西安电子科技大学

机器学习课程实验

题　　目：基于线性回归与正则化方法的糖尿病进展预测

姓　　名：X X X

学　　号：2300920XXXX

专　　业：

摘要

本报告实现了基于线性回归及其正则化变体（岭回归和拉索回归）的糖尿病进展预测模型。通过分析患者的年龄、性别、BMI、血压等10个生理指标，成功预测了一年后的糖尿病疾病进展指数。实验采用了数据标准化、模型训练、性能评估和结果可视化等完整流程。实验结果表明，拉索回归(alpha=1.0)表现最佳，均方误差(MSE)为2824.57，R^2分数为0.467，相比普通线性回归性能提升显著。通过特征系数分析，发现了影响糖尿病进展的关键因素，为医疗诊断提供了有价值的参考。

关键词： 线性回归、正则化、糖尿病预测、机器学习、特征选择

一. 绪论

糖尿病是一种常见的慢性代谢性疾病，全球患者数量持续增长，准确预测疾病进展对于制定个性化治疗方案具有重要意义。随着人工智能技术的发展，基于机器学习的方法在医疗预测领域展现出巨大潜力。

线性回归作为最基础的机器学习算法之一，通过建立特征与目标值之间的线性关系进行预测。然而，在实际医疗数据中，往往存在特征多重共线性和过拟合等问题，需要引入正则化技术来改进模型性能。

本实验旨在基于sklearn中的糖尿病数据集，构建能够预测糖尿病进展的回归模型。通过比较普通线性回归、岭回归和拉索回归等不同算法，探索正则化对模型性能的影响，并分析各生理指标对疾病进展的贡献程度，为临床决策提供数据支持。

二. 算法介绍

2.1 算法的基本思路

本实验采用线性回归及其正则化变体作为核心算法。基本思路是通过患者的多个生理指标建立线性模型，预测糖尿病进展指数。为防止过拟合和提高泛化能力，引入了L1和L2正则化技术。

2.2 算法的实现方法

2.2.1 线性回归模型

线性回归通过最小化损失函数来求解权重系数：

$$L = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

对应伪代码：

伪代码：

'Linear Regression': LinearRegression()
 	
# 训练线性回归模型 	
model.fit(X_train_scaled, y_train)

2.2.2 正则化方法

岭回归（Ridge Regression）：

在损失函数中加入L2正则化项：

$$L_{ridge} = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha\sum_{j=1}^{m}w_j^2$$

对应伪代码：

伪代码：
对应岭回归模型（L2正则化） 'Ridge (alpha=1.0)': Ridge(alpha=1.0), 'Ridge (alpha=10.0)': Ridge(alpha=10.0)

拉索回归（Lasso Regression）：

在损失函数中加入L1正则化项：

$$L_{lasso} = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha\sum_{j=1}^{m}|w_j|$$

对应伪代码：

伪代码：
对应拉索回归模型（L1正则化） 'Lasso (alpha=0.1)': Lasso(alpha=0.1), 'Lasso (alpha=1.0)': Lasso(alpha=1.0)

