中北大学软件学院

实 验 报 告

专　　业：	软件工程
方　　向：	人工智能
课程名称：	机器学习实践
班　　级：	22130418
学　　号：	2213041835、2213041816
姓　　名：	吕炳荣、刘晓峰
辅导教师：	程晓鼎

2024 年 3 月制

成绩：_____

实验时间	2025 年 03 月 16 日 14 时至 18 时	学时数	4 学时

1. 实验名称

线性模型

2. 实验目的

了解线性回归模型、逻辑回归模型。会采用线性模型、逻辑回归模型根据给定的数据进行实际应用，掌握线性模型的编码实现。

3.实验内容

1）分析空气质量监测数据，根据数据建立预测 PM2.5 浓度的一元线性回归模型
2）建立预测 PM2.5 浓度的多元线性回归模型
3）分析空气质量监测数据，根据数据建立是否出现污染的逻辑回归预测模型

4. 实验原理或流程图

以空气质量监测的部分数据为例，对 PM2.5（输出变量）进行预测。首先考虑只有一个输入变量 CO 的情况，建立一元线性回归模型。然后，研究 SO2 和 CO 对 PM2.5 的影响，建立多元线性回归模型。

对是否有污染（二分类输出变量）进行预测。首先对数据进行预处理，将质量等级是优和良的合并为 0 类（无污染），其余合并为 1 类（有污染）。可以考虑 PM2.5 和 PM10 对有无污染的影响，作为输入变量，只有 0 和 1 两个取值的有无污染作为输出变量，建立逻辑回归模型。

线性回归 (Linear Regression):

原理：线性回归假设输入变量和输出变量之间存在线性关系。通过找到一条最佳拟合直线（或超平面），使得预测值与真实值之间的误差最小化（通常使用最小二乘法）。

一元线性回归公式： $y = \beta_0 + \beta_1 x + \varepsilon$

多元线性回归公式： $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p + \varepsilon$

y: 因变量（要预测的变量，如 PM2.5）

$x, x_1, x_2, ..., x_p$: 自变量（输入变量，如 CO, SO2）

β_0: 截距

$\beta_1, \beta_2, ..., \beta_p$: 自变量的系数（斜率）

ε: 误差项

逻辑回归 (Logistic Regression):

原理：逻辑回归虽然名字里有“回归”，但实际上是一种分类算法。它用于预测二元结果（0 或 1）。逻辑回归通过 sigmoid 函数（也称为 logistic 函数）将线性回归的输出映射到 0 和 1 之间，表示概率。

公式： $P(y=1) = 1 / (1 + \exp(-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p)))$

P(y=1): 因变量为 1 的概率（如发生污染的概率）

$x_1, x_2, ..., x_p$: 自变量（如 PM2.5, PM10）
β_0: 截距
$\beta_1, \beta_2, ..., \beta_p$: 自变量的系数
决策边界： 通常，如果 $P(y=1) \geq 0.5$，则预测为类别 1；否则预测为类别 0。

5. 实验过程或源代码

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, accuracy_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  # 导入 3D 绘图工具
import seaborn as sns #更容易的创建图表

# 1. 数据读取和预处理
# 假设数据文件名为 "air_quality_data.xlsx"
data = pd.read_excel("air_quality_data.xlsx")

# 查看数据的前几行
print(data.head())

# 检查缺失值
print(data.isnull().sum())

# 简单处理：删除含有缺失值的行 (更复杂的处理方法包括填充)
data = data.dropna()

# 2. 一元线性回归模型
# 2.1 数据准备
X_uni = data[['CO']]
y_uni = data['PM2.5']
X_train_uni, X_test_uni, y_train_uni, y_test_uni = train_test_split(X_uni, y_uni, test_size=0.2, random_state=42)  # 80% 训练集, 20% 测试集

# 2.2 模型训练
model_uni = LinearRegression()
model_uni.fit(X_train_uni, y_train_uni)
```

```
# 2.3 模型预测
y_pred_uni = model_uni.predict(X_test_uni)

# 2.4 模型评估
mse_uni = mean_squared_error(y_test_uni, y_pred_uni)
print(f"一元线性回归 (CO vs PM2.5) - 均方误差: {mse_uni:.2f}")
print(f"  截距 (Intercept): {model_uni.intercept_:.2f}")
print(f"  CO 系数 (Coefficient): {model_uni.coef_[0]:.2f}")

# 2.5 可视化
plt.figure(figsize=(8, 6))
plt.scatter(X_test_uni, y_test_uni, color='blue', label='实际值')  # 真实值散点图
plt.plot(X_test_uni, y_pred_uni, color='red', label='预测值')  # 预测的回归线
plt.xlabel('CO')
plt.ylabel('PM2.5')
plt.title('一元线性回归 (CO vs PM2.5)')
plt.legend()
#plt.show() #如果想在非交互式环境运行，取消这行注释

