华南理工大学

《计算方法》课程实验报告

实验题目： **实验一 基于插值的数据扩充及不平衡分类问题**

**实验二 基于多变量线性回归模型的数据拟合**

姓名： 黄鸿展 学号： 202230441138 班级： 计科2班

指导教师： 王家兵

|  |
| --- |
| **实验概述** |
| 【实验目的及要求】  **实验一：基于插值的数据扩充及不平衡分类问题**  **实验目的：**  基于插值的数据扩充及不平衡分类问题  **实验要求：**   1. 了解数据挖掘python库scikit-learn 的使用，参见https://scikit-learn.org/。   2、从 UCI 机器学习数据库中(http://archive.ics.uci.edu/ml/) 下载任务为 Classification 的类严重不平衡的数据集（至少2个数据集）。  3、用线性插值法对少数类样本进行扩充（参见插值法PPT中对SMOTE的简介， 其详细介绍可以参见论文SMOTE: Synthetic Minority Over-sampling Technique）， 然后使用决策树（Decision Trees）及贝叶斯分类器（Naive Bayes）对比扩充前后 的分类准确率（参数可以多使用几个不同的值，评价指标使用F1-measure）（用 图表列出）。  **实验二：基于多变量线性回归模型的数据拟合**  **实验目的：**  掌握用多变量线性回归模型及最小二乘法进行数据拟合的基本原理。  **实验要求：**   1. 从UCI机器学习数据库中(UCI机器学习数据库：http://archive.ics.uci.edu/ml/) 下载任务为回归（Regression）的数据集（至少2个）。   2、用多变量线性模型拟合数据，并计算测试集的平方误差和的平均值（用5折 交叉验证）（用图表列出）。  3、用多变量线性模型拟合数据，但使用 L2正则项，重复上项实验。关于l参数 要进行多个参数的设置实验，并就有无正则项比较平方误差和的平均值（用图表列出）。  【实验环境】  Python 3.7.12  Imlearn(imbalanced-learn) 0.6.2  scikit-learn 1.0.2 |
| **实验内容** |
| **【实验过程】**  **实验一：基于插值的数据扩充及不平衡分类问题**   1. **数据集准备与处理**   不平衡数据集是机器学习和人工智能普遍存在的挑战。当一个类别中的样本数量明显超过另一类别时，机器学习模型往往会偏向多数类别，从而导致性能不佳。  在<https://archive.ics.uci.edu/>上，我找到了WineQuality与Abalone数据集，数据集为类严重不平衡的数据集，为了更好体现线性插值法扩充的效果，数据集均以连续值非离散值作为特征，类别分布差异大，且样本数量均超过1000：  **WineQuality（葡萄酒质量数据集）**  特征：11个指标（如酒精含量、pH值等）。  分类：**quality**，将其二值化为高质量（≥7）和普通质量（<7）。  类别分布：原始数据中高质量样本仅占约13.6%，严重不平衡。  **Abalone（鲍鱼年龄数据集）**  特征：8个形态特征（如长度等），首列性别映射为数值（0/1/2）。  分类：鲍鱼年龄是否超过10年（Rings>10）。  类别分布：超过10年的样本占比约20.6%，存在明显不平衡。  对数据进行预处理：  标准化处理：使用StandardScaler对特征进行标准化。  数据集划分：按7:3比例划分训练集和测试集，固定随机种子。   1. **线性插值法扩充数据**   SMOTE（Synthetic Minority Over-sampling Technique）是一种针对类别不平衡问题的过采样方法，核心思想是通过线性插值在少数类样本的特征空间中生成合成样本，从而平衡类别分布。  算法原理如下：   1. 选择少数类样本： 遍历所有少数类样本，对每个样本xi，执行以下步骤。 2. 寻找最近邻： 计算的k个最近邻（通常使用欧氏距离），仅从少数类样本中选择。 （默认k=5，通过k\_neighbors参数调整） 3. 生成合成样本： 对每个最近邻 ，随机选择一个插值比例λ（0<λ<1），生成新样本： 4. 根据目标过采样比例（如将少数类样本数量翻倍），重复上述过程。   SMOTE算法适用于特征空间连续且特征间关系线性可分的样本，而且更好作用域少数类样本数量较少但分布较为密集的数据集，因此满足我们少数类数据集的分类需求。然而若少数类包含噪声点（如离群值），SMOTE会生成无效样本。因此需要我们主动剔除SMOTE数据集的离散值，避免影响结果.   1. **使用决策树与贝叶斯分类**   3.1 决策树分类  模型选择：使用DecisionTreeClassifier，调节max\_depth参数控制模型。  参数设置：   * 1. max\_depth：树的最大深度，取值[3,5,7,None]（None不限制深度）。   2. random\_state=42：固定随机种子，确保结果一致性。   训练与评估：  ① 原始数据训练：在标准化后的训练集上直接训练。  ② SMOTE 数据训练：在 SMOTE 生成的平衡数据上训练。  测试集预测：对标准化后的测试集进行预测，计算F1分数。  3.2 贝叶斯分类  模型选择：使用GaussianNB(高斯朴素贝叶斯).  参数设置：var\_smoothing：方差平滑参数，用于避免零方差问题，取值[1e-9, 1e-6, 1e-3]。  测试集预测：计算 F1 分数，评估模型性能。  **实验二：基于多变量线性回归模型的数据拟合**   1. **数据集准备与处理**   多变量线性回归是一种统计方法，它用于分析两个或多个自变量（特征）与因变量（目标）之间的线性关系。