糖尿病风险分析与预测

摘要

糖尿病是一种常见且潜在严重的健康问题，全球数百万人受其影响。准确预测糖尿病风险有助于早期检测和预防工作的开展。本研究提出了一个综合框架，通过数据分析、建模和预测空腹血糖水平来预测糖尿病风险。我们使用了一个包含 42 个指标的训练数据集（不包括血糖）和一个不包含血糖信息的测试数据集。

我们首先对训练数据集进行了数据清洗，删除了不相关的列，并使用均值填充法填充缺失值。为了增强数据的可比性，我们进行了最大最小标准化，并利用箱线图和正态分布曲线探索了数据分布规律。随后，我们使用 Pearson 相关系数评估特征之间的相关性，并筛选出最相关的预测指标。通过数据分布图比较训练数据集和测试数据集的数据分布差异，以及后续模型测试阶段对不同特征集的测试，我们进一步优化了特征集，最终选择了 13 个特征，确保其适用于准确的预测。

在模型选择阶段，我们评估了多个常用的回归模型，如线性回归、岭回归、Lasso 回归等。然而，单个模型的预测效果并不理想，无法满足我们的需求。为构建稳健的预测模型，我们采用了模型融合技术，将多个模型结合在一起，对经过优化的特征集进行训练和验证。我们使用了均方误差、标准差和 R 方等评估指标来评估模型的性能。经过筛选的特征集表现出较强的预测能力（均方误差: 0.0303; 标准差: 0.8844; R 方: 0.9627），并在应用于测试数据集, 后对测试数据集的预测时也体现出了较好的泛化能力。

综上所述，本研究提出了一个基于回归分析的综合工作流，用于预测糖尿病风险。通过严格的数据预处理、特征选择和模型融合，我们实现了对空腹血糖水平的准确和一致预测。研究结果有助于早期识别糖尿病风险个体，促进有针对性的预防策略的制定，改善公共健康结果。未来的研究可以探索更多的特征工程技术，并扩展模型融合方法，进一步提高预测性能。

关键字: 数据预处理　特征工程　回归模型　模型融合　模型评估

一、 问题重述

1.1 问题背景

糖尿病是一种代谢性疾病，其特征是血液中的血糖水平持续高于正常范围。糖尿病的发生通常与胰岛素的不足或胰岛素的功能异常有关。糖尿病发生时，由于胰腺无法产生足够的胰岛素或身体对胰岛素的利用效率降低，导致血糖在体内累积，从而发生高血糖。糖尿病作为一种常见的慢性疾病，目前无法根治。对检测数据进行数据分析与挖掘以发现疾病并尽早地进行干预可以减轻并发症的风险，帮助进行针对性的治疗，对于了解糖尿病的患病情况、控制管理策略的效果以及预防干预措施的实施具有重要意义。

1.2 已知条件

1. 有血糖值的检测数据
2. 无血糖值的检测数据

1.3 需解决的问题

1. 结合附件 1 的检测数据信息，对样本数据进行预处理，并作检测指标的统计分析，解释数据揭示的规律性。
2. 在问题 1 的基础上，通过降维的方法从 42 个检测指标中筛选出主要变量指标，使之尽可能具有代表性，并详细说明建模的主要变量的筛选过程及其合理性。
3. 在问题 2 的基础上，采用上述样本和建模的主要变量，通过数据挖掘技术建立血糖值预测模型，并进行模型验证。
4. 在问题 3 基础上，利用该模型对附件 2 的检测数据的血糖值进行预测，并对糖尿病风险进行说明。

二、 问题分析

2.1 问题一分析

由于在数据采集的过程中，某些受试者可能没有提供血糖值或其他相关检测指标的数据，人为错误或输入错误等都会对数据的质量产生影响。为了清洗数据，首先检查数据是否有缺失值，其变量类型，数据集所占的总内存等信息。经检查发现：存在字符型类型变量”性别“，需要将其转为数值型；存在缺失值，考虑使用 3σ 原则进行插值填充。最后需要进行 min-max 标准化去除变量的量纲。

为了探索数据的规律，需要进行数据的统计分析以及可视化。首先计算其均值，标准差，最值，并用箱线图对其进行可视化分析。为了检查数据的分布，进行异常值的检测，需要绘制各个指标的正态分布图。

2.2 问题二分析

问题二要求使用降维的方法从 42 个指标中筛选出具有代表性的指标。首先可以根据现实经验删除与血糖无关的指标，如 id 和体检日期；在问题一的基础上，可以根据数据分析结果发现数值基本没有发生变化的特征，由于数值变化过小，故可以认为其数值的变化不会干扰到血糖值的预测结果，可以去除；然后可以观察训练集的数据和测试集的数据分布情况，若不一致则会影响模型的泛化能力，应该进行去除。

最后本模型使用 Pearson 相关性系数的方法，将所有指标按照与目标指标” 血糖 “的相关性进行从大到小的排序。由于难以确定选择指标的个数，我们在问题三中选定模型后进行了测试，最后发现当指标个数为 13 的时候有最好的训练效果。

2.3 问题三分析

问题三要求在选定指标的基础上，建立血糖值预测模型。在本赛题中，仍然具有多个特征变量，由于其预测值” 血糖 “为连续型数值变量，故此问题为回归预测求解。常见的回归模型有线性回归模型、岭回归模型、SVR 模型、随机森林模型等，我们将采用这些模型来预测目标值并根据均方误差 MSE 和决定系数 R^2 来进行评估。

若预测效果不理想则在单模型的基础上，选出效果相对较好的模型进行多模型融合，进一步观察预测效果。

2.4 问题四分析

问题四要求在问题三的基础上对附件 2 中的数据进行血糖值预测。运行选择好的模型即可。为了进一步说明预测情况，我们将绘制图表以可视化结果。

三、 模型假设与符号说明

3.1 模型假设

1. 独立同分布假设：假设数据是从同一总体独立地采样得到的，即每个样本之间是独立的，且样本之间具有相同的概率分布。
2. 缺失数据假设：在数据采集过程中，假设数据的缺失是随机的，并且缺失的原因与缺失值本身无关。

3. 特征无关假设：在选择代表性指标的过程中，假设删除的特征对血糖值的预测结果没有重要影响。这个假设是基于观察到的数据分布和特征与目标变量之间的相关性。
4. 泛化假设：在使用训练好的模型对附件 2 中的数据进行预测时，假设模型能够泛化到新的未见数据。这意味着模型在训练集上学到的关系和模式能够适用于测试集或附件 2 中的数据。

3.2 符号说明

表 1　符号说明表

符号	含义
MSE	均方误差
mean	平均值
std	标准差
min	最小值
max	最大值
R^2	决定系数

四、 问题一模型的建立与求解

4.1 数据预处理

数据预处理在数据分析中起到非常重要的作用。对数据进行预处理，从而提高模型的质量。首先检查数据是否有缺失值，其变量类型，数据集所占的总内存等信息。经检查发现：存在字符型类型变量”性别“，这里使用 0 代表男性，1 代表女性将其数值化；存在缺失值，为了不影响数据的整体分布情况，使用均值进行插值填充。

初步清洗后进行异常值处理。异常值指的是与整体数据情况偏离很大的数据点即超过某个不合理范围的数据点，为保证分析的准确性应剔除异常值。本模型采用常见的 3σ 原则进行异常值检测，并使用均值替换各个变量中的异常值。

进一步的，为了消除不同特征之间的量纲差异，将数据映射到相同的尺度范围内进行评估，需要对数据进行去量纲和标准化。这里使用 min-max 方法对原始数据应用线性

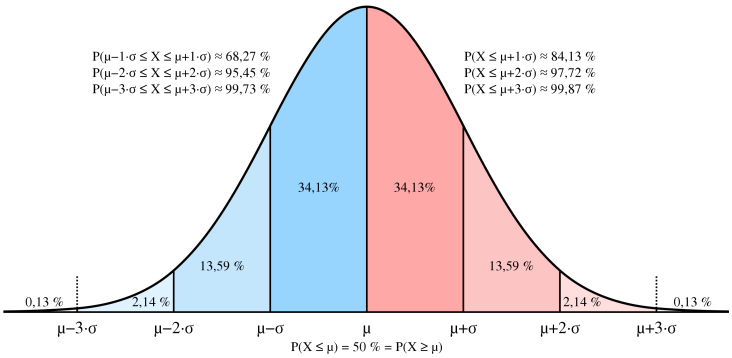

图 1　3σ 原则

变换，使得数据的最小值映射为 0，最大值映射为 1，其他值按比例映射到 0 和 1 之间。其去量纲方法的公式如下：

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

其中，原始值是指输入数据中的一个特定数值，最小值和最大值分别是输入数据的最小值和最大值。部分数据标准化后的折线图如下： 可以看出所有数据均是在 [0,1] 之间的。

