

**2024年《机器学习》**

**工程报告**

**（个人版）**



**课 程：** 机器学习

**姓 名：** 高培骏

**学 号：** 2022217509

**完成时间：** 2024年5月18日

|  |  |  |  |
| --- | --- | --- | --- |
| **“机器学习-大作业”评分细则** | | | |
| **成绩等级** | **具体表现** | **教师评分** | |
| 优秀（100-90] | 报告撰写优秀，题目本身难度大，工作量饱满；Notebook代码清晰准确，实验与可视化优秀 | □ |  |
| 良好（90-80] | 报告撰写良好，题目本身难度较大，工作量较多；Notebook代码较好，实验与可视化良好 | □ |  |
| 中等（80-70] | 报告撰写中等，题目本身难度一般，工作量达标；Notebook代码一般，实验与可视化达标 | □ |  |
| 及格（70-60] | 能完成基本要求，工作量较少，非Notebook代码 | □ |  |
| 不及格（<60） | 未达最低要求或抄袭线上线下资源 | □ |  |

教师签名：

一． 自己所作工作的简介

摘要：介绍自己在本项目中的具体工作，比如分工的具体内容，使用方法，如何设计，实验结果与分析等。

在本次的机器学习的大作业中，我完成的主要具体内容有：

1. 实现对数据的预处理，主要的操作有缺失值的清洗、重复值的清洗、异常值的清洗、数据归一化以及对应的可视化功能。
2. 我使用了**7**种方法实现了对数据的分类：分别是随机森林模型、AdaBoost模型、XGBoost模型、朴素贝叶斯模型、K最近邻模型、多层感知机模型、以及支持向量机模型。
3. 在本次的大作业中，我首先对数据集进行了清洗，并实现了数据的可视化，然后基于清洗后的数据集，划分为训练集和测试集两部分，分别带入到七种方法中，比较七种方法，得到相对的最优结果。
4. 七种方法对数据的预测结果如下，准确度从高到低排序为**XGBoost、AdaBoost、随机森林、支持向量机、多层感知机、K最近邻、高斯朴素贝叶斯**。

二、研究背景与意义

说明本工程所涉及的研究背景研究的意义，包括主要涉及领域，主要研究方法，主要存在问题，现有解决方案等等，可以自行补充其他内容。

1. **主要存在问题：**

我国关于胎儿健康存在的问题有很多，比如中国面临着先天性畸形和遗传疾病高发的情况，这些疾病可能会对胎儿的生命和生活质量产生长期影响。因此，我们亟需寻找到能够判断胎儿健康状态的方案，比如推广和普及产前诊断技术，如无创产前基因检测（NIPT）、胎儿超声检查等。

1. **主要研究方法：**

主要使用机器学习中的分类模型来实现对胎儿健康状态的预测，分为fetal\_health=1、fetal\_health=2、fetal\_health=3三类，主要使用到的分类模型有随机森林模型、AdaBoost模型、XGBoost模型、朴素贝叶斯模型、K最近邻模型、多层感知机模型、以及支持向量机模型。

**3、 研究意义：**

1. 早期干预：通过预测胎儿健康状况，医生和医疗团队可以在孕期或胎儿发育的早期阶段采取必要的干预措施。这些干预措施可以包括调整母亲的生活方式、药物治疗、手术干预等，以提高胎儿的健康状况和生存率。
2. 降低孕产妇焦虑：对于孕产妇来说，了解胎儿的健康状况可以减少焦虑和不确定性。通过胎儿健康预测，医生可以向孕产妇提供及时的信息和建议，帮助她们更好地应对孕期和分娩过程。
3. 减少新生儿死亡率：胎儿健康预测可以帮助识别高风险的胎儿，及早采取措施降低新生儿死亡率。通过及时诊断和治疗胎儿可能存在的健康问题，可以减少因疾病或先天性异常导致的新生儿死亡。
4. 优化医疗资源分配：在资源有限的医疗环境下，胎儿健康预测可以帮助医疗机构和医生更有效地分配医疗资源。通过识别高风险的胎儿，医生可以将更多的资源和关注重点放在这些患者身上，从而提高医疗效率和治疗成功率。
5. 促进科学研究：胎儿健康预测的研究不仅可以促进医学技术的进步，还可以为科学家们提供更深入的理解胎儿发育和健康的机制。这些研究成果有助于开发新的诊断工具、治疗方法和预防措施，为未来的医学实践提供更多的可能性。

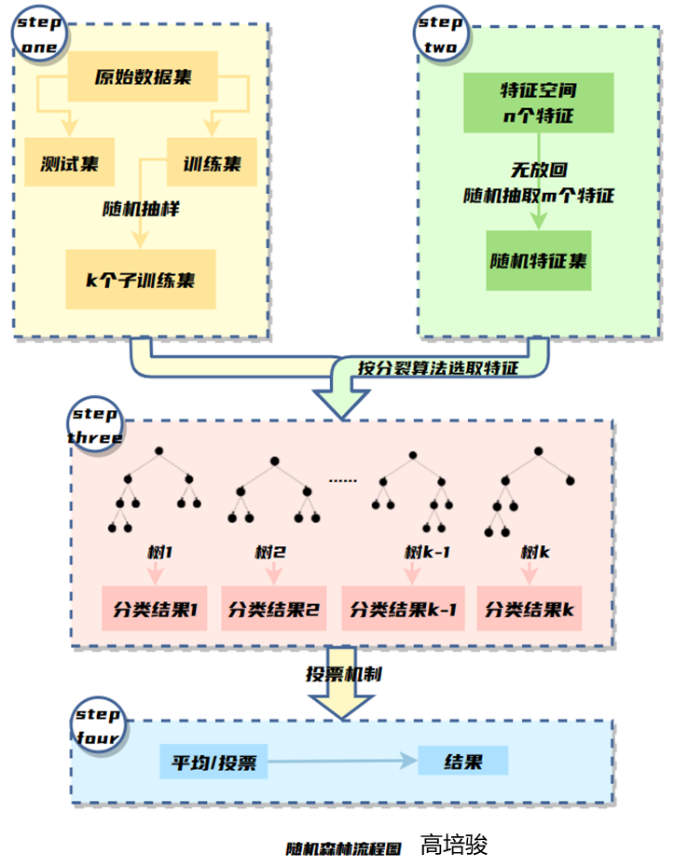
三、模型方法

本项目所涉及到的模型有随机森林模型、AdaBoost模型、XGBoost模型、高斯朴素贝叶斯模型、K最近邻模型、多层感知机模型、支持向量机模型。

1. 随机森林模型：
   1. 原理

随机森林的出现主要是为了解单一决策树可能出现的很大误差和overfitting的问题。这个算法的核心思想就是将多个不同的决策树进行组合，利用这种组合降低单一决策树有可能带来的片面性和判断不准确性。用我们常说的话来形容这个思想就是“三个臭皮匠赛过诸葛亮”。