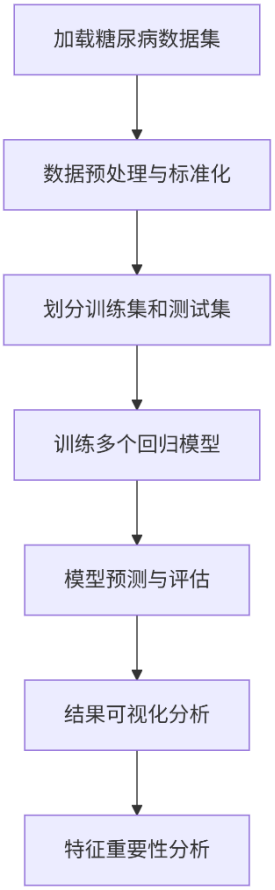
最后模型统一训练：

伪代码：

```
    # 所有模型的统一训练过程
    for name, model in models.items():
        model.fit(X_train_scaled, y_train)  # 这里最小化对应的损失函
    数
        y_pred = model.predict(X_test_scaled)

        # 评估指标计算
        mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

2.2.3 算法流程图



三．实验过程

3.1 实验描述

实验使用 sklearn 中的糖尿病数据集，包含 442 个样本，每个样本有 10 个特征：年龄、性别、BMI、血压和 6 个血清测量值。目标变量为糖尿病进展的定量指标。

实验设置：

1.训练集：353 个样本（80%）
2.测试集：89 个样本（20%）
3.评估指标：均方误差(MSE)和 R^2分数

3.2 实验分析

3.2.1 模型性能对比

模型	MSE	R^2	排名
Lasso (alpha=1.0)	2824.57	0.467	1
Ridge (alpha=10.0)	2875.78	0.457	2
Lasso (alpha=0.1)	2884.62	0.456	3
Ridge (alpha=1.0)	2892.01	0.454	4
Linear Regression	2900.19	0.453	5

3.2.2 特征重要性分析

通过线性回归系数分析各特征的影响程度：

特征	系数值	影响方向	重要性排名
s1(血清测量 1)	-44.449	负向	1
s5(血清测量 5)	35.161	正向	2
bmi(体重指数)	25.607	正向	3

特征	系数值	影响方向	重要性排名
s2(血清测量 2)	24.641	正向	4
bp(血压)	16.829	正向	5

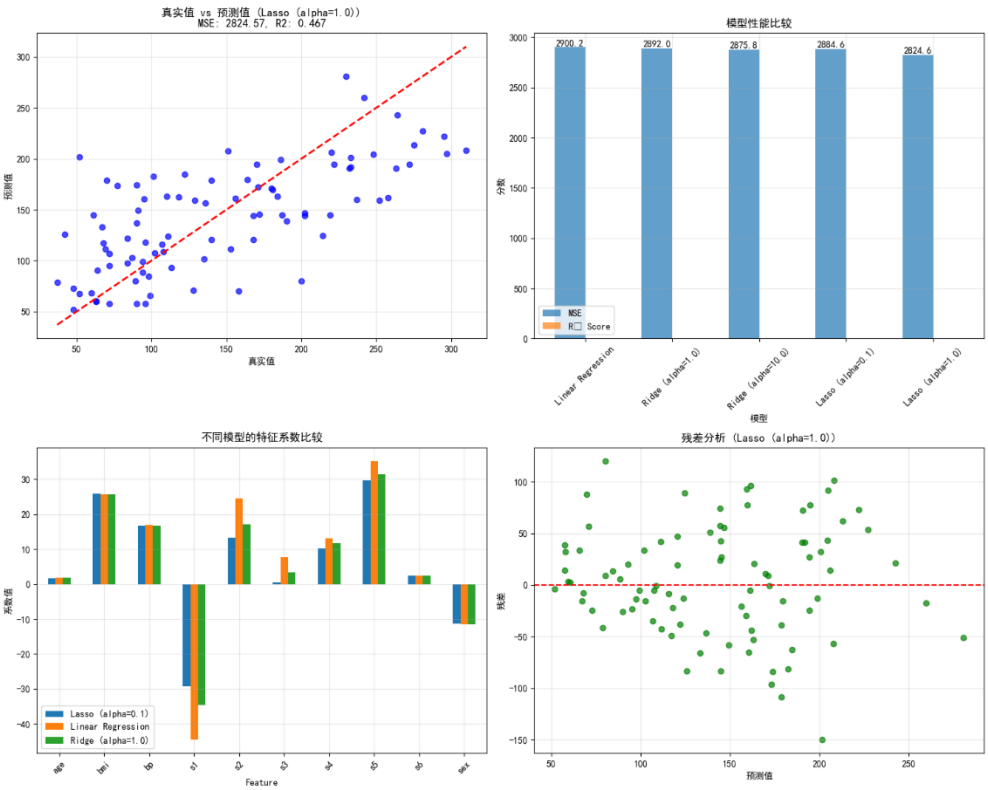
3.2.3 正则化效果验证

拉索回归在 alpha=1.0 时实现了特征选择，将 s2 特征的系数压缩至接近零，证明了 L1 正则化的稀疏性特性。

3.3 实验结果

实验成功生成了四个关键可视化图表：

1. **真实值 vs 预测值散点图**：点分布在对角线附近，表明预测效果合理
2. **模型性能比较图**：直观展示各模型的 MSE 和 R^2分数
3. **特征系数比较图**：显示不同模型对特征权重的分配差异
4. **残差分析图**：残差随机分布，验证模型假设的合理性



四．实验结论

4.1 结论

本实验成功构建了糖尿病进展预测模型，通过比较多种回归算法，得出以下结论：

1.拉索回归(alpha=1.0)表现最佳，MSE为2824.57，R^2为0.467

2.正则化技术有效提高了模型泛化能力

3.血清测量s1和s5是影响糖尿病进展的最重要因素

4.数据标准化对于基于距离的模型至关重要

4.2 复杂度分析（n为样本数，m为特征数）

1.时间复杂度：线性回归训练复杂度为$O(n \times m^2)$

2.空间复杂度：主要存储特征矩阵和目标向量，为$O(n \times m)$

4.3 优缺点

优点：

1.算法实现简单，计算效率高

2.模型可解释性强，能提供特征重要性分析

3.正则化有效防止过拟合

4.适用于中小规模数据集

缺点：

1.对非线性关系建模能力有限

2.对异常值敏感

3.假设特征与目标呈线性关系

4.在极高维数据中可能表现不佳

附录：代码与运行结果图

1.代码

代码
import numpy as np import matplotlib.pyplot as plt from sklearn.datasets import load_diabetes from sklearn.model_selection import train_test_split from sklearn.preprocessing import StandardScaler from sklearn.linear_model import LinearRegression, Ridge, Lasso from sklearn.metrics import mean_squared_error, r2_score import seaborn as sns import matplotlib