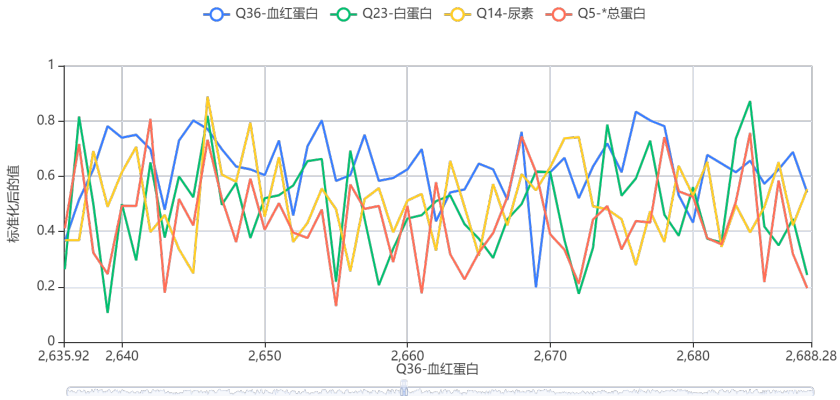

图 2　标准化后的数据

4.2 描述性统计分析

为了探索数据的规律，我们进行了数据的统计分析以及可视化。

首先计算其均值、标准差和最值. 由于指标数量较多，无法完整展示统计性分析的结果，这里仅展示部分指标的统计结果, 如图 3 所示。

从表中可以看出，进行数据预处理后保留了 4373 条数据。从数据的变异程度来看，标准差大部分小于 0.15，说明值在均值附近的波动较小；从数据的分布形态来看，这些指标的分布大致接近正态分布，因为它们的中位数（50%）接近均值，而且上下四分位数 25%□75% 相对接近，说明数据集的分布比较均匀。

	*总蛋白	中性粒细胞%	低密度脂蛋白胆固醇	单核细胞%	嗜碱细胞%	嗜酸细胞%	尿素	尿酸	年龄	性别
count	4373.000000	4373.000000	4373.000000	4373.000000	4373.000000	4373.000000	4373.000000	4373.000000	4373.000000	4373.000000
mean	0.495151	0.495487	0.490012	0.436654	0.416263	0.266053	0.489327	0.475395	0.436685	0.515436
std	0.155604	0.163471	0.152005	0.169215	0.181569	0.185699	0.148521	0.154810	0.184210	0.499819
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.411652	0.385621	0.401766	0.309524	0.285714	0.126761	0.403298	0.379992	0.289855	0.000000
50%	0.497043	0.494553	0.496088	0.428571	0.428571	0.211268	0.499113	0.484828	0.434783	1.000000
75%	0.571156	0.605664	0.565121	0.547619	0.500000	0.352113	0.544228	0.537778	0.565217	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

图 3　统计分析结果

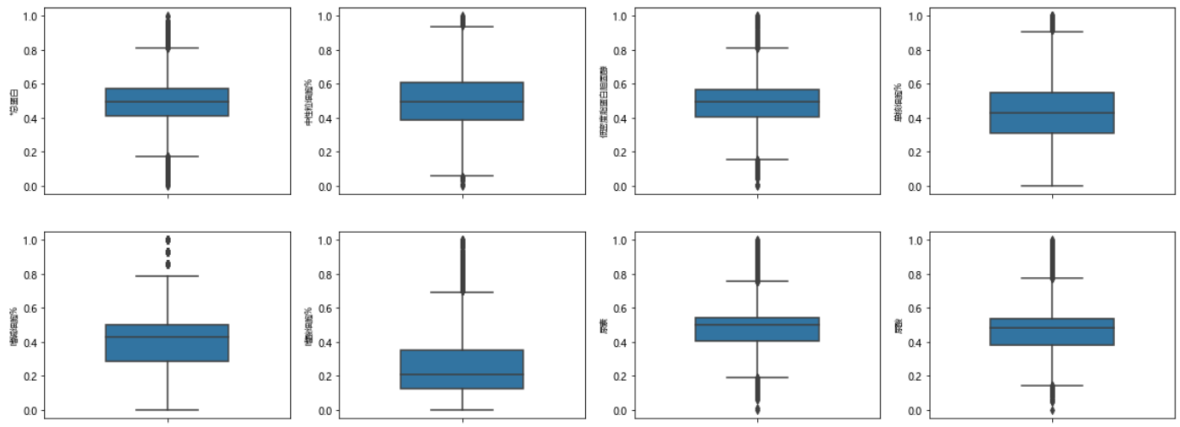

图 4　箱线图分析

为了可视化表格中的结果，我们使用箱线图对其进行可视化分析。

通过箱型图，可以观察到 g 各个指标的中位数和四分位数，从而了解数据的中心趋势和分布情况。此外，通过盒须的长度和异常值的存在，我们可以判断数据的离群情况和可能存在的异常值。

为了检查数据的分布，进行异常值的检测，我们绘制了各个指标的正态分布图，由于指标较多，这里仅展示部分结果（图 5）。

可以看出并不是所有的指标都符合正态分布，后面可以通过对数变换等方法进行矫正。

五、 问题二模型的建立与求解

由于数据集中存在 42 个指标，维度较高，所以需要进行降维处理。通过降维可以达到简化数据集、提高数据分析和建模的效率、减少计算复杂度的效果，对于进一步进行预测与分析具有重要作用。

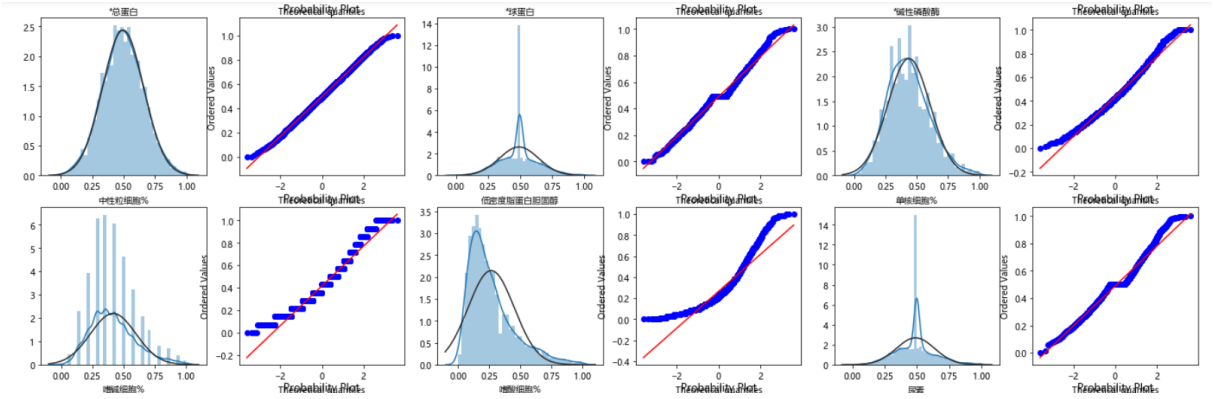

图 5　部分指标正态分布图

5.1 根据现实因素剔除无关指标

指标血糖作为因变量，不作分析。指标 id 只是数据记录的唯一标识符，不包含任何与预测结果相关的信息。其无法提供有关目标变量的任何洞见或解释力，因此可以剔除。体检日期与目标变量血糖之间没有实质性的关联，不会对预测结果产生直接或间接的影响，因此可剔除以减少模型的复杂性。部分指标缺失值较多，这可能是因为数据的性质或测量方法决定的。由于数据的缺失，在本模型中进行剔除。