在多变量线性回归中，每个自变量的影响被赋予一个系数，这些系数共同决定了因变量的预测值。在<https://archive.ics.uci.edu/>上，我找到了Concrete与Energy数据集，均为回归数据集，且样本数量均超过700：  **Concrete（混凝土抗压强度数据集）**  特征：8个输入变量，包括水泥含量、骨料比例及养护天数等。  目标变量：混凝土抗压强度（MPa），取值范围为0~82.6 MPa。  样本数量：1030条。  **Energy（建筑能效数据集）**  特征：8个输入变量，包括建筑表面积、外墙隔热性能等。  目标变量：供暖负荷，取值范围为6~43 kWh。样本数量：768条。  对数据进行预处理：   1. 标准化处理：使用 StandardScaler 进行标准化，消除量纲差异。 2. 数据划分策略：采用五折交叉验证，避免单次划分的随机性影响。 3. **模型选择与参数设置**   最小二乘法（Least Squares Method）是一种通过最小化预测值与真实值之间的残差平方和来估计模型参数的数学优化方法。其核心思想是寻找一组参数，使得模型预测值与实际观测值之间的差异（即残差）的平方和达到最小。该方法广泛应用于线性回归模型的参数估计中。  设多维线性回归模型为：    目标是通过最小化残差平方和来求解参数*θ*。  我们定义，为了最小化*J*(*θ*)，对*θ*求导并令导数为零，得：  。  本次实验中，首先选择普通多变量线性回归：  普通线性回归（Linear Regression）：基于最小二乘法拟合数据，作为模型。  然而，最小二乘法对多重共线性敏感：若特征间存在高度相关性，其逆矩阵可能不稳定，导致参数估计方差过大。而且当特征维度高或样本量少时，模型容易过拟合训练数据，泛化性能下降。  因此，为解决上述问题，岭回归（Ridge Regression）在目标函数中引入L2正则化项，修改后的目标函数为：    令导数为0，得到解：    通过调节*λ*权衡拟合能力与模型复杂度，抑制过拟合。   1. **模型评估方法**   6.1 五折交叉验证  五折交叉验证（5-fold cross-validation）是一种常用的机器学习模型评估技术，通常用于估计模型的性能和泛化能力。它的基本思想是将原始数据集分成五个相等大小的子集（或折叠），其中四个子集用于训练模型，而剩下的一个子集用于测试模型。这个过程重复五次，每次选择不同的一个子集作为测试集，其余的作为训练集。最后，将五次的性能评估结果取平均值以得到最终评估结果。  五折交叉验证使用KFold(n\_splits=5, shuffle=True)划分数据。计算每折的MSE，取平均值作为最终结果。均方误差（MSE）公式为：    6.2 正则化λ参数  λ通过控制L2正则化项的强度，直接影响模型的复杂度与过拟合风险。较小的λ（如0.1）对参数惩罚较弱，模型更倾向于拟合训练数据，但在高方差数据中可能过拟合；较大的λ（如10）则增强惩罚，限制参数大小，降低模型复杂度，适用于缓解多重共线性或噪声干扰。  **【实验结果】**  **实验一：基于插值的数据扩充及不平衡分类问题**  **决策树分类实验结果分析**  数据集：WineQuality  参数设置：最大深度（max\_depth）分别为3、5、7和无限制（None）。  原始数据表现：F1分数随深度增加呈先升后降趋势。当max\_depth=5时达到峰值（约0.54），无限制时下降至0.46，表明过拟合。  SMOTE扩充后表现：F1分数整体提升，尤其在max\_depth=5时达到0.65，较原始数据提升约20%。无限制时仍保持0.55，缓解了过拟合问题。  数据集：Abalone  原始数据表现：F1分数随深度增加逐步提高，无限制时达到0.61。  SMOTE扩充后表现：F1分数在max\_depth=5时最高（0.63）表现更稳定。  SMOTE在Abalone数据集上提高了分数，但提升幅度较小（约3%），  **贝叶斯分类实验结果分析**  数据集：WineQuality  参数设置：方差平滑参数（var\_smoothing）分别为1e-9、1e-6、1e-3。  原始数据表现：F1分数在贝叶斯分类后有所上升，最佳为1e-9，F1=0.540。  SMOTE扩充后表现：F1分数在1e-9时提升至0.462，整体有所下降。  结论：贝叶斯分类器对特征分布的假设（高斯独立）可能不符合实际数据分布，SMOTE生成的线性插值样本未能显著改善模型性能。  数据集：Abalone  原始数据表现：最佳参数为1e-9，F1=0.548；平滑参数增大导致性能下降。  SMOTE扩充后表现：F1分数在1e-9时提升至0.553。  结论：贝叶斯分类器对Abalone数据集的敏感度较低，SMOTE的改进有限，可能与贝叶斯模型对类别先验概率的依赖较强有关，而SMOTE未调整先验分布。  **综上所述，**SMOTE方法对少数集样本填充在决策树模型上效果显著，尤其在WineQuality数据集上F1提升20%；对朴素贝叶斯分类器有提升，但效果有限。      **实验二：基于多变量线性回归模型的数据拟合**  1. 任务2结果分析（普通线性回归，无正则化）  Concrete数据集：MSE ≈ 100  原因分析：混凝土抗压强度（0~82.6 MPa）的绝对数值较高，导致平方误差的绝对值自然较大；特征间可能存在高度相关性（如水泥与骨料比例），导致参数估计不稳定，模型泛化能力差。  结论：普通线性回归在该数据集上表现较差，需引入正则化提升性能。  Energy数据集：MSE ≈ 8.8  原因分析：目标变量范围小：供暖负荷（6~43 kWh）的数值范围较窄，平方误差绝对值天然较低；建筑参数（如隔热性能、玻璃面积）与供暖负荷的线性关系显著，模型能较好拟合数据。