

print(f'多元线性回归评估指数: ')
print(f'测试集均方误差 MSE: {mse_m:.4f}')
print(f'测试集决定系数 R^2: {r2_m:.4f}')

[56] ✓ 0.0s
```

... 多元线性回归评估指数:
测试集均方误差 MSE: 0.2374
测试集决定系数 R^2: 0.7306

图 10 多元线性回归评估图

四、实验结果

一元线性回归中的模型训练的比较好，参数也很接近真实值，模拟数据生成了 100 组训练集划分 80%，测试集 20%，评估中 R^2 取得了 0.98 的拟合度。

在多元线性回归中，我开始同样采用了 100 组数据和划分 80% 的训练集模拟数据，但发现评估的时候 R^2 只有 0.47 的拟合度，效果并不是很好，最后我生成 1000 组数据， R^2 拟合度变成了 50 多，然后将测试集划分到模拟数据的 40% 会发现拟合效果大幅度提高，最后 R^2 保持在 73%，了解到这是因为训练集足够，且测试集也划分足够后就会减少生成的数据恰好偏离真实值的误差，并且能够达到较好的效果。

五、模型评价

1. 简单线性：

训练参数：

系数 w : 2.0121

截距 b : 2.8198

详情见实验步骤中的图 5。

评估指数：

测试集均方误差 MSE: 0.6220

测试集决定系数 R^2 : 0.9818

2. 多元线性回归：

训练参数：

系数 w : 2.0121

截距 b : 2.8198

详情见实验步骤中的图 5。

评估指数：

测试集均方误差 MSE: 0.6220

测试集决定系数 R^2 : 0.9818

附件

```
# 线性回归实验完整代码

# 简单线性回归
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

#生成特征 x(100 个样本，范围 0-10)
x = np.linspace(0, 10, 100).reshape(-1, 1)

print(x.shape)

# 生成标签（加入误差-噪声）
np.random.seed(42) #随机种子
y = 2 * x + 3 + np.random.normal(0, 1, size = x.shape) #噪声均值 0，标准差 1

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.scatter(x, y, color = 'm', label = '样本数据')
plt.xlabel('特征 x')
plt.ylabel('标签 y')
plt.title('简单线性回归分布')
```

```

plt.legend()
plt.show()

# 模型构建与训练
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

#划分训练集(80%)和测试集(20%)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 42)
print(x_train.shape, x_test.shape)

# 初始化 model
model = LinearRegression()
# 训练(拟合)
model.fit(x_train, y_train)

# 输出训练得到的参数
print(f'简单线性回归参数: ')
print(f'系数 w: {model.coef_[0][0]:.4f}') # 真实 w = 2
print(f'截距 b: {model.intercept_[0]:.4f}') # 真实 b = 3

# 模型评估
from sklearn.metrics import mean_squared_error, r2_score
# 预测
y_pred = model.predict(x_test)

print(f'简单线性回归评估指数: ')

# 计算均方误差 MSE
mse = mean_squared_error(y_test, y_pred)
print(f'测试集均方误差 MSE: {mse:.4f}')

# 计算 R^2
r2 = r2_score(y_test, y_pred)
print(f'测试集决定系数 R^2: {r2:.4f}')

plt.scatter(x_test, y_test, color = 'b', label = '测试集数据')
plt.plot(x_test, y_pred, color = 'r', linewidth = 2, label = '回归线')
plt.xlabel('x(特征)')
plt.ylabel('y(标签)')
plt.title('简单线性回归拟合结果')
plt.legend()
plt.show()

# 多元线性回归
# 生成三个特征 (1000 个样本)
np.random.seed(42)
x_multi = np.random.rand(1000, 3) #3 个特征

#真实参数
w_true = [1.5, 0.8, -2.3]
b_true = 5

```

```

#生成标签（加入噪声）
y_multi = x_multi @ w_true + b_true + np.random.normal(0, 0.5, size = 1000)

print(x_multi.shape)
print(y_multi.shape)

x_train_m, x_test_m, y_train_m, y_test_m = train_test_split(x_multi, y_multi, test_size = 0.4,
random_state = 42)

#初始化并训练多元线性回归模型
model_m = LinearRegression()
model_m.fit(x_train_m, y_train_m)

print(f'多元线性回归参数: ')
print(f'系数 w: {model_m.coef_} (真实值: {w_true})')
print(f'截距 b: {model_m.intercept_:.4f} (真实值: {b_true})')

# 评估
y_pred_m = model_m.predict(x_test_m)

mse_m = mean_squared_error(y_test_m, y_pred_m)
r2_m = r2_score(y_test_m, y_pred_m)

print(f'多元线性回归评估指数: ')
print(f'测试集均方误差 MSE: {mse_m:.4f}')
print(f'测试集决定系数 R^2: {r2_m:.4f}')

```