5.2 淘汰数值基本无变化的指标

观察箱线图可以发现，有几个指标的数值基本没有发生变化，如

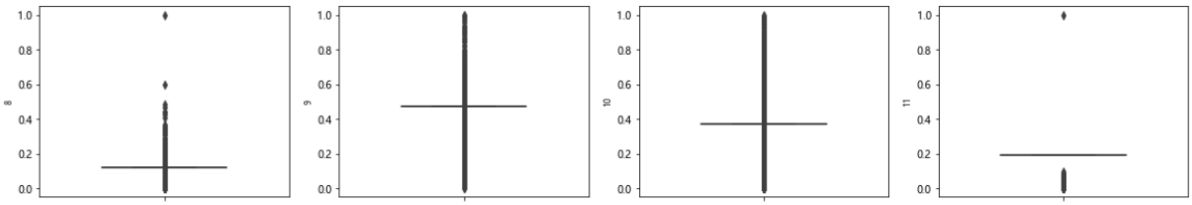

图 6　数值基本不变的特征

这些指标分别是：乙肝 e 抗体，乙肝 e 抗原，乙肝核心抗体，乙肝表面抗体和乙肝表面抗原。

观察原始数据集可以发现，这些指标存在大量的缺失值，大部分是在数据预处理的过程中，通过少数数据的均值进行填充的。考虑到样本量较少故不具有代表性，应该进行删除。

然后本模型从训练集与测试集的角度出发，绘制了各个指标的数值分布图（图 6）。

为了更加专业的描述附件 1 与附件 2，我们将附件 1 中的数据称为 train，附件 2 中的数据称为 test。

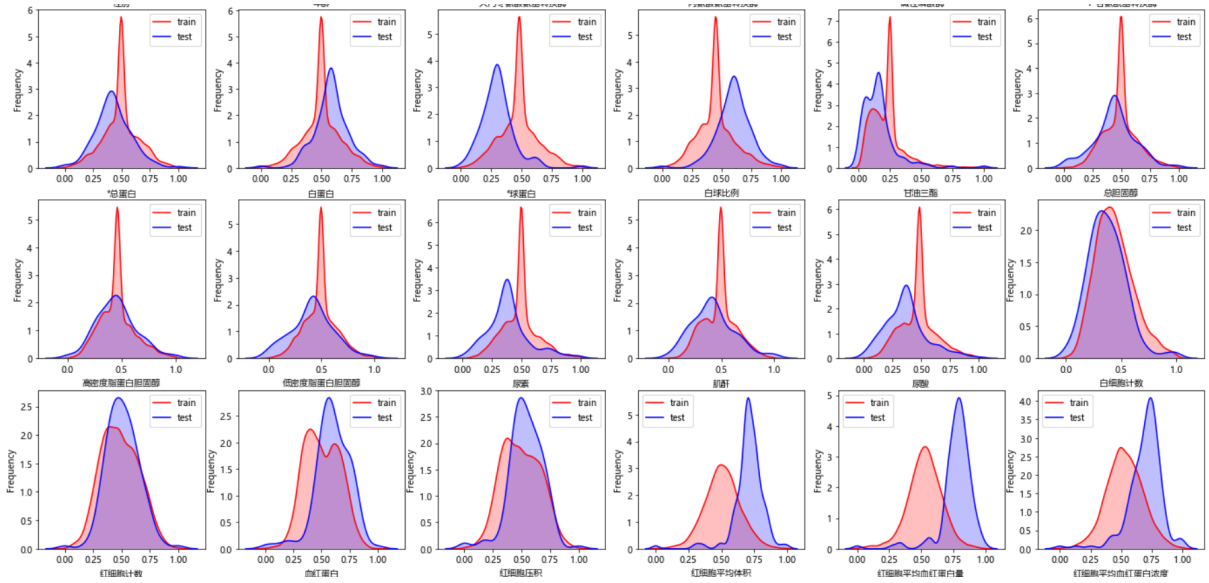

图 7　附件 1 数据与附件 2 数据对比

查看特征”* 天门冬氨酸氨基转换酶”,”* 丙氨酸氨基转换酶”,”* 碱性磷酸酶”,”*r-谷氨酰基转换酶”,”* 球蛋白”,” 白球比例”,” 红细胞平均体积”,” 红细胞平均血红蛋白量”,” 红细胞平均血红蛋白浓度”,” 红细胞体积分布宽度”,” 血小板体积分布宽度”,” 血小板比积” 数据的数据分布:

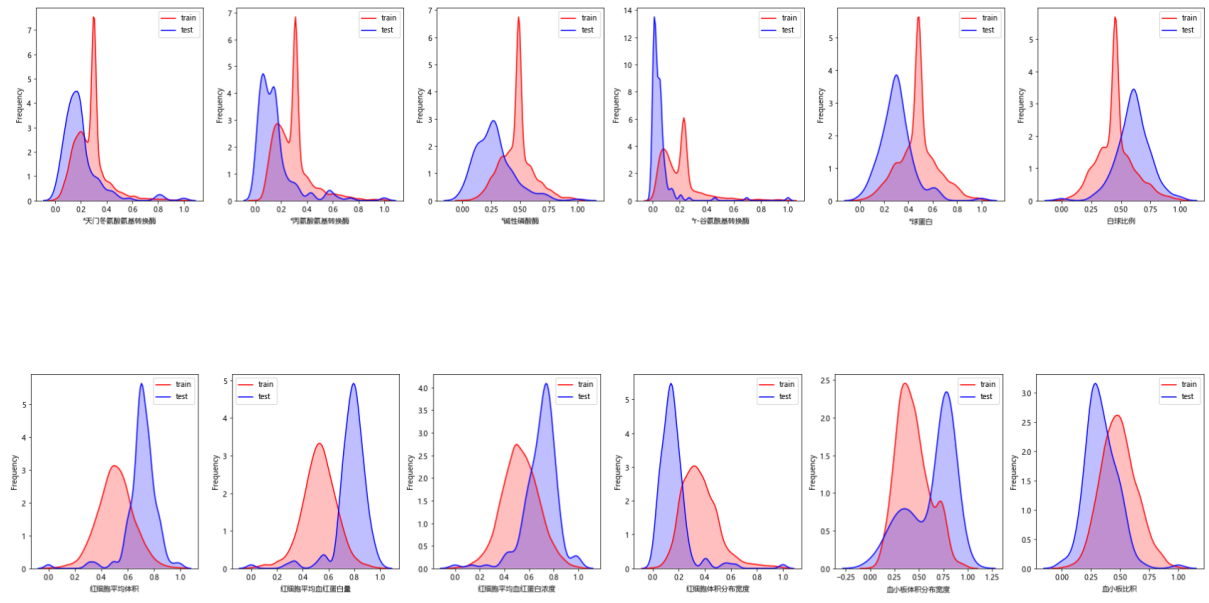

图 8　训练集与测试集中分布情况不一致的特征

这几个特征下，训练集的数据和测试集的数据分布不一致，会影响模型的泛化能力，故删除天门冬氨酸氨基转换酶”,”丙氨酸氨基转换酶等 12 个特征。

最后该模型使用 Pearson 算法计算各个特征相对于血糖的相关系数，并进行热力图绘制（图 9）筛选出与” 血糖” 相关性最高的前 10 个指标进行展示:

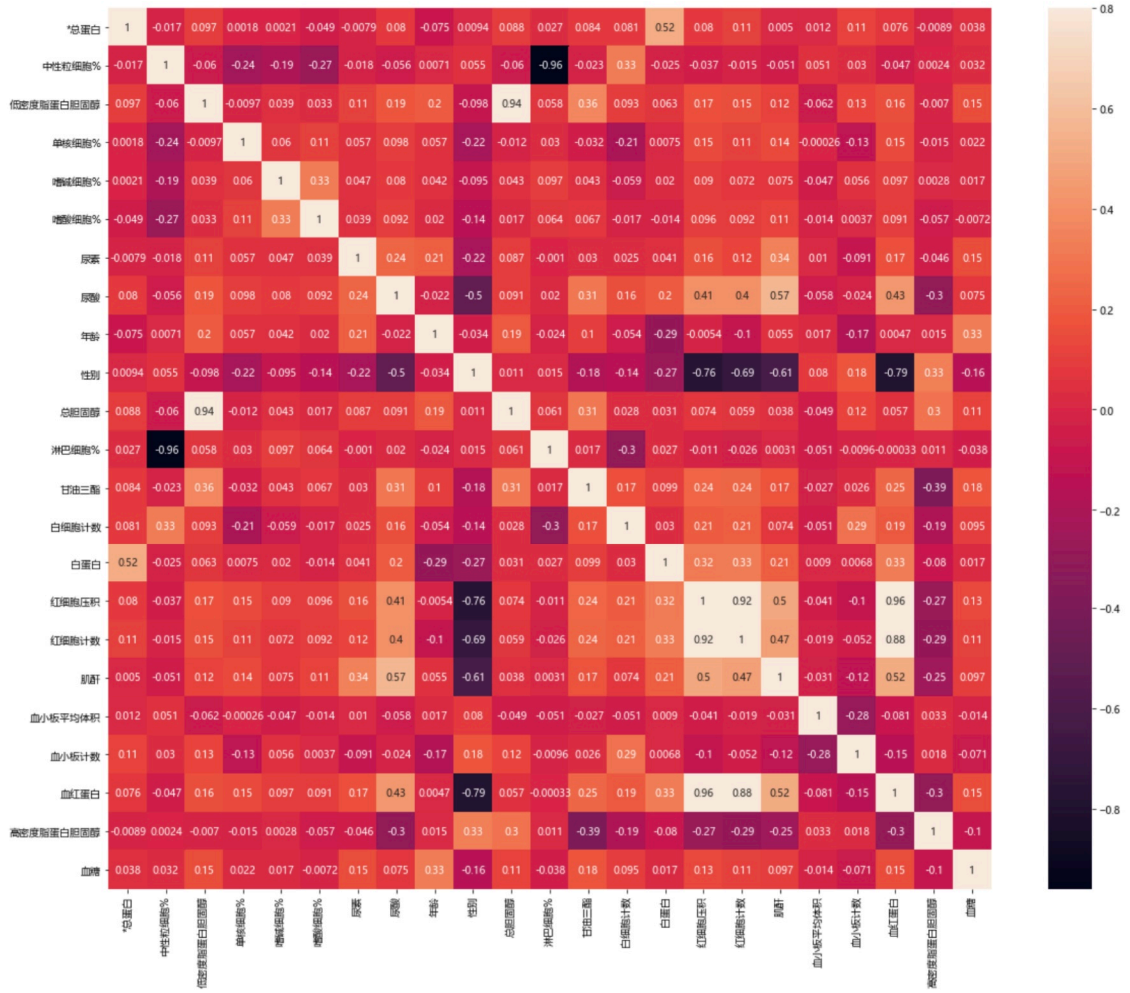

图 9　训练集与测试集中分布情况不一致的特征

由于难以确定选择指标的个数，我们在问题三中选定模型后进行了测试，最后发现当指标个数为 13 的时候有最好的训练效果。

六、 问题三模型的建立与求解

由于指标都是连续型数值变量，故此问题为回归预测求解。常见的回归模型有线性回归模型、岭回归模型、SVR 模型、随机森林模型等，我们将采用这些模型来预测目标值

K 值交叉验证进行模型选择

6.1 数据划分

我们将预处理过后训练数据按 6:2:2 划分为训练集、验证集和测试集。

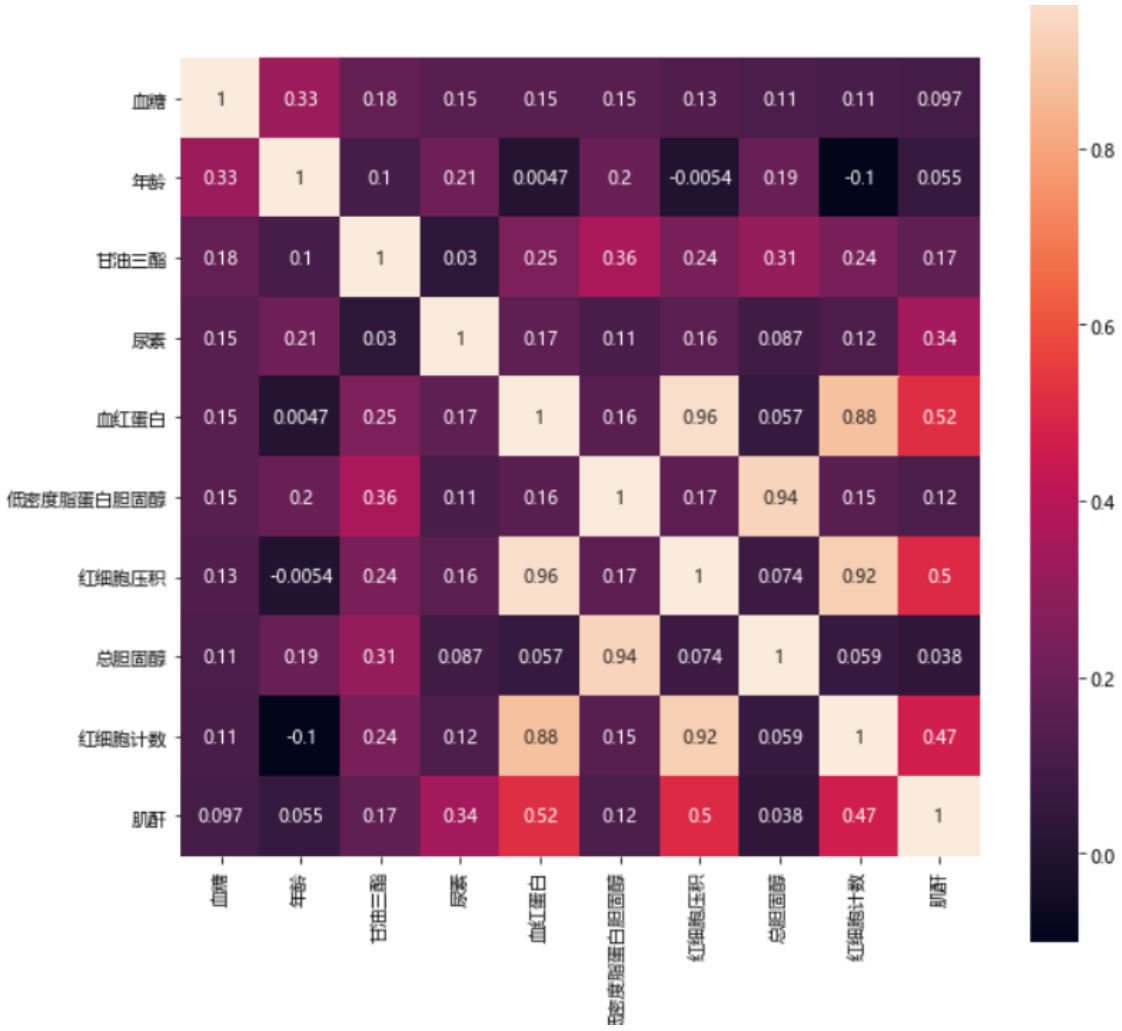

图 10　与”血糖”相关性最高的前 10 个指标

6.2 单模型测试

本题为回归问题，所以初步选定如下模型进行训练与验证：线性回归模型、岭回归模型、lasso 回归模型、SVR 模型、梯度提升树回归模型、随机森林模型、神经网络模型。在以上模型中选择最优模型。

使用训练集对以上 7 个模型训练过后，使用验证集中数据利用 K 折交叉验证方法选择最优模型，观察各个模型的 MSE,MSE_std, R^2, R_s^2td 指标值，结果如图 11。

MSE 最小，R^2 最大的模型其预测效果越好，于是我们选择岭回归模型作为最优模型进行后续验证与预测。我们使用划分好的测试集数据对岭回归模型进行验证，验证结果如图 12。