具体来讲，随机森林是用随机的方式建立一个森林，这个随机性表述的含义我们接下来会讲。随机森林是由很多的决策树组成，但每一棵决策树之间是没有关联的。在得到森林之后，当对一个新的样本进行判断或预测的时候，让森林中的每一棵决策树分别进行判断，看看这个样本应该属于哪一类（对于分类算法），然后看看哪一类被选择最多，就预测这个样本为那一类。



**随机森林模型流程图(自制，非可视化的结果)**

* 1. 优点：

随机森林能够提供较高的预测准确性，尤其适用于处理高维数据和复杂数据集；由于其并行计算的能力和特征选择，随机森林可以高效地处理大规模数据，减少训练时间；随机森林通过随机选择特征子集进行节点划分，可以有效处理高维特征空间，降低数据维度。

* 1. 缺点：

训练多个决策树模型并进行预测需要较高的计算复杂度，特别是在处理大规模数据集时。由于需要存储多个决策树模型，随机森林算法需要大量的内存空间。

1. AdaBoost模型
   1. 原理：

Adaboost算法是Boosting算法的一种，基本原理就是将多个弱分类器（弱分类器一般选用单层决策树）进行合理的结合，使其成为一个强分类器。Adaboost采用迭代的思想，每次迭代只训练一个弱分类器，训练好的弱分类器将参与下一次迭代的使用。也就是说，在第N次迭代中，一共就有N个弱分类器，其中N-1个是以前训练好的，其各种参数都不再改变，本次训练第N个分类器。其中弱分类器的关系是第N个弱分类器更可能分对前N-1个弱分类器没分对的数据，最终分类输出要看这N个分类器的综合效果。

* 1. 优点：

可以在AdaBoost框架下使用各种回归分类模型来构建弱学习器，不限制弱学习器的种类，可以使用不同的学习算法来构建弱分类器。AdaBoost具有很高的精度，训练误差以指数速率下降。由于AdaBoost允许使用简单的弱分类器，不需要筛选特征，很少发生过拟合现象。

* 1. 缺点：

数据不平衡可能导致分类精度下降。训练比较耗时，每次需要重新选择当前分类器最好的切分点。AdaBoost对异常样本敏感，异常样本在迭代中可能会获得较高的权重，影响最终的强学习器的预测准确性。

1. XGBoost模型：
2. 原理：

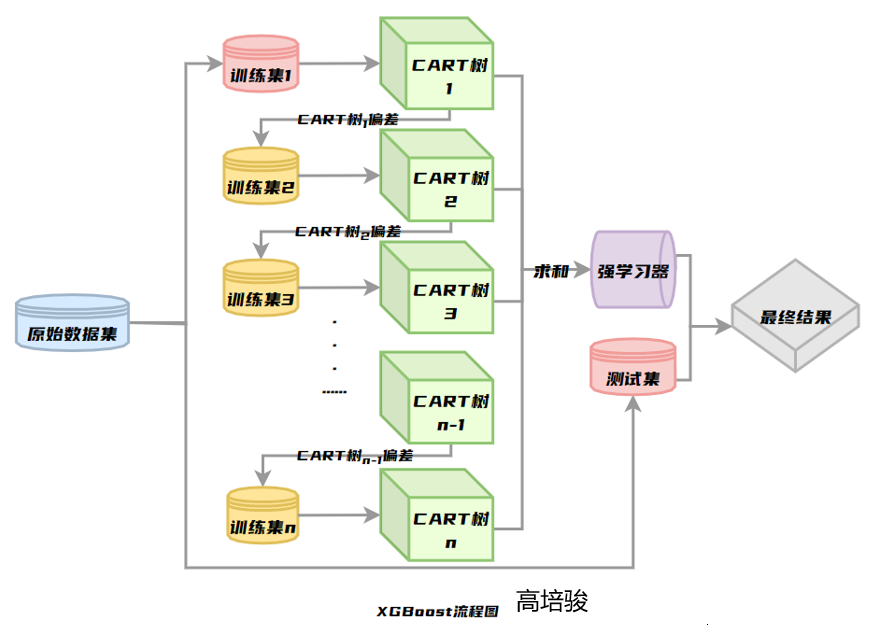
与AdaBoost类似，XGBoost算法也是Boosting算法的一种。但与AdaBoost不同的是，其所用到的树模型是CART回归树模型。XGBoost算法思想就是不断地添加树，不断地进行特征分裂来生长一棵树，每次添加一个树，其实是学习一个新函数，去拟合上次预测的残差。当我们训练完成得到k棵树，我们要预测一个样本的分数，其实就是根据这个样本的特征，在每棵树中会落到对应的一个叶子节点，每个叶子节点就对应一个分数，最后只需要将每棵树对应的分数加起来就是该样本的预测值。

CART回归树是假设树为二叉树，通过不断将特征进行分裂。比如当前树结点是基于第j个特征值进行分裂的，设该特征值小于s的样本划分为左子树，大于s的样本划分为右子树。

CART回归树实质上就是在该特征维度对样本空间进行划分，而这种空间划分的优化是一种NP难问题，因此，在决策树模型中是使用启发式方法解决。典型CART回归树产生的目标函数为：