# 3. 多元线性回归模型 (CO, SO2 vs PM2.5)
# 3.1 数据准备
X_multi = data[['CO', 'SO2']]
y_multi = data['PM2.5']
X_train_multi, X_test_multi, y_train_multi, y_test_multi = train_test_split(X_multi, y_multi, test_size=0.2, random_state=42)

# 3.2 模型训练
model_multi = LinearRegression()
model_multi.fit(X_train_multi, y_train_multi)

# 3.3 模型预测
y_pred_multi = model_multi.predict(X_test_multi)

# 3.4 模型评估
mse_multi = mean_squared_error(y_test_multi, y_pred_multi)
print(f"\n多元线性回归 (CO, SO2 vs PM2.5) - 均方误差: {mse_multi:.2f}")
print(f"  截距 (Intercept): {model_multi.intercept_:.2f}")
print(f"  CO 系数: {model_multi.coef_[0]:.2f}")
print(f"  SO2 系数: {model_multi.coef_[1]:.2f}")

# 3.5 可视化 (3D 散点图)
```

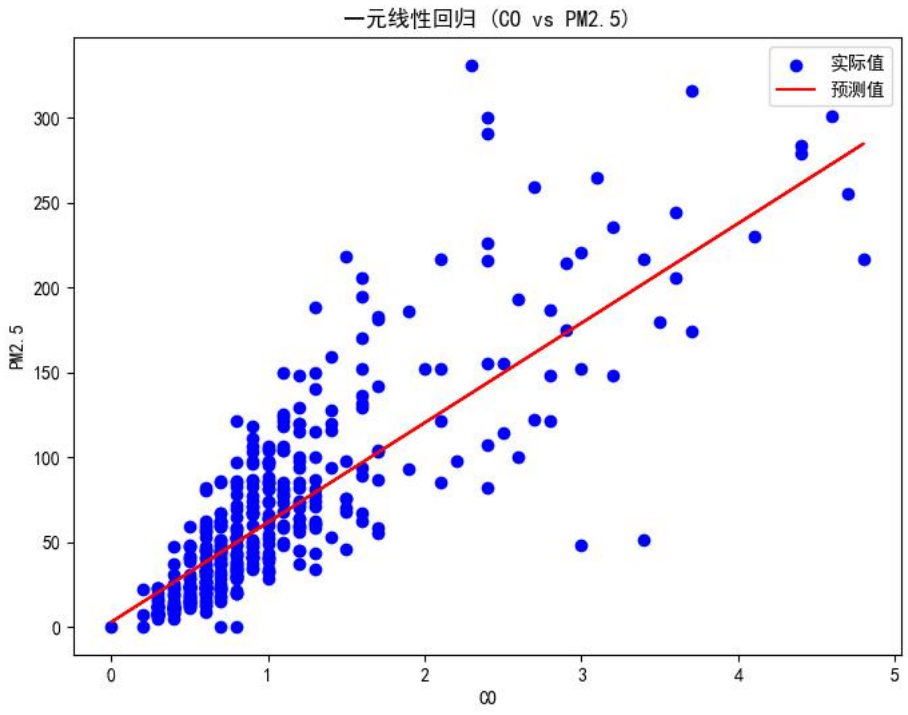
```
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X_test_multi['CO'], X_test_multi['SO2'], y_test_multi, color='blue', label='实际值')
ax.scatter(X_test_multi['CO'], X_test_multi['SO2'], y_pred_multi, color='red', marker='x', label='预测值')
ax.set_xlabel('CO')
ax.set_ylabel('SO2')
ax.set_zlabel('PM2.5')
ax.set_title('多元线性回归 (CO, SO2 vs PM2.5)')
ax.legend()
#plt.show() #如果想在非交互式环境运行，取消这行注释

# 4. 逻辑回归模型 (PM2.5, PM10 vs 污染等级)
# 4.1 数据准备: 将“质量等级”转换为二元变量 (0: 无污染, 1: 有污染)
data['污染'] = data['质量等级'].apply(lambda x: 0 if x in ['优', '良'] else 1)
X_log = data[['PM2.5', 'PM10']]
y_log = data['污染']
X_train_log, X_test_log, y_train_log, y_test_log = train_test_split(X_log, y_log, test_size=0.2, random_state=42)