MSE,R^2 与验证集得出的结果相差无几，说明经过训练后的岭回归模型泛化能力较好。但由于模型的 R^2 只有 0.1261，所以我们认为只使用单模型进行预测的结果并不理想。

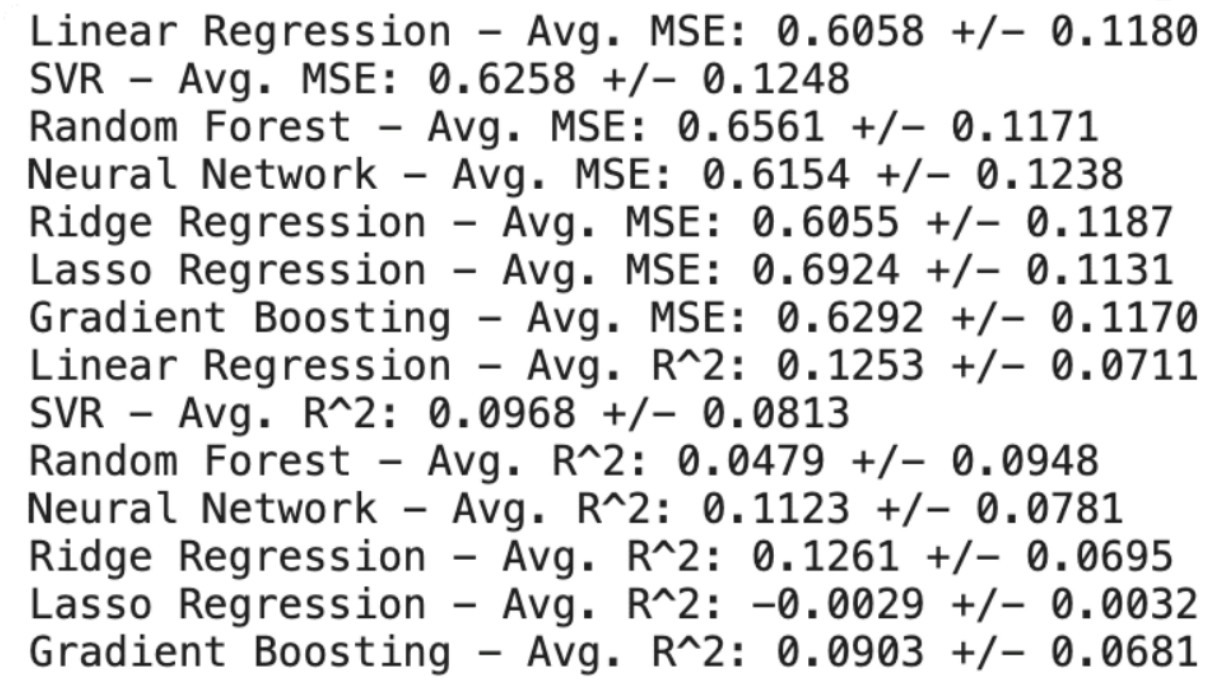

图 11　单模型训练结果

6.3 融合模型进行集成学习

由于仅仅选出单个最优模型效果并不理想，而使用融合模型可以提高预测性能，尤其在数据特征与目标变量之间的线性关系较差的情况下。融合模型可以综合多个单个模型的预测结果，通过组合它们的优势来提高整体性能。于是我们考虑将效果相对较好的几个模型进行集成学习，我们选定线性回归、岭回归、SVR、随机森林、梯度提升树模型进行融合。

我们使用堆叠 (Stacking) 的集成学习方法融合模型，不需要设定权重，所以我们将训练集和验证集合并成，即将原始训练数据划分为 8:2 的训练集与测试集。

使用训练集对各个模型训练后融合成 stacking 模型，得到该融合模型的验证结果如图 13。

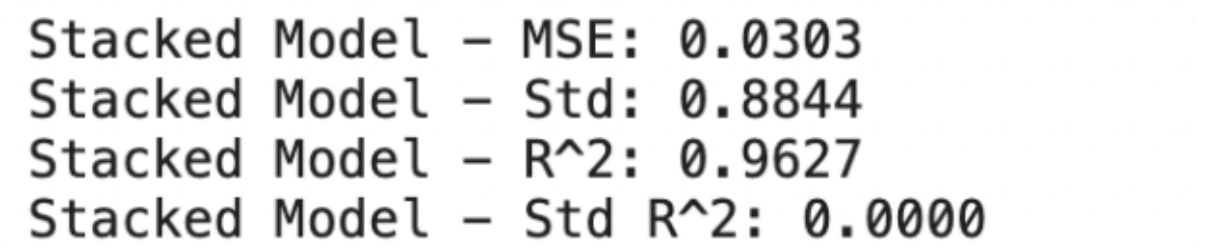

图 12　多模型训练结果

可以看出融合模型效果大大超过单模型的预测效果。

6.4 融合模型验证结果

随后利用划分的测试集对融合模型进行检验，检验结果如图 14. 由结果可得该融合

```
Stacked Model on Test Data - MSE: 0.0357
Stacked Model on Test Data - Std: 0.8814
Stacked Model on Test Data - R^2: 0.9561
Stacked Model on Test Data - Std R^2: 0.0000
```

图 13　融合模型验证结果

模型泛化能力较好，故对于本问，我们最终选择了由线性回归、岭回归、SVR、随机森林、梯度提升树模型集成学习后的融合模型。

七、 问题四模型的建立与求解

最终，我们将训练好的模型应用于测试数据集进行预测，并将预测结果添加到测试数据集中。随后，我们绘制了训练集和测试集中样本血糖的频率分布图进行观察。通过观察这些图形，我们发现采用模型融合的方法能够实现良好的回归预测效果。

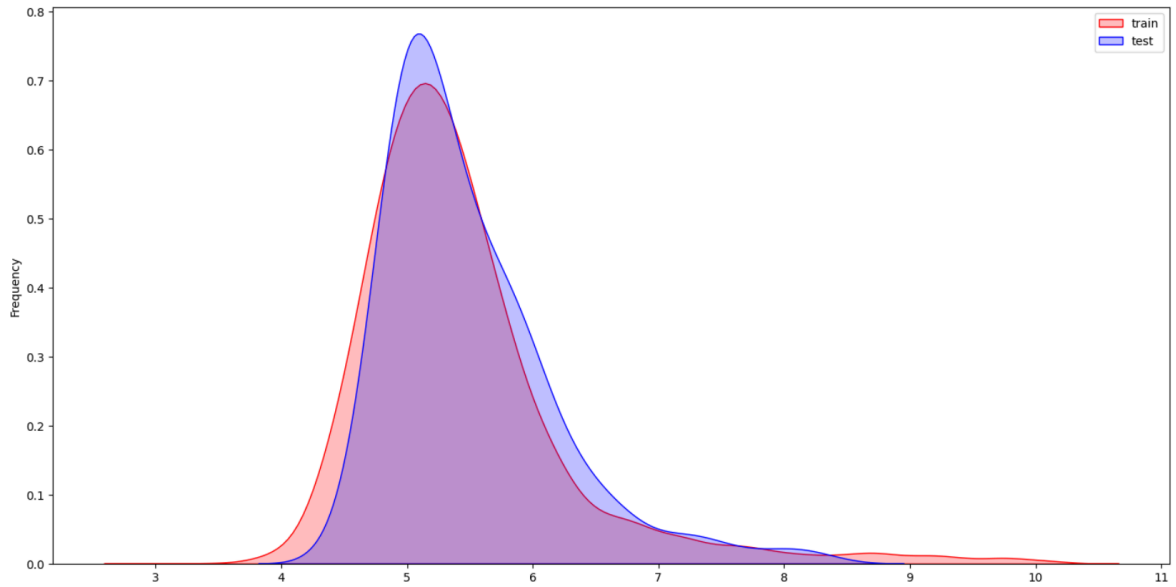

图 14　对附件 2 中的数据进行预测

图中的 train 是附件 1 中血糖的分布情况，test 是根据附件 2 中的数据进行预测后的数据。可以观察到二者的分布形态较为一致。

根据我们的预测结果，在测试数据集中，约 6% 的样本被预测为空腹血糖高于 6.7 毫摩尔每升的水平。这可以被视为患有糖尿病的风险较高的指标。

高血糖是糖尿病的一个主要特征，根据我们的预测结果，那些被预测为血糖高于6.7 毫摩尔每升的样本，可能具有较高的糖尿病风险。这些个体应该引起重视，并积极采取措施控制血糖水平，包括饮食调整、适度运动、保持健康体重以及定期监测血糖水平等。

参考文献

[1] 全国大学生数学建模竞赛论文格式规范 (2020 年 8 月 25 日修改).