因此，当我们为了求解最优的切分特征j和最优的切分点s，就转化为求解这么一个目标函数：

所以我们只要遍历所有特征的的所有切分点，就能找到最优的切分特征和切分点。最终得到一棵回归树。



**XGBoost模型流程图(自制，非可视化的结果)**

1. 优点：

XGBoost通过迭代地优化模型，通常在预测精度上优于其他算法。使用并行计算和缓存技术，XGBoost能够快速训练大规模数据集。支持用户自定义目标函数和评估函数，只要目标函数二阶可导。允许在每一轮Boosting迭代中使用交叉验证，方便获得最优Boosting迭代次数。基于决策树集成，XGBoost具有较好的可解释性，能够输出每个特征的重要性程度。

1. 缺点：

XGBoost有多个参数需要调优，不同的参数组合可能会导致不同的效果。在处理大规模数据时可能会占用较大的内存空间。在模型过于复杂或数据噪声较大时，可能会出现过拟合的情况。

1. 朴素贝叶斯模型：
   1. 原理：

先验概率：即基于统计的概率，是基于以往历史经验和分析得到的结果，不需要依赖当前发生的条件。

后验概率：则是从条件概率而来，由因推果，是基于当下发生了事件之后计算的概率，依赖于当前发生的条件。

条件概率：记事件A发生的概率为P(A)，事件B发生的概率为P(B)，则在B事件发生的前提下，A事件发生的概率即为条件概率，记为P(A|B)。

贝叶斯公式：贝叶斯公式便是基于条件概率，通过来求，如下：

将A看成“规律”，B看成“现象”，那么贝叶斯公式看成：

全概率公式：表示若事件构成一个完备事件组且都有正概率，则对任意一个事件都有公式成立：

特征条件假设：假设每个特征之间没有联系，给定训练数据集，其中每个样本x都包括n维特征，即，类标记集合含有k种类别，即。

对于给定的新样本x，判断其属于哪个标记的类别，根据贝叶斯定理，可以得到x属于类别的概率

后验概率最大的类别记为预测类别，即：。

朴素贝叶斯算法对条件概率分布作出了独立性的假设，通俗地讲就是说假设各个维度的特征互相独立，在这个假设的前提上，条件概率可以转化为：

代入上面贝叶斯公式中，得到：

于是，朴素贝叶斯分类器可表示为：

因为对所有的y\_k，上式中的分母的值都是一样的，所以可以忽略分母部分，朴素贝叶斯分类器最终表示为：

* 1. 优点：

朴素贝叶斯算法有很强的泛化能力，能够快速地处理大量的数据。有很好的可解释性，几乎可以看到它的计算过程，它是一种简单有效的算法。能自动找到与样本最相关的特征，可以方便地处理多类别问题。不需要构建复杂的数据结构，它的计算速度也很快。

* 1. 缺点：

对于数据的准备有一定的要求，它要求样本的特征之间是条件独立的，这样朴素贝叶斯算法才能得出正确的结果。忽略了数据的局部结构，它假设样本特征之间是独立的，但实际情况往往不是这样。结果可能会受到噪声和失误影响，使它的结果变得不准确。

1. K近邻模型：
   1. 原理：

* 选择一个适当的距离度量标准，通常使用欧氏距离或曼哈顿距离来度量数据点之间的距离。
* 给定一个要预测的数据点，计算它与训练数据集中所有数据点之间的距离。
* 根据距离度量，选择与预测数据点最近的K个邻居。
* 对于分类问题，根据这K个邻居中最常见的类别来预测目标数据点的类别。通常采用多数表决的方式，即取K个邻居中出现最频繁的类别作为预测结果。对于回归问题，可以取K个邻居的平均值或加权平均值作为预测结果。

欧氏距离（Euclidean Distance）：也称为直线距离，表示两点之间的最短距离。在二维空间中，欧氏距离计算公式为：

曼哈顿距离（Manhattan Distance）： 表示两点之间沿坐标轴的距离总和。在二维空间中，曼哈顿距离计算公式为：

K值的选择

* K值是KNN算法中需要谨慎选择的参数。选择不同的K值可能会导致不同的预测结果。一般来说：1.较小的K值会使模型对噪声敏感，产生波动较大的预测结果。
* 较大的K值会使模型更加稳定，但可能会忽略局部细节。
* 选择K值的方法包括交叉验证、网格搜索和启发式方法。根据问题的复杂性和数据的分布，选择合适的K值是关键的。
  1. 优点：

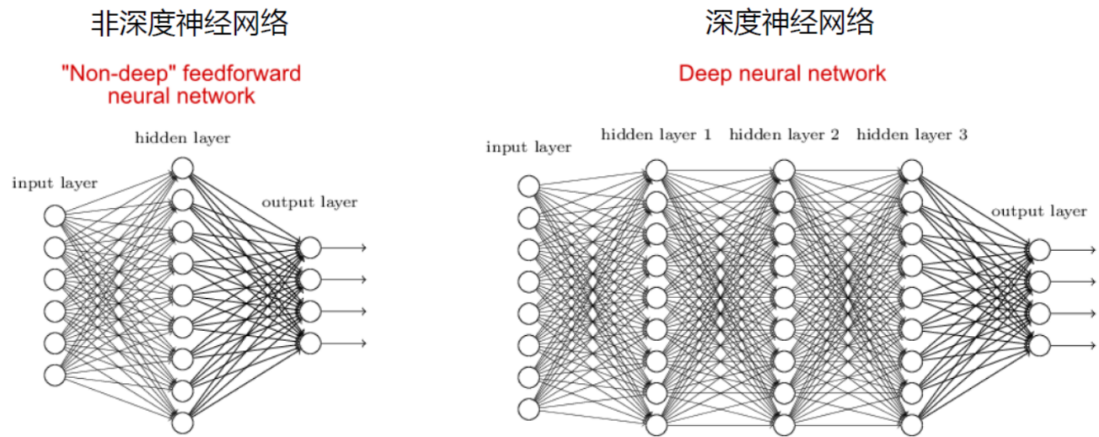
k近邻算法是一种在线技术，新数据可以直接加入数据集而不必进行重新训练。理论简单，容易实现。准确性高，对异常值和噪声有较高的容忍度。天生就支持多分类，区别与感知机、逻辑回归、SVM。