样本特征分布较为均匀，减少异常值对预测的影响。  结论：普通线性回归在此类低复杂度、高线性可分数据上表现优异。    2. 任务3结果分析（带L2正则化）  Concrete数据集：MSE仍接近100，λ=0.1/1/10情况下略微降低，但改变效果不明显。可能原因是存在异常值或测量误差，线性模型可能放大噪声影响。  Energy数据集：MSE稳定在8.8左右，在带入L2正则化的情况下明显误差变小，λ=0.1时误差最小。此时可以看出L2正则化以后，对多变量线性回归提高了泛化能力，防止回归模型过拟合，而且一定程度上减少了对多重共线性的敏感  综上所述，最小二乘法通过最小化预测值与实际值之间的误差平方和来拟合模型，可以有效地进行多变量回归拟合；而正则化项则通过对模型复杂度的惩罚来防止过拟合，提高模型的性能与泛化能力。    【源码】  实验一：   |  | | --- | | import pandas as pd import numpy as np from sklearn.model\_selection import train\_test\_split from sklearn.tree import DecisionTreeClassifier from sklearn.naive\_bayes import GaussianNB from sklearn.metrics import f1\_score from imblearn.over\_sampling import SMOTE from sklearn.preprocessing import StandardScaler import matplotlib.pyplot as plt *# 本地路径配置* DATASET\_PATHS = {      'WineQuality': './winequality-red.csv',     'Abalone': './abalone.data' }  def load\_winequality():     df = pd.read\_csv(DATASET\_PATHS['WineQuality'], sep=';')     *# 创建二元标签：quality >= 7 为高质量（1），否则为0*     y = (df['quality'] >=7).astype(int)     X = df.drop(columns=['quality']).values     print("类别分布:", pd.Series(y).value\_counts())  *# 检查不平衡程度*     return X, y def load\_abalone():     df = pd.read\_csv(DATASET\_PATHS['Abalone'], header=None)     df[0] = df[0].map({'M': 0, 'F': 1, 'I': 2})     X = df.iloc[:, :-1].values     y = (df.iloc[:, -1] > 10).astype(int)     return X, y datasets = {     'WineQuality': load\_winequality(),     'Abalone': load\_abalone() } *# 决策树参数和图表设置* depths = [3, 5, 7, None] plt.figure(figsize=(15, 8)) for idx, (name, (X, y)) in enumerate(datasets.items(), 1):     X\_train,X\_test,y\_train,y\_test=train\_test\_split(X, y,test\_size=0.3,random\_state=42)     scaler = StandardScaler()     X\_train\_scaled = scaler.fit\_transform(X\_train)     X\_test\_scaled = scaler.transform(X\_test)     sm = SMOTE(random\_state=42)     X\_res, y\_res = sm.fit\_resample(X\_train\_scaled, y\_train)     *# 训练决策树并记录结果*     dt\_results = {'Original': [], 'SMOTE': []}     for depth in depths:         clf = DecisionTreeClassifier(max\_depth=depth, random\_state=42)         *# 原始数据*         clf.fit(X\_train\_scaled, y\_train)         y\_pred = clf.predict(X\_test\_scaled)         dt\_results['Original'].append(f1\_score(y\_test, y\_pred))         *# SMOTE数据*         clf.fit(X\_res, y\_res)         y\_pred = clf.predict(X\_test\_scaled)         dt\_results['SMOTE'].append(f1\_score(y\_test, y\_pred))      plt.subplot(1, 2, idx)     x\_labels = [str(d) if d else 'No Limit' for d in depths]     x\_ticks = np.arange(len(depths))     plt.plot(x\_ticks, dt\_results['Original'], marker='o', label='Original')     plt.plot(x\_ticks, dt\_results['SMOTE'], marker='o', label='SMOTE')     plt.xticks(x\_ticks, x\_labels)     plt.title(f'{name} - Decision Tree (F1 Score)')     plt.xlabel('Max Depth')     plt.ylabel('F1 Score')     plt.legend() plt.tight\_layout() plt.show() *# 贝叶斯参数和图表设置* var\_smoothings = [1e-15, 1e-9, 1e-3] plt.figure(figsize=(15, 8)) for idx, (name, (X, y)) in enumerate(datasets.items(), 1):     X\_train,X\_test,y\_train, y\_test=train\_test\_split(X,y,test\_size=0.3,random\_state=42)     scaler = StandardScaler()     X\_train\_scaled = scaler.fit\_transform(X\_train)     X\_test\_scaled = scaler.transform(X\_test)     sm = SMOTE(k\_neighbors=5, random\_state=42)     X\_res, y\_res = sm.fit\_resample(X\_train\_scaled, y\_train)      nb\_results = {'Original': [], 'SMOTE': []}     for vs in var\_smoothings:         clf = GaussianNB(var\_smoothing=vs)          clf.fit(X\_train\_scaled, y\_train)         y\_pred = clf.predict(X\_test\_scaled)         nb\_results['Original'].append(f1\_score(y\_test, y\_pred))          clf.fit(X\_res, y\_res)         y\_pred = clf.predict(X\_test\_scaled)         nb\_results['SMOTE'].append(f1\_score(y\_test, y\_pred))      plt.subplot(1, 2, idx)     x\_ticks = np.arange(len(var\_smoothings))     width = 0.35     bars\_orig = plt.bar(x\_ticks - width / 2, nb\_results['Original'], width, label='Original')     bars\_smote = plt.bar(x\_ticks +width/ 2, nb\_results['SMOTE'], width, label='SMOTE')      for bars, values in zip([bars\_orig, bars\_smote], [nb\_results['Original'], nb\_results['SMOTE']]):         for bar, value in zip(bars, values):             height = bar.get\_height()             plt.text(                 bar.get\_x() + bar.get\_width() / 2,  *# x坐标：柱子中心*                 height + 0.01,  *# y坐标：柱子顶部上方0.01*                 f'{value:.3f}',  *# 显示值保留3位小数*                 ha='center',  *# 水平居中*                 va='bottom'  *# 垂直底部对齐*             )     plt.xticks(x\_ticks, [f'vs={vs:.0e}' for vs in var\_smoothings])     plt.title(f'{name} - Naive Bayes (F1 Score)')     plt.xlabel('Var Smoothing')     plt.ylabel('F1 Score')     plt.legend() plt.tight\_layout() plt.show() |   实验二：   |  |  | | --- | --- | | import pandas as pd from sklearn.linear\_model import LinearRegression, Ridge from sklearn.model\_selection import cross\_val\_score, KFold from sklearn.preprocessing import StandardScaler import matplotlib.pyplot as plt import