附录 A　数据预处理源代码

```
import pandas as pd
import numpy as np
df = pd.read_csv('without_blood.csv',encoding='gbk')
df1 = pd.read_csv('with_blood.csv',encoding = 'gbk')
#df
df.info()

df = df.drop(['id','体检日期'],axis=1)
df1 = df1.drop(['id','体检日期'],axis=1)

# 使用 replace() 方法将 '男' 替换为 0, '女' 替换为 1
df['性别']=df['性别'].replace({'男':0,'女':1})
df['性别']

df1['性别']=df1['性别'].replace({'男':0,'女':1})
df1['性别'] = pd.to_numeric(df1['性别'], errors='coerce').fillna(0).astype(int)
df1['性别']

# 获取所有列名
columns = df.columns
columns1 = df1.columns

# 对所有列中的缺失值进行均值填充
df[columns] = df[columns].fillna(df[columns].mean())
df1[columns1] = df1[columns1].fillna(df1[columns].mean())
# print(df.head())
# print(df1.head())


# 计算每个特征的均值和标准差
means = df.mean()
stds = df.std()
means1 = df1.mean()
stds1 = df1.std()
df1_no_outliers = df1[(np.abs(df1 - means1) <= 3 * stds1).all(axis=1)]
df1 = df1_no_outliers
df1['血糖'].values
# 进行min-max标准化
from sklearn.preprocessing import MinMaxScaler

column_data = df1['血糖'].values
column = column_data.reshape(-1,1)

# 指定需要标准化的列
```

```
columns_to_normalize = [col for col in df1.columns if col != '血糖']

# 创建 MinMaxScaler 对象
scaler = MinMaxScaler()

# 对 DataFrame 进行 min-max 标准化
df_normalized = scaler.fit_transform(df)
df1_normalized = scaler.fit_transform(df1[columns_to_normalize])



# 将标准化后的数据转换回 DataFrame
df_normalized = pd.DataFrame(df_normalized, columns=df.columns)
df1[columns_to_normalize] = scaler.fit_transform(df1[columns_to_normalize])

df = df_normalized
df1 = df1_normalized
df1 = np.concatenate((df1, column), axis=1)

df.to_csv('test.csv', index=False)

column_names =
    ['*r-谷氨酰基转换酶','*丙氨酸氨基转换酶','*天门冬氨酸氨基转换酶','*总蛋白','*球蛋白','*碱性磷酸酶','中性粒细胞%



df1 = pd.DataFrame(df1)
df1.columns = column_names
df1.to_csv('train.csv', index=False)
```

附录 B　数据分析源代码

```
# 指定要丢弃的列
cols_to_drop = ['乙肝e抗体', '乙肝e抗原', '乙肝核心抗体','乙肝表面抗体','乙肝表面抗原']

# 使用drop()函数删除多列数据
df = df.drop(cols_to_drop, axis=1)



# 保存
df.to_csv('train.csv', index=False)

df = pd.read_csv('train.csv')
df.head()

import matplotlib.pyplot as plt
import seaborn as sns
```

```
# 解决坐标轴字体中文乱码
import matplotlib
matplotlib.rc("font", family='Microsoft YaHei')

# 画箱式图
column = df.columns.tolist()[:41] # 列表头
fig = plt.figure(figsize=(20, 40)) # 指定绘图对象宽度和高度
for i in range(40):
    plt.subplot(10, 4, i + 1) # 13行3列子图
    sns.boxplot(df[column[i]], orient="v", width=0.5) # 箱式图
    plt.ylabel(column[i], fontsize=8)
plt.show()
plt.savefig('./箱线图.png')

# 绘制正态分布图
train_cols = 6
train_rows = len(df.columns)
plt.figure(figsize=(4*train_cols,4*train_rows))

i=0
for col in df.columns:
    i+=1
    ax=plt.subplot(train_rows,train_cols,i)
    sns.distplot(df[col],fit=stats.norm)

    i+=1
    ax=plt.subplot(train_rows,train_cols,i)
    res = stats.probplot(df[col], plot=plt)
plt.show()

# 描述性统计信息
import pandas as pd
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
train.describe()

# 比较附件1与附件2的数据
import matplotlib
plt.rc("font", family='Microsoft YaHei')
dist_cols = 6
dist_rows = len(test.columns)

plt.figure(figsize=(4*dist_cols,4*dist_rows))


for i, col in enumerate(test.columns):
```

```
    ax=plt.subplot(dist_rows,dist_cols,i+1)
    ax = sns.kdeplot(train[col], color="Red", shade=True)
    ax = sns.kdeplot(test[col], color="Blue", shade=True)
    ax.set_xlabel(col)
    ax.set_ylabel("Frequency")
    ax = ax.legend(["train","test"])

plt.show()
# 导出为PNG图像文件
plt.savefig('./train_test.jpg')

drop_col = 12
drop_row = 1

plt.figure(figsize=(5*drop_col,5*drop_row))

i=1
for i, col in enumerate(
    ["*天门冬氨酸氨基转换酶","*丙氨酸氨基转换酶","*碱性磷酸酶","*r-谷氨酰基转换酶","*球蛋白","白球比例","红细胞平均
    ax =plt.subplot(drop_row,drop_col,i+1)
    ax = sns.kdeplot(train[col], color="Red", shade=True)
    ax = sns.kdeplot(test[col], color="Blue", shade=True)
    ax.set_xlabel(col)
    ax.set_ylabel("Frequency")
    ax = ax.legend(["train","test"])

plt.show()

# 丢掉分布不一致的列
cols_to_drop =
    ["*天门冬氨酸氨基转换酶","*丙氨酸氨基转换酶","*碱性磷酸酶","*r-谷氨酰基转换酶","*球蛋白","白球比例","红细胞平均
train = train.drop(cols_to_drop, axis=1)
test = test.drop(cols_to_drop, axis=1)

# 保存
train.to_csv('train.csv', index=False)
test.to_csv('test.csv', index=False)

# 检查线性分布情况
fcols = 6
frows = len(test.columns)
plt.figure(figsize=(5*fcols,4*frows))

i=0
for col in test.columns:
    i+=1
    ax=plt.subplot(frows,fcols,i)
```

```
    sns.regplot(x=col, y='血糖', data=train, ax=ax,
                scatter_kws={'marker':'.','s':3,'alpha':0.3},
                line_kws={'color':'k'});
    plt.xlabel(col)
    plt.ylabel('target')

    i+=1
    ax=plt.subplot(frows,fcols,i)
    sns.distplot(train[col].dropna())
    plt.xlabel(col)

# 相关系数计算
train_corr = train.corr()
train_corr

# 画出相关性热力图
ax = plt.subplots(figsize=(20, 16))#调整画布大小
ax = sns.heatmap(train_corr, vmax=.8, square=True, annot=True)#画热力图
annot=True #显示系数

#寻找K个最相关的特征信息
k = 10 # number of variables for heatmap
cols = train_corr.nlargest(k, '血糖')['血糖'].index

cm = np.corrcoef(train[cols].values.T)
hm = plt.subplots(figsize=(10, 10))#调整画布大小
#hm = sns.heatmap(cm, cbar=True, annot=True, square=True)
#g = sns.heatmap(train_data[cols].corr(),annot=True,square=True,cmap="RdYlGn")
hm = sns.heatmap(train[cols].corr(),annot=True,square=True)

plt.show()
```