* 1. 缺点：

基本的 k近邻算法每预测一个“点”的分类都会重新进行一次全局运算，对于样本容量大的数据集计算量比较大。而且K近邻算法容易导致维度灾难，在高维空间中计算距离的时候，就会变得非常远；样本不平衡时，预测偏差比较大，k值大小的选择得依靠经验或者交叉验证得到。k的选择可以使用交叉验证，也可以使用网格搜索。k的值越大，模型的偏差越大，对噪声数据越不敏感。

1. 多层感知机模型：
   1. 原理：

多层感知机是一种前向人工神经网络，由多层神经元组成。它的基本结构包括输入层、隐藏层和输出层。每一层都由多个神经元组成，其中隐藏层可以有多个。多层感知机的每个神经元都与上一层的所有神经元相连，通过权重和激活函数来进行信息传递和处理。



**多层感知机网络架构(网图，非可视化结果)**

* 1. 优点：

具有较强的表达能力，能够处理非线性问题和高维数据，通过反向传播算法进行训练，可以自动学习特征和模式，适用于处理多分类问题和回归问题，具有较好的泛化能力。MLP还可以通过添加正则化项、dropout等技术来防止过拟合。

* 1. 缺点：

训练时间较长，需要大量的计算资源和时间。对初始权重和偏置的选择比较敏感，可能会导致模型陷入局部最优解。对数据的标准化和预处理要求较高，需要进行归一化、标准化等处理。难以解释和理解，不如决策树等模型具有可解释性。

1. 支持向量机模型：
   1. 原理：

SVM学习的基本想法是求解能够正确划分训练数据集并且几何间隔最大的分离超平面。如下图所示，即为分离超平面，对于线性可分的数据集来说，这样的超平面有无穷多个（即感知机），但是几何间隔最大的分离超平面却是唯一的。

在推导之前，先给出一些定义。假设给定一个特征空间上的训练数据集

其中，，为第个特征向量，为类标记，当它等于+1时为正例；为-1时为负例。再假设训练数据集是线性可分的。

几何间隔：对于给定的数据集和超平面，定义超平面关于样本点的几何间隔为

超平面关于所有样本点的几何间隔的最小值为

实际上这个距离就是我们所谓的支持向量到超平面的距离。

根据以上定义，SVM模型的求解最大分割超平面问题可以表示为以下约束最优化问题

将约束条件两边同时除以，得到

因为都是标量，所以为了表达式简洁起见，令

得到

又因为最大化，等价于最大化，也就等价于最小化是为了后面求导以后形式简洁，不影响结果），因此SVM模型的求解最大分割超平面问题又可以表示为以下约束最优化问题

这是一个含有不等式约束的凸二次规划问题，可以对其使用拉格朗日乘子法得到其对偶问题（dual problem）。

首先，我们将有约束的原始目标函数转换为无约束的新构造的拉格朗日目标函数

其中为拉格朗日乘子，且。现在我们令

当样本点不满足约束条件时，即在可行解区域外：

此时，将设置为无穷大，则也为无穷大。

当满本点满足约束条件时，即在可行解区域内：

此时，为原函数本身。于是，将两种情况合并起来就可以得到我们新的目标函数

于是原约束问题就等价于

看一下我们的新目标函数，先求最大值，再求最小值。这样的话，我们首先就要面对带有需要求解的参数和的方程，而又是不等式约束，这个求解过程不好做。所以，我们需要使用拉格朗日函数对偶性，将最小和最大的位置交换一下，这样就变成了：

要有，需要满足两个条件：

① 优化问题是凸优化问题

② 满足KKT条件

首先，本优化问题显然是一个凸优化问题，所以条件一满足，而要满足条件二，即要求

为了得到求解对偶问题的具体形式，令对和的偏导为0，可得

将以上两个等式带入拉格朗日目标函数，消去和，得

即

求的极大，即是对偶问题

把目标式子加一个负号，将求解极大转换为求解极小

现在我们的优化问题变成了如上的形式。对于这个问题，我们有更高效的优化算法，即序列最小优化（SMO）算法。这里暂时不展开关于使用SMO算法求解以上优化问题的细节，下一篇文章再加以详细推导。

我们通过这个优化算法能得到，再根据，我们就可以求解出和，进而求得我们最初的目的：找到超平面，即”决策平面”。

前面的推导都是假设满足KKT条件下成立的，KKT条件如下

另外，根据前面的推导，还有下面两个式子成立

由此可知在中，至少存在一个（反证法可以证明，若全为0，则，矛盾），对此有

因此可以得到

对于任意训练样本，总有或者 $ 。若，则该样本不会在最后求解模型参数的式子中出现。若，则必有，所对应的样本点位于最大间隔边界上，是一个支持向量。这显示出支持向量机的一个重要性质：训练完成后，大部分的训练样本都不需要保留，最终模型仅与支持向量有关。

到这里都是基于训练集数据线性可分的假设下进行的，但是实际情况下几乎不存在完全线性可分的数据，为了解决这个问题，引入了“软间隔”的概念，即允许某些点不满足约束

采用hinge损失，将原优化问题改写为

其中为“松弛变量”， ，即一个hinge损失函数。每一个样本都有一个对应的松弛变量，表征该样本不满足约束的程度。称为惩罚参数，值越大，对分类的惩罚越大。跟线性可分求解的思路一致，同样这里先用拉格朗日乘子法得到拉格朗日函数，再求其对偶问题。