numpy as np DATASET\_PATHS = {     'Concrete': './Concrete\_Data.xls',       'Energy': './ENB2012\_data.xlsx'    } def load\_concrete():     df = pd.read\_excel(DATASET\_PATHS['Concrete'])  *# 注意使用 read\_excel 读取 .xls 文件*     X = df.iloc[:, :-1].values     y = df.iloc[:, -1].values     return X, y def load\_energy():     df = pd.read\_excel(DATASET\_PATHS['Energy'])     X = df.iloc[:, :-2].values     y = df.iloc[:, -2].values     return X, y datasets = {     'Concrete': load\_concrete(),     'Energy': load\_energy() } *# ------------------------- 任务2：普通线性回归（无正则项） -------------------------* plt.figure(figsize=(10, 5)) for idx, (name, (X, y)) in enumerate(datasets.items(), 1):     scaler = StandardScaler()     X\_scaled = scaler.fit\_transform(X)     model = LinearRegression()     kf = KFold(n\_splits=5, shuffle=True, random\_state=42)     scores = -cross\_val\_score(model, X\_scaled, y, cv=kf, scoring='neg\_mean\_squared\_error')     mse = np.mean(scores)     plt.subplot(1, 2, idx)     plt.bar(['Linear Regression'], [mse])     plt.title(f'{name} - Task 2: MSE (No Regularization)')     plt.ylabel('Mean Squared Error') plt.tight\_layout() plt.show() *# ------------------------- 任务3：岭回归（带L2正则项） -------------------------* plt.figure(figsize=(15, 6)) for idx, (name, (X, y)) in enumerate(datasets.items(), 1):     scaler = StandardScaler()     X\_scaled = scaler.fit\_transform(X)     *# 定义不同λ值的岭回归模型，并包含普通线性回归作为对比*     models = {         'Linear': LinearRegression(),         'Ridge (λ=0.1)': Ridge(alpha=0.1),         'Ridge (λ=1)': Ridge(alpha=1),         'Ridge (λ=10)': Ridge(alpha=10)     }     results = {}     kf = KFold(n\_splits=5, shuffle=True, random\_state=42)     for model\_name, model in models.items():         scores = -cross\_val\_score(model, X\_scaled, y, cv=kf, scoring='neg\_mean\_squared\_error')         results[model\_name] = np.mean(scores)     plt.subplot(1, 2, idx)     plt.bar(results.keys(), results.values())     plt.xticks(rotation=45)     plt.title(f'{name} - Task 3: MSE with L2 Regularization')     plt.ylabel('Mean Squared Error') plt.tight\_layout() plt.show() |  | |
| **小结** |
| 实验一中我学习了利用线性插值SMOTE方法对少量类样本进行扩充，以优化分类效果。实验过程中我选取了红酒质量与鲍鱼数据集作为验证数据集，利用SMOTE方法对少数类样本抽样扩充，并利用决策树模型与贝叶斯分类器进行分类，以F1Score为评判标准，最后结果表明SMOTE方法优化分类对决策树与贝叶斯均有效。实验二中我了解了多变量线性回归模型，以及最小二乘法进行数据拟合的基本原理。最小二乘法可以解决多变量线性回归的问题，为了提高泛化能力防止过拟合，可以配合L1L2正则化，通过控制正则化参数，调整最小二乘法的解，从而增强最小二乘法的拟合效果。 |
| **指导教师评语及成绩** |
| 评语：  成绩：           指导教师签名：                                                 批阅日期： |