附录 C　最优模型选择过程

```
import pandas as pd
import numpy as np

from sklearn import preprocessing
from sklearn.metrics import confusion_matrix
from sklearn import svm
from sklearn.model_selection import KFold
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
```

```
from sklearn.metrics import mean_squared_error
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from keras.models import Sequential
from keras.layers import Dense
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Lasso
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import r2_score

from imblearn.over_sampling import RandomOverSampler

import matplotlib.pyplot as plt
import matplotlib.mlab as mlab

import seaborn


%matplotlib inline


# 读入数据
data = pd.read_csv('/train.csv')
df = pd.DataFrame(data)

df = df.iloc[:10000]
#读取清洗后的数据，并且做统计分析。包括总值、平均值、标准差、最小值、1/4、1/2、3/4位数、最大值
df.describe()

# 将描述统计信息导出为表格（CSV 文件）
# description.to_csv("description.csv")


# 假设特征数据为X，目标变量为y
# 设置 y 值为第 39 列——血糖
Y = data.iloc[:, 39].values
column_names=['*r-谷氨酰基转换酶', '*丙氨酸氨基转换酶', '*天门冬氨酸氨基转换酶', '*碱性磷酸酶',
    '低密度脂蛋白胆固醇', '尿素', '年龄', '总胆固醇', '甘油三酯', '红细胞压积',
    '红细胞平均血红蛋白浓度', '红细胞计数', '血红蛋白']
X = data[column_names].values

# 将数据划分为训练集、验证集和测试集
X_train, X_val, Y_train, Y_val = train_test_split(X, Y, test_size=0.4, random_state=42)
X_val, X_test, Y_val, Y_test = train_test_split(X_val, Y_val, test_size=0.5, random_state=42)
```

```
### 训练集训练模型

# 创建线性回归模型
lr = LinearRegression()
# 训练模型
lr.fit(X_train, Y_train)


# 创建岭回归模型对象
ridge = Ridge()
# 训练模型
ridge.fit(X_train, Y_train)


# 创建Lasso回归模型对象
lasso = Lasso()
# 训练模型
lasso.fit(X_train, Y_train)


# 创建梯度提升树回归模型对象
gb = GradientBoostingRegressor()
# 训练模型
gb.fit(X_train, Y_train)


# 创建SVR对象
svr = SVR()
# 训练模型
svr.fit(X_train, Y_train)


# 创建RandomForestRegressor对象
rf = RandomForestRegressor()
# 训练模型
rf.fit(X_train, Y_train)


# 创建Sequential模型
NN = Sequential()
# 添加神经网络层
NN.add(Dense(units=64, activation='relu', input_dim=X_train.shape[1]))
NN.add(Dense(units=64, activation='relu'))
NN.add(Dense(units=1, activation='linear'))
# 编译模型
NN.compile(loss='mean_squared_error', optimizer='adam')
# 训练模型
```

```
NN.fit(X_train, Y_train, epochs=10, batch_size=32)



### K折交叉验证——验证集选择模型

# 定义交叉验证的折数
k = 5

# 创建 KFold 对象
kf = KFold(n_splits=k)


# 初始化结果列表
mse_scores_lr = []
mse_scores_svr = []
mse_scores_rf = []
mse_scores_nn = []
mse_scores_ridge = []
mse_scores_lasso = []
mse_scores_gb = []
r2_scores_lr = []
r2_scores_svr = []
r2_scores_rf = []
r2_scores_nn = []
r2_scores_ridge = []
r2_scores_lasso = []
r2_scores_gb = []

# 进行 k 折交叉验证
for train_index, val_index in kf.split(X_val):
    # 获取训练集和验证集
    x_train_fold = X_val[train_index]
    Y_train_fold = Y_val[train_index]
    x_val_fold = X_val[val_index]
    Y_val_fold = Y_val[val_index]

    # 使用线性回归模型进行预测
    y_pred_lr_fold = lr.predict(x_val_fold)
    # 计算线性回归模型的均方误差 (MSE)
    mse_lr_fold = mean_squared_error(Y_val_fold, y_pred_lr_fold)
    mse_scores_lr.append(mse_lr_fold)
    # 计算线性回归模型的R^2指标
    r2_lr_fold = r2_score(Y_val_fold, y_pred_lr_fold)
    r2_scores_lr.append(r2_lr_fold)

    # 使用岭回归模型进行预测
```

```
y_pred_ridge_fold = ridge.predict(x_val_fold)
# 计算岭回归模型的均方误差 (MSE)
mse_ridge_fold = mean_squared_error(Y_val_fold, y_pred_ridge_fold)
mse_scores_ridge.append(mse_ridge_fold)
# 计算岭回归模型的R^2指标
r2_ridge_fold = r2_score(Y_val_fold, y_pred_ridge_fold)
r2_scores_ridge.append(r2_ridge_fold)

# 使用支持向量回归模型进行预测
y_pred_svr_fold = svr.predict(x_val_fold)
# 计算支持向量回归模型的均方误差 (MSE)
mse_svr_fold = mean_squared_error(Y_val_fold, y_pred_svr_fold)
mse_scores_svr.append(mse_svr_fold)
# 计算支持向量回归模型的R^2指标
r2_svr_fold = r2_score(Y_val_fold, y_pred_svr_fold)
r2_scores_svr.append(r2_svr_fold)

# 使用随机森林模型进行预测
y_pred_rf_fold = rf.predict(x_val_fold)
# 计算随机森林模型的均方误差 (MSE)
mse_rf_fold = mean_squared_error(Y_val_fold, y_pred_rf_fold)
mse_scores_rf.append(mse_rf_fold)
# 计算随机森林模型的R^2指标
r2_rf_fold = r2_score(Y_val_fold, y_pred_rf_fold)
r2_scores_rf.append(r2_rf_fold)

# 使用已训练的神经网络模型进行预测
x_val_fold = x_val_fold.astype(float)
y_pred_nn_fold = NN.predict(x_val_fold)
# 计算神经网络模型的均方误差 (MSE)
mse_nn_fold = mean_squared_error(Y_val_fold, y_pred_nn_fold)
mse_scores_nn.append(mse_nn_fold)
# 计算神经网络模型的R^2指标
r2_nn_fold = r2_score(Y_val_fold, y_pred_nn_fold)
r2_scores_nn.append(r2_nn_fold)

# 使用Lasso回归模型进行预测
y_pred_lasso_fold = lasso.predict(x_val_fold)
# 计算Lasso回归模型的均方误差 (MSE)
mse_lasso_fold = mean_squared_error(Y_val_fold, y_pred_lasso_fold)
mse_scores_lasso.append(mse_lasso_fold)
# 计算Lasso回归模型的R^2指标
r2_lasso_fold = r2_score(Y_val_fold, y_pred_lasso_fold)
r2_scores_lasso.append(r2_lasso_fold)

# 使用梯度提升树回归模型进行预测
y_pred_gb_fold = gb.predict(x_val_fold)
```

```
    # 计算梯度提升树回归模型的均方误差 (MSE)
    mse_gb_fold = mean_squared_error(Y_val_fold, y_pred_gb_fold)
    mse_scores_gb.append(mse_gb_fold)
    # 计算梯度提升树回归模型的R^2指标
    r2_gb_fold = r2_score(Y_val_fold, y_pred_gb_fold)
    r2_scores_gb.append(r2_gb_fold)

# 计算线性回归模型的平均均方误差 (Avg. MSE) 和标准差 (Std. MSE)
avg_mse_lr = np.mean(mse_scores_lr)
std_mse_lr = np.std(mse_scores_lr)
# 计算线性回归模型的平均R^2指标和标准差
avg_r2_lr = np.mean(r2_scores_lr)
std_r2_lr = np.std(r2_scores_lr)

# 计算岭回归模型的平均均方误差 (Avg. MSE) 和标准差 (Std. MSE)
avg_mse_ridge = np.mean(mse_scores_ridge)
std_mse_ridge = np.std(mse_scores_ridge)
# 计算岭回归模型的平均R^2指标和标准差
avg_r2_ridge = np.mean(r2_scores_ridge)
std_r2_ridge = np.std(r2_scores_ridge)

# 计算支持向量回归模型的平均均方误差 (Avg. MSE) 和标准差 (Std. MSE)
avg_mse_svr = np.mean(mse_scores_svr)
std_mse_svr = np.std(mse_scores_svr)
# 计算支持向量回归模型的平均R^2指标和标准差
avg_r2_svr = np.mean(r2_scores_svr)
std_r2_svr = np.std(r2_scores_svr)

# 计算随机森林模型的平均均方误差 (Avg. MSE) 和标准差 (Std. MSE)
avg_mse_rf = np.mean(mse_scores_rf)
std_mse_rf = np.std(mse_scores_rf)
# 计算随机森林模型的平均R^2指标和标准差
avg_r2_rf = np.mean(r2_scores_rf)
std_r2_rf = np.std(r2_scores_rf)

# 计算神经网络模型的平均均方误差 (Avg. MSE) 和标准差 (Std. MSE)
avg_mse_nn = np.mean(mse_scores_nn)
std_mse_nn = np.std(mse_scores_nn)
# 计算神经网络模型的平均R^2指标和标准差
avg_r2_nn = np.mean(r2_scores_nn)
std_r2_nn = np.std(r2_scores_nn)

# 计算Lasso回归模型的平均均方误差 (Avg. MSE) 和标准差 (Std. MSE)
avg_mse_lasso = np.mean(mse_scores_lasso)
std_mse_lasso = np.std(mse_scores_lasso)
# 计算Lasso回归模型的平均R^2指标和标准差
avg_r2_lasso = np.mean(r2_scores_lasso)
```

```
std_r2_lasso = np.std(r2_scores_lasso)

# 计算梯度提升树回归模型的平均均方误差 (Avg. MSE) 和标准差 (Std. MSE)
avg_mse_gb = np.mean(mse_scores_gb)
std_mse_gb = np.std(mse_scores_gb)
# 计算梯度提升树回归模型的平均R^2指标和标准差
avg_r2_gb = np.mean(r2_scores_gb)
std_r2_gb = np.std(r2_scores_gb)

# 输出结果
print(f"Linear Regression - Avg. MSE: {avg_mse_lr:.4f} +/- {std_mse_lr:.4f}")
print(f"SVR - Avg. MSE: {avg_mse_svr:.4f} +/- {std_mse_svr:.4f}")
print(f"Random Forest - Avg. MSE: {avg_mse_rf:.4f} +/- {std_mse_rf:.4f}")
print(f"Neural Network - Avg. MSE: {avg_mse_nn:.4f} +/- {std_mse_nn:.4f}")
print(f"Ridge Regression - Avg. MSE: {avg_mse_ridge:.4f} +/- {std_mse_ridge:.4f}")
print(f"Lasso Regression - Avg. MSE: {avg_mse_lasso:.4f} +/- {std_mse_lasso:.4f}")
print(f"Gradient Boosting - Avg. MSE: {avg_mse_gb:.4f} +/- {std_mse_gb:.4f}")

print(f"Linear Regression - Avg. R^2: {avg_r2_lr:.4f} +/- {std_r2_lr:.4f}")
print(f"SVR - Avg. R^2: {avg_r2_svr:.4f} +/- {std_r2_svr:.4f}")
print(f"Random Forest - Avg. R^2: {avg_r2_rf:.4f} +/- {std_r2_rf:.4f}")
print(f"Neural Network - Avg. R^2: {avg_r2_nn:.4f} +/- {std_r2_nn:.4f}")
print(f"Ridge Regression - Avg. R^2: {avg_r2_ridge:.4f} +/- {std_r2_ridge:.4f}")
print(f"Lasso Regression - Avg. R^2: {avg_r2_lasso:.4f} +/- {std_r2_lasso:.4f}")
print(f"Gradient Boosting - Avg. R^2: {avg_r2_gb:.4f} +/- {std_r2_gb:.4f}")



### 对测试集运算，观察岭回归模型的能力

# 定义交叉验证的折数
k = 5

# 创建 KFold 对象
kf = KFold(n_splits=k)

# 初始化结果列表
mse_scores_ridge = []
r2_scores_ridge = []

# 进行 k 折交叉验证
for train_index, val_index in kf.split(X_test):
    # 获取训练集和验证集
    x_train_fold = X_val[train_index]
    Y_train_fold = Y_val[train_index]
    x_val_fold = X_val[val_index]
    Y_val_fold = Y_val[val_index]
```

```
    # 使用岭回归模型进行预测
    y_pred_ridge_fold = ridge.predict(x_val_fold)

    # 计算岭回归模型的均方误差 (MSE)
    mse_ridge_fold = mean_squared_error(Y_val_fold, y_pred_ridge_fold)
    mse_scores_ridge.append(mse_ridge_fold)

    # 计算岭回归模型的R^2指标
    r2_ridge_fold = r2_score(Y_val_fold, y_pred_ridge_fold)
    r2_scores_ridge.append(r2_ridge_fold)


# 计算岭回归模型的平均均方误差 (Avg. MSE) 和标准差 (Std. MSE)
avg_mse_ridge = np.mean(mse_scores_ridge)
std_mse_ridge = np.std(mse_scores_ridge)

# 计算岭回归模型的平均R^2指标和标准差
avg_r2_ridge = np.mean(r2_scores_ridge)
std_r2_ridge = np.std(r2_scores_ridge)

# 输出结果
print(f"Ridge Regression - Avg. MSE: {avg_mse_ridge:.4f}")
print(f"Ridge Regression - Avg. MSE: {avg_mse_ridge:.4f} +/- {std_mse_ridge:.4f}")
print(f"Ridge Regression - Avg. R^2: {avg_r2_ridge:.4f}")
print(f"Ridge Regression - Avg. R^2: {avg_r2_ridge:.4f} +/- {std_r2_ridge:.4f}")
```