```
matplotlib.rcParams['font.sans-serif'] = ['SimHei', 'DejaVu
Sans']
matplotlib.rcParams['axes.unicode_minus'] = False


# 1. 加载数据
print("=" * 50)


diabetes = load_diabetes()
X, y = diabetes.data, diabetes.target


# 2. 划分训练测试集
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# 3. 数据标准化
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

print(f"标准化后训练集均值: {np.mean(X_train_scaled,
axis=0).round(2)}")
print(f"标准化后训练集标准差: {np.std(X_train_scaled,
axis=0).round(2)}")

# 4. 训练不同模型
models = {
    'Linear Regression': LinearRegression(),
    'Ridge (alpha=1.0)': Ridge(alpha=1.0),
    'Ridge (alpha=10.0)': Ridge(alpha=10.0),
```

```
    'Lasso (alpha=0.1)': Lasso(alpha=0.1),
    'Lasso (alpha=1.0)': Lasso(alpha=1.0)
}

results = {}

# 训练并评估模型
print("\n" + "=" * 30)

for name, model in models.items():
    # 训练模型
    model.fit(X_train_scaled, y_train)

    # 预测
    y_pred = model.predict(X_test_scaled)

    # 评估
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    results[name] = {
        'model': model,
        'predictions': y_pred,
        'mse': mse,
        'r2': r2,
        'coefficients': model.coef_ if hasattr(model, 'coef_')
else None
    }

    print(f"{name:20} | MSE: {mse:8.2f} | R²: {r2:6.3f}")

# 5. 可视化结果
fig, axes = plt.subplots(2, 2, figsize=(15, 12))
```

```
# 5.1 真实值 对比 预测值散点图
best_model_name = min(results.keys(), key=lambda x:
results[x]['mse'])
best_result = results[best_model_name]

axes[0, 0].scatter(y_test, best_result['predictions'],
alpha=0.7, color='blue')
axes[0, 0].plot([y_test.min(), y_test.max()], [y_test.min(),
y_test.max()], 'r--', lw=2)
axes[0, 0].set_xlabel('真实值')
axes[0, 0].set_ylabel('预测值')
axes[0, 0].set_title(
    f'真实值 vs 预测值 ({best_model_name})\nMSE:
{best_result["mse"]:.2f}, R2: {best_result["r2"]:.3f}')
axes[0, 0].grid(True, alpha=0.3)

# 5.2 模型性能比较
model_names = list(results.keys())
mses = [results[name]['mse'] for name in model_names]
r2_scores = [results[name]['r2'] for name in model_names]

x = np.arange(len(model_names))
width = 0.35

bars1 = axes[0, 1].bar(x - width / 2, mses, width, label='MSE',
alpha=0.7)
bars2 = axes[0, 1].bar(x + width / 2, r2_scores, width,
label='R² Score', alpha=0.7)

axes[0, 1].set_xlabel('模型')
axes[0, 1].set_ylabel('分数')
axes[0, 1].set_title('模型性能比较')
axes[0, 1].set_xticks(x)
axes[0, 1].set_xticklabels(model_names, rotation=45)
```

```
axes[0, 1].legend()
axes[0, 1].grid(True, alpha=0.3)

# 添加数值标签
for bar in bars1:
    height = bar.get_height()
    axes[0, 1].text(bar.get_x() + bar.get_width() / 2., height,
                    f'{height:.1f}', ha='center', va='bottom')

# 5.3 特征重要性（系数比较）
feature_names = diabetes.feature_names
coefficients_data = []

for name in ['Linear Regression', 'Ridge (alpha=1.0)', 'Lasso
(alpha=0.1)']:
    if results[name]['coefficients'] is not None:
        for feature, coef in zip(feature_names,
results[name]['coefficients']):
            coefficients_data.append({'Model': name, 'Feature':
feature, 'Coefficient': coef})

import pandas as pd

coef_df = pd.DataFrame(coefficients_data)

pivot_df = coef_df.pivot(index='Feature', columns='Model',
values='Coefficient')
pivot_df.plot(kind='bar', ax=axes[1, 0])
axes[1, 0].set_title('不同模型的特征系数比较')
axes[1, 0].set_ylabel('系数值')
axes[1, 0].tick_params(axis='x', rotation=45)
axes[1, 0].grid(True, alpha=0.3)
axes[1, 0].legend()
```