综合以上讨论，我们可以得到线性支持向量机学习算法如下：

输入：训练数据集其中，；

输出：分离超平面和分类决策函数

（1）选择惩罚参数，构造并求解凸二次规划问题

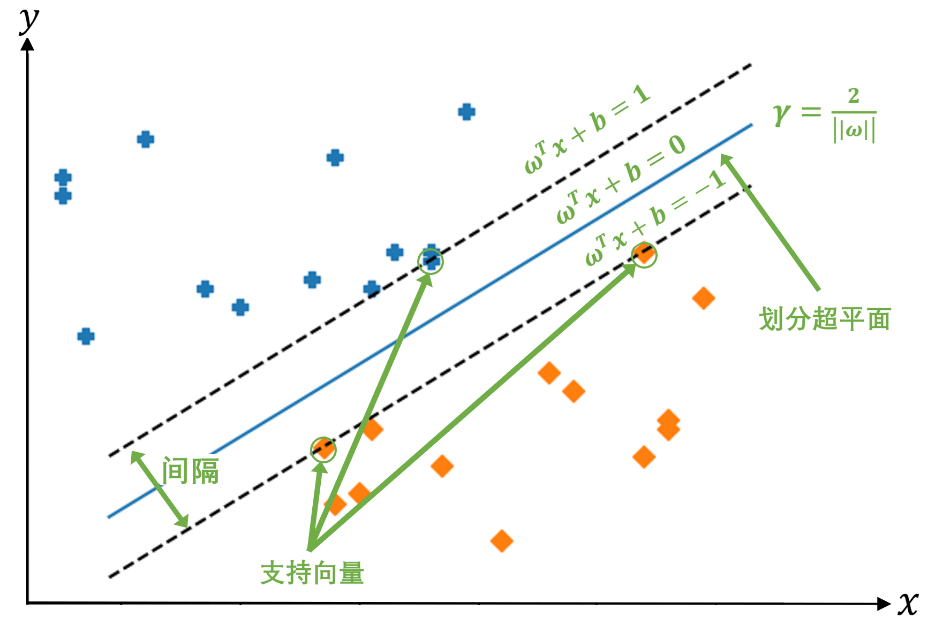
得到最优解

（2）计算

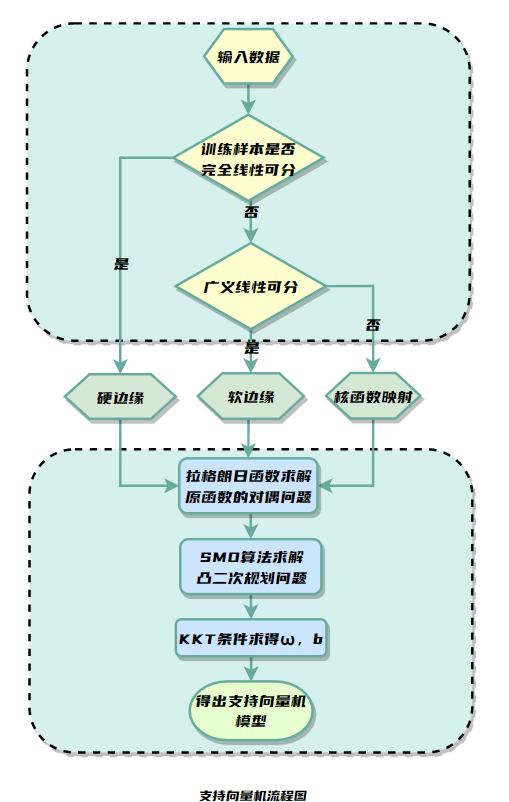
选择的一个分量满足条件，计算

（3）求分离超平面

分类决策函数：



**支持向量机原理图(手绘，非可视化结果)**

****

**支持向量机流程图(手绘，非可视化结果)**

* 1. 优点：

SVM 在高维空间中非常高效，即使数据维度比样本数量大时仍能保持有效。最终决策函数主要由少数支持向量决定，而不是样本空间的维数，这在一定程度上避免了“维数灾难”。在决策函数中使用训练集的子集，因此能够高效利用内存。具有较强的泛化能力，相对于其他算法如神经网络，SVM 无局部极小值问题。

* 1. 缺点：

对大规模训练样本难以实施，特别是当样本数量很大时，求解过程涉及大量矩阵计算，导致内存和运算时间消耗大。经典的 SVM 算法只给出二类分类的算法，解决多类分类问题存在困难。对大规模训练样本难以实施，特别是当样本数量很大时，求解过程涉及大量矩阵计算，导致内存和运算时间消耗大。