# 4.2 模型训练
model_log = LogisticRegression()
model_log.fit(X_train_log, y_train_log)

# 4.3 模型预测
y_pred_log = model_log.predict(X_test_log)

# 4.4 模型评估
accuracy = accuracy_score(y_test_log, y_pred_log)
conf_matrix = confusion_matrix(y_test_log, y_pred_log)
class_report = classification_report(y_test_log, y_pred_log)

print(f"\n 逻辑回归 (PM2.5, PM10 vs 污染) - 准确率: {accuracy:.2f}")
print("混淆矩阵 (Confusion Matrix):\n", conf_matrix)
print("分类报告 (Classification Report):\n", class_report)
print(f" 截距 (Intercept): {model_log.intercept_[0]:.2f}")
print(f" PM2.5 系数: {model_log.coef_[0][0]:.2f}")
print(f" PM10 系数: {model_log.coef_[0][1]:.2f}")
```

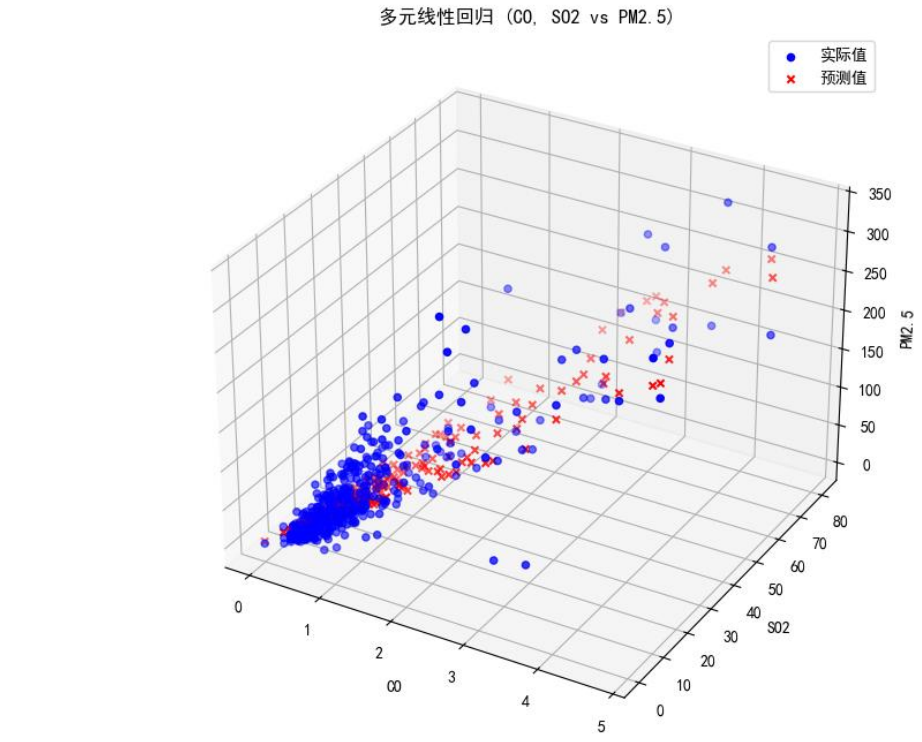
```
# 4.5 可视化 (决策边界) - 使用 seaborn
plt.figure(figsize=(8,6))
sns.scatterplot(x='PM2.5', y='PM10', hue='污染', data=data,
palette={0: 'green', 1: 'red'})

# 绘制决策边界
xlim = plt.xlim()
ylim = plt.ylim()
xx, yy = np.meshgrid(np.linspace(xlim[0], xlim[1], 100),
                     np.linspace(ylim[0], ylim[1], 100))
Z = model_log.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contour(xx, yy, Z, colors='black', levels=[0.5]) # 0.5 是决策边
界

plt.title("逻辑回归 (PM2.5, PM10 vs 污染) - 决策边界")
plt.show() #如果想在非交互式环境运行，取消这行注释
```

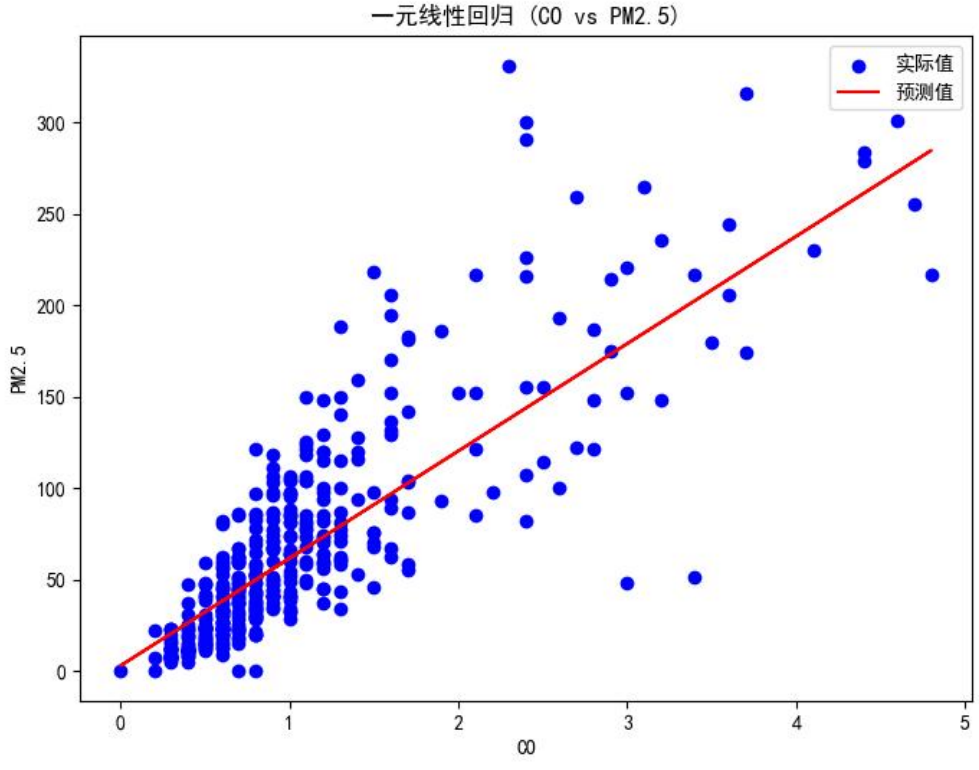
实验结果分析与讨论:

```
C:\Users\86198\.conda\envs\YOLOv8\python.exe D:\PaddlePadd
一元线性回归 (CO vs PM2.5) - 均方误差: 1117.68
  截距: 2.72
  CO 系数: 58.76

多元线性回归 (CO, SO2 vs PM2.5) - 均方误差: 1129.16
  截距: 2.68
  CO 系数: 56.56
  SO2 系数: 0.23

逻辑回归 (PM2.5, PM10 vs 污染) - 准确率: 0.84
混淆矩阵:
 [[213  23]
 [ 46 149]]
```

```
分类报告:
              precision    recall  f1-score   support

           0       0.82      0.90      0.86       236
           1       0.87      0.76      0.81       195

    accuracy                           0.84       431
   macro avg       0.84      0.83      0.84       431
weighted avg       0.84      0.84      0.84       431

  截距: -4.45
  PM2.5 系数: 0.05
  PM10 系数: 0.02
```




多元线性回归（CO, SO2 vs PM2.5）
实际值
预测值
PM2.5
CO
SO2


一元线性回归（CO vs PM2.5）
实际值
预测值
PM2.5
CO

一元线性回归 (CO vs. PM2.5):

均方误差 (MSE): 反映了模型预测值与真实值之间的平均偏差。MSE 越小，模型拟合得越好。

CO 系数: 系数为正，表明 CO 浓度增加，PM2.5 浓度也倾向于增加。系数的大小反映了 CO 对 PM2.5 影响的程度。

可视化: 散点图显示了实际数据点的分布，回归线显示了模型的预测趋势。

多元线性回归 (CO, SO2 vs. PM2.5):

MSE: 与一元线性回归相比，如果 MSE 更小，说明加入 SO2 变量提高了模型的预测能力。

CO 和 SO2 系数: 两个系数都为正，表明 CO 和 SO2 浓度增加，PM2.5 浓度都倾向于增加。 比较两个系数的绝对值大小，可以了解哪个变量的影响更大。

3D 可视化: 展示了在 CO 和 SO2 两个维度上，PM2.5 的预测值和实际值的分布情况。

逻辑回归 (PM2.5, PM10 vs. 污染):

准确率 (Accuracy): 模型正确预测污染等级的比例。

混淆矩阵 (Confusion Matrix):

展示了模型预测的正确和错误情况，包括真阳性 (TP)、假阳性 (FP)、真阴性 (TN)、假阴性 (FN)。

可以计算精确率 (Precision)、召回率 (Recall) 和 F1 值 (F1-score) 等更详细的指标。

分类报告 (Classification Report): 提供了精确率、召回率、F1 值等指标。

PM2.5 和 PM10 系数：系数的正负号表明该变量对污染概率的影响方向。系数越大（绝对值），影响越大。

可视化 (决策边界): 显示了模型如何根据 PM2.5 和 PM10 的值来划分“无污染”和“有污染”区域。

6.实验心得

通过本次实验，我深入理解了线性回归和逻辑回归模型的原理、适用场景和实现方法。

数据预处理是建模的重要环节，包括缺失值处理、数据转换等。

可视化对于理解数据和模型结果非常有帮助。

选择合适的评估指标对于比较不同模型和选择最佳模型至

关重要。

线性模型虽然简单，但在很多实际问题中仍然非常有效。

未来可以尝试：

更复杂的数据预处理方法（如插值、特征工程）。

考虑更多的输入变量（如温度、湿度、风速等）。

尝试其他类型的模型（如多项式回归、决策树、支持向量机等）。

使用交叉验证等更严格的模型评估方法。

对模型参数进行调优。