附录 D　数据分析源代码

```
kk=2;
[mdd,ndd]=size(dd);
while ~isempty(V)
    [tmpd,j]=min(W(i,V));tmpj=V(j);
for k=2:ndd
    [tmp1,jj]=min(dd(1,k)+W(dd(2,k),V));
    tmp2=V(jj);tt(k-1,:)=[tmp1,tmp2,jj];
end
    tmp=[tmpd,tmpj,j;tt];[tmp3,tmp4]=min(tmp(:,1));
if tmp3==tmpd, ss(1:2,kk)=[i;tmp(tmp4,2)];
else,tmp5=find(ss(:,tmp4)~=0);tmp6=length(tmp5);
if dd(2,tmp4)==ss(tmp6,tmp4)
    ss(1:tmp6+1,kk)=[ss(tmp5,tmp4);tmp(tmp4,2)];
else, ss(1:3,kk)=[i;dd(2,tmp4);tmp(tmp4,2)];
end;
end
```

```
    dd=[dd,[tmp3;tmp(tmp4,2)]];V(tmp(tmp4,3))=[];
    [mdd,ndd]=size(dd);
    kk=kk+1;
end;
S=ss;
D=dd(1,:);
```

附录 E　数据分析源代码

```
kk=2;
[mdd,ndd]=size(dd);
while ~isempty(V)
    [tmpd,j]=min(W(i,V));tmpj=V(j);
for k=2:ndd
    [tmp1,jj]=min(dd(1,k)+W(dd(2,k),V));
    tmp2=V(jj);tt(k-1,:)=[tmp1,tmp2,jj];
end
    tmp=[tmpd,tmpj,j;tt];[tmp3,tmp4]=min(tmp(:,1));
if tmp3==tmpd, ss(1:2,kk)=[i;tmp(tmp4,2)];
else,tmp5=find(ss(:,tmp4)~=0);tmp6=length(tmp5);
if dd(2,tmp4)==ss(tmp6,tmp4)
    ss(1:tmp6+1,kk)=[ss(tmp5,tmp4);tmp(tmp4,2)];
else, ss(1:3,kk)=[i;dd(2,tmp4);tmp(tmp4,2)];
end;
end
    dd=[dd,[tmp3;tmp(tmp4,2)]];V(tmp(tmp4,3))=[];
    [mdd,ndd]=size(dd);
    kk=kk+1;
end;
S=ss;
D=dd(1,:);
```