四、系统设计

系统的**详细**设计，系统流程，系统的每一步的具体流程，例如，如何处理语料，如何训练模型，如何测试模型，最后对模型进行评估等。

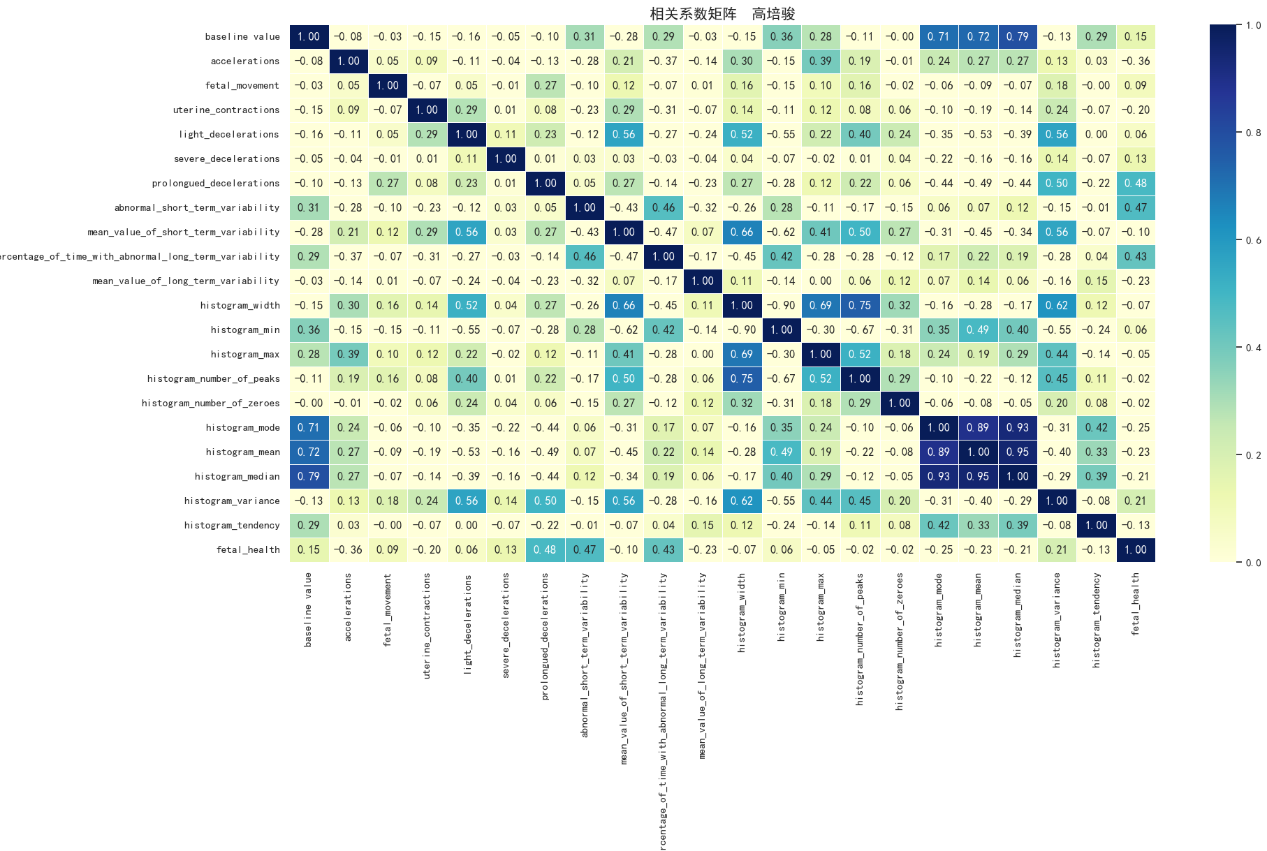
1. 数据预处理

处理前数据可视化：

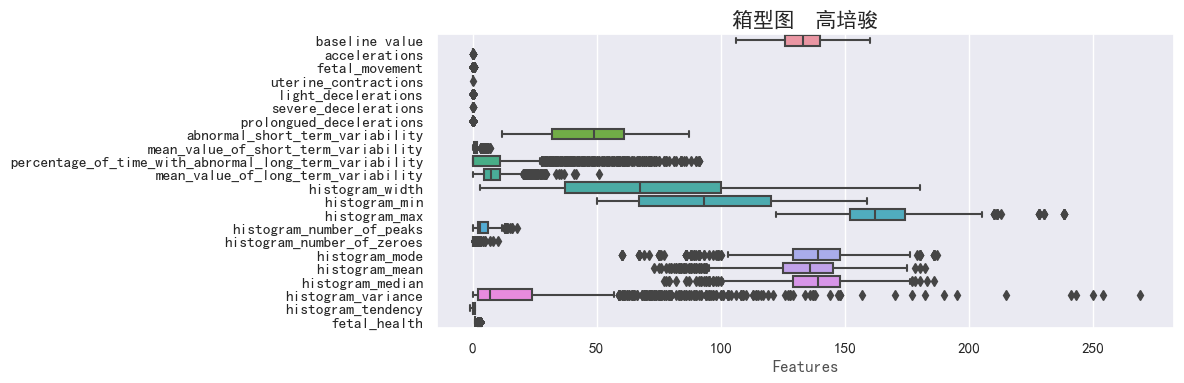
1. 处理前特征直方图：



1. 处理前相关系数矩阵



1. 处理前箱型图：



在使用这些数据之前，我们需要对数据进行预处理，具体的操作有：缺失值处理、重复值处理、异常值处理以及归一化处理。

print(np.isnan(data).any())# 查看是否存在缺失值  
# 因为不存在缺失值，所以直接跳过缺失值处理

data\_no\_duplicate = data.drop\_duplicates(keep='first')

data\_no\_error = drop\_outlier(data\_no\_duplicate, 4)  
data\_no\_error = data\_no\_error.drop('severe\_decelerations', axis=1)

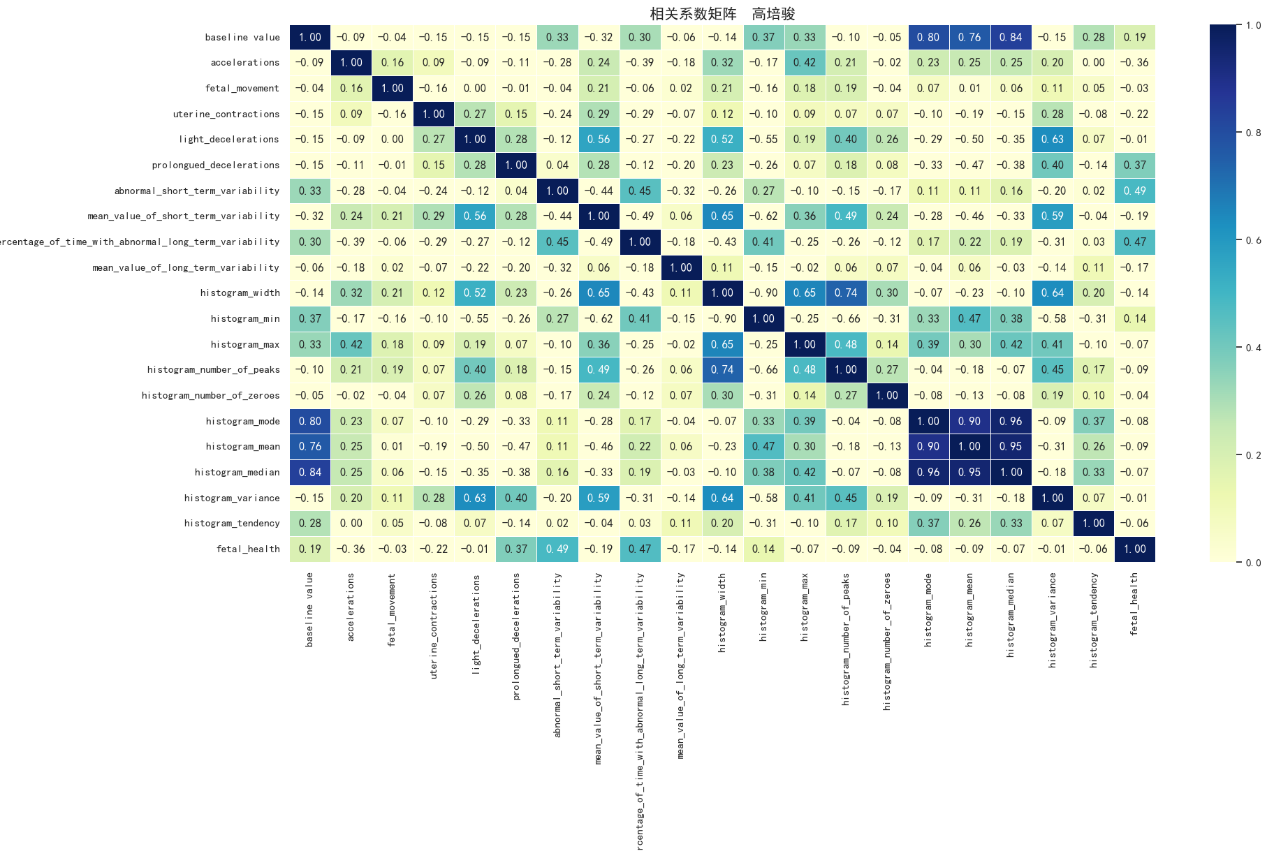
data\_no\_error.iloc[:,:-1] = ((data\_no\_error.iloc[:,:-1] - data\_no\_error.iloc[:,:-1].min()) / (data\_no\_error.iloc[:,:-1].max() - data\_no\_error.iloc[:,:-1].min()))  
data\_no\_error.to\_csv('./dataset/fetal\_health\_after.csv', index=False)