```
# 5.4 残差分析
residuals = y_test - best_result['predictions']
axes[1, 1].scatter(best_result['predictions'], residuals,
alpha=0.7, color='green')
axes[1, 1].axhline(y=0, color='red', linestyle='--')
axes[1, 1].set_xlabel('预测值')
axes[1, 1].set_ylabel('残差')
axes[1, 1].set_title(f'残差分析 ({best_model_name})')
axes[1, 1].grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

# 6. 详细结果分析
print("\n" + "=" * 50)

# 找到最佳模型
best_model = min(results.items(), key=lambda x: x[1]['mse'])
print(f"最佳模型: {best_model[0]}")
print(f"最佳MSE: {best_model[1]['mse']:.2f}")
print(f"最佳R2: {best_model[1]['r2']:.3f}")

# 系数分析
print("\n特征系数分析:")
linear_coef = results['Linear Regression']['coefficients']
for feature, coef in zip(feature_names, linear_coef):
    print(f"  {feature:15}: {coef:7.3f}")

# 正则化效果分析
print("\n正则化效果分析:")

# 显示拉索回归的特征选择效果
lasso_coef = results['Lasso (alpha=1.0)']['coefficients']
selected_features = [feature for feature, coef in
```

```
    zip(feature_names, lasso_coef) if abs(coef) > 0.01]
    print(f"Lasso (alpha=1.0) 选择的特征数:
    {len(selected_features)}/{len(feature_names)}")
    print(f"选择的特征: {selected_features}")

    # 7. 数据标准化效果验证
    print("\n" + "=" * 30)

    # 比较标准化前后的效果
    lr_no_scale = LinearRegression()
    lr_no_scale.fit(X_train, y_train)
    y_pred_no_scale = lr_no_scale.predict(X_test)
    mse_no_scale = mean_squared_error(y_test, y_pred_no_scale)
    r2_no_scale = r2_score(y_test, y_pred_no_scale)

    print(f"未标准化 - MSE: {mse_no_scale:.2f}, R²:
    {r2_no_scale:.3f}")
    print(f"标准化后 - MSE: {results['Linear
    Regression']['mse']:.2f}, R²: {results['Linear
    Regression']['r2']:.3f}")

    if results['Linear Regression']['mse'] < mse_no_scale:
        print("标准化提高了模型性能")
    else:
        print("标准化效果不明显")
```

2.运行结果

运行结果

```
D:\app\miniconda\envs\pytorch\python.exe D:\app\基本组件
\Desktop\jiqixuexi.py
==================================================
标准化后训练集均值: [ 0.  0. -0.  0. -0. -0.  0.  0.  0. -0.]
标准化后训练集标准差: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

```
==============================
Linear Regression    | MSE:  2900.19 | R²:  0.453
Ridge (alpha=1.0)    | MSE:  2892.01 | R²:  0.454
Ridge (alpha=10.0)   | MSE:  2875.78 | R²:  0.457
Lasso (alpha=0.1)    | MSE:  2884.62 | R²:  0.456
Lasso (alpha=1.0)    | MSE:  2824.57 | R²:  0.467
D:\app\基本组件\Desktop\jiqixuexi.py:141: UserWarning: Glyph 178
(\N{SUPERSCRIPT TWO}) missing from font(s) SimHei.
  plt.tight_layout()
D:\app\JetBrains\PyCharm
2023.1\plugins\python\helpers\pycharm_matplotlib_backend\backend
_interagg.py:68: UserWarning: Glyph 178 (\N{SUPERSCRIPT TWO})
missing from font(s) SimHei.
  FigureCanvasAgg.draw(self)

==================================================
最佳模型: Lasso (alpha=1.0)
最佳MSE: 2824.57
最佳R2: 0.467

特征系数分析:
  age              :   1.754
  sex              : -11.512
  bmi              :  25.607
  bp               :  16.829
  s1               : -44.449
  s2               :  24.641
  s3               :   7.677
  s4               :  13.139
  s5               :  35.161
  s6               :   2.351

正则化效果分析:
Lasso (alpha=1.0) 选择的特征数: 9/10
```

选择的特征: ['age', 'sex', 'bmi', 'bp', 's1', 's3', 's4', 's5', 's6']

==========

未标准化 - MSE: 2900.19, R^2: 0.453

标准化后 - MSE: 2900.19, R^2: 0.453

标准化效果不明显

进程已结束,退出代码0

3. 运行结果图

运行结果图