处理后数据可视化：

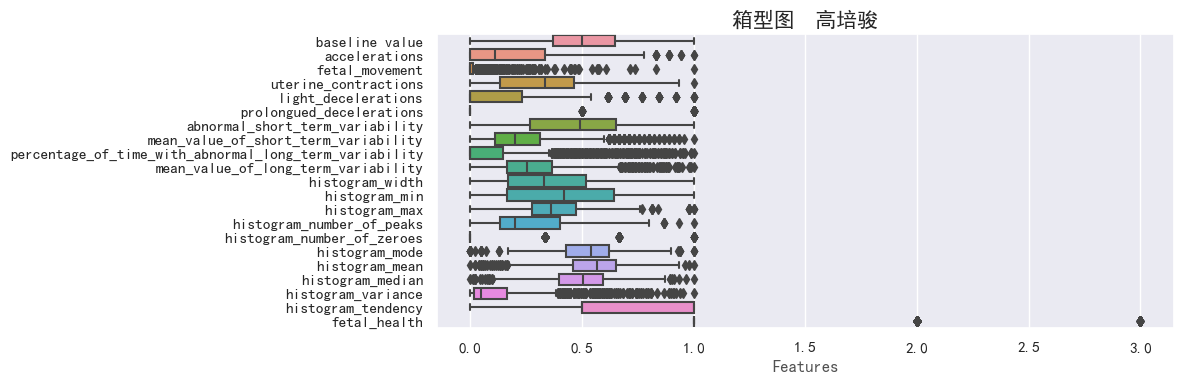
1. 处理后特征直方图



1. 处理后相关系数矩阵



1. 处理后箱型图



1. 随机森林模型

构建随机森林模型我们使用到的方法是```RandomForestRegressor()```，其中随机森林中决策树的个数```n\_estimators```我们设定为100。

# 构建随机森林三分类的模型：  
model = RandomForestRegressor(n\_estimators=100, random\_state=random.randint(0, 100))  
model.fit(X\_train, y\_train)

构建好模型后，我们对测试集进行预测

# 通过训练的模型进行预测  
y\_pred\_before = model.predict(X\_test)  
print(y\_pred\_before)

预测得到的结果和我们想要的数据标签[0, 1, 2]是不一样的，因此，我们采用就近原则的方式对数据进行修正

# 根据就近原则更新数据集  
y\_pred = []  
for data in y\_pred\_before:  
 if abs(data-0) <= abs(data-1) and abs(data-0) < abs(data-2):  
 y\_pred.append(0)  
 elif abs(data-1) < abs(data-0) and abs(data-1) <= abs(data-2):  
 y\_pred.append(1)  
 elif abs(data-2) <= abs(data-0) and abs(data-2) < abs(data-1):  
 y\_pred.append(2)  
print('修正得到的数据如下:')  
print(y\_pred)

在得到预测值之后，我们就可以对模型的性能进行评估，主要涉及到的指标有precision、recall、f1-score、support，并通过混淆矩阵的方式实现对预测结果的可视化。

1. AdaBoost模型

构建随机森林模型我们使用到的方法是AdaBoostClassifier(estimater)，其中estimater我们采用的是决策树分类器DecisionTreeClassifier，深度为5

# AdaBoost模型的搭建  
model = AdaBoostClassifier(DecisionTreeClassifier(max\_depth=5), algorithm="SAMME") # 初始化  
model.fit(X\_train,y\_train)

在搭建好模型之后，我们对测试集进行预测

# 对结果进行预测  
y\_pred = model.predict(X\_test)  
print(y\_pred)

同随机森林算法一样，我们使用到的指标是precision、recall、f1-score、support，并通过混淆矩阵的方式实现对预测结果的可视化。

1. XGBoost模型

构建XGBoost模型，我们使用的是XGBClassifier()方法。

# 创建XGBoost模型  
model = XGBClassifier()  
model.fit(X\_train,y\_train)

构建完模型后，对测试集进行预测

y\_pred = model.predict(X\_test)  
print(y\_pred)

同样的使用precision、recall、f1-score、support，并通过混淆矩阵的方式实现对预测结果的可视化。

1. 高斯朴素贝叶斯模型

构建高斯朴素贝叶斯模型我们使用GaussianNB()方法。

# 高斯朴素贝叶斯模型的搭建  
gnb = GaussianNB()  
gnb.fit(X\_train,y\_train)

使用模型对测试集进行预测

# 预测  
y\_pred\_gnb = gnb.predict(X\_test)  
print(y\_pred\_gnb)

同样的使用precision、recall、f1-score、support，并通过混淆矩阵的方式实现对预测结果的可视化。

1. K最近邻模型

构建K最近邻模型我们使用KNeighborsClassifier()方法，其中n\_neighbors我们设置为5。

# 使用k最近邻算法进行分类  
KNN = KNeighborsClassifier(n\_neighbors=5)  
KNN.fit(X\_train,y\_train)

使用模型对测试集进行预测

y\_pred = KNN.predict(X\_test)  
print(y\_pred)

同样的使用precision、recall、f1-score、support，并通过混淆矩阵的方式实现对预测结果的可视化。

1. 多层感知机模型

构建多层感知机模型，首先先将y标签处理成独热编码的形式

# 使用多层感知机算法进行分类  
y\_train\_ones = to\_categorical(y\_train, num\_classes=3)  
y\_test\_ones = to\_categorical(y\_test, num\_classes=3)

紧接着，使用Sequential()和add()方法顺序搭建神经网络模型

# 搭建模型  
model=Sequential()  
model.add(Dense(16, input\_shape=(20,)))  
model.add(layers.Dropout(0.15))  
model.add(Dense(8))  
model.add(layers.Dropout(0.15))  
model.add(Dense(3, activation='softmax'))

使用compile()函数设置优化器和损失函数，具体的参数为optimizer=Adam(0.0001),loss='categorical\_crossentropy',metrics=['accuracy']

# 设置优化器和损失函数，使用交叉熵作为损失函数  
model.compile(  
 optimizer=Adam(0.0001),  
 loss='categorical\_crossentropy',  
 metrics=['accuracy']  
)

进行模型的训练，并将训练历史返回给history

history = model.fit(X\_train, y\_train\_ones, batch\_size=5, epochs=100, verbose=1, validation\_split=0.2)

使用训练得到的模型对测试集进行预测

y\_pred\_ones = model.predict(X\_test)  
print(y\_pred\_ones)

得到的预测结果和之前的结果并不相同，因此我们将其处理成和原来相同的格式

y\_pred = []  
for i in y\_pred\_ones:  
 i\_series = pd.Series(i)  
 i\_series[i\_series[i\_series == i\_series.max()].index[0]] = 1  
 i\_series[i\_series[i\_series != i\_series.max()].index] = 0  
 i = list(i\_series)  
 y\_pred.append(i\_series[i\_series == i\_series.max()].index[0])  
print(y\_pred)

同样的使用precision、recall、f1-score、support，并通过混淆矩阵的方式实现对预测结果的可视化。同时我们还绘制了训练历史的损失loss和准确度accuracy的曲线。

1. 支持向量机模型

首先我们先对数据进行分组

# 先转化为DataFrame类型再对数据进行分组  
grouped = pd.DataFrame({'fetal\_health':y\_train}).groupby('fetal\_health')

因为要使用支持向量机进行多分类，所以需要构建个支持向量机模型，对于我们这个问题就是，因此，我们将数据集分为三部分。

# 获取到0、1、2的行标签  
X\_train\_0\_index = list(grouped.groups[0])  
X\_train\_1\_index = list(grouped.groups[1])  
X\_train\_2\_index = list(grouped.groups[2])

# 获取到0、1、2的对应的x数据：  
X\_train\_0 = x.iloc[X\_train\_0\_index]  
X\_train\_1 = x.iloc[X\_train\_1\_index]  
X\_train\_2 = x.iloc[X\_train\_2\_index]

# 获取到0、1、2的对应的y数据：  
y\_train\_0 = y.iloc[X\_train\_0\_index]  
y\_train\_1 = y.iloc[X\_train\_1\_index]  
y\_train\_2 = y.iloc[X\_train\_2\_index]

得到三部分的数据集后，再将三部分的数据集两两拼接，得到只包含两类标签的三个数据集

# 将三组数据两两配对  
x\_train\_0\_1 = pd.concat([X\_train\_0, X\_train\_1])  
x\_train\_0\_2 = pd.concat([X\_train\_0, X\_train\_2])  
x\_train\_1\_2 = pd.concat([X\_train\_1, X\_train\_2])  
  
# 分类器的标签数据必须要是离散的  
y\_train\_0\_1 = pd.concat([y\_train\_0, y\_train\_1])  
y\_train\_0\_2 = pd.concat([y\_train\_0, y\_train\_2])  
y\_train\_1\_2 = pd.concat([y\_train\_1, y\_train\_2])

构建三个支持向量机模型

# 第一个支持向量机  
svc0 = SVC(C=2,kernel='rbf',gamma=10,decision\_function\_shape='ovo')  
svc0.fit(x\_train\_0\_1, y\_train\_0\_1)  
y\_pred0 = svc0.predict(X\_test)  
print(y\_pred0)

# 第二个支持向量机  
svc1 = SVC(C=2,kernel='rbf',gamma=10,decision\_function\_shape='ovo')  
svc1.fit(x\_train\_0\_2, y\_train\_0\_2)  
y\_pred1 = svc1.predict(X\_test)  
print(y\_pred1)

# 第三个支持向量机  
svc2 = SVC(C=2,kernel='rbf',gamma=10,decision\_function\_shape='ovo')  
svc2.fit(x\_train\_1\_2, y\_train\_1\_2)  
y\_pred2 = svc2.predict(X\_test)  
print(y\_pred2)

使用投票的原理进行计数，确定最终的标签

# 用来计数，进行投票  
pred\_result = []  
for i in range(len(y\_pred0)):  
 num\_0, num\_1, num\_2 = 0, 0, 0  
 if y\_pred0[i] == 0:  
 num\_0 += 1  
 else:  
 num\_1 += 1  
  
 if y\_pred1[i] == 0:  
 num\_0 += 1  
 else:  
 num\_2 += 1  
  
 if y\_pred2[i] == 1:  
 num\_1 += 1  
 else:  
 num\_2 += 1  
  
 if max(num\_0, num\_1, num\_2) == num\_0:  
 pred\_result.append(0)  
 elif max(num\_0, num\_1, num\_2) == num\_1:  
 pred\_result.append(1)  
 else:  
 pred\_result.append(2)  
print(pred\_result)

同样的使用precision、recall、f1-score、support，并通过混淆矩阵的方式实现对预测结果的可视化。

五．实验结果分析、对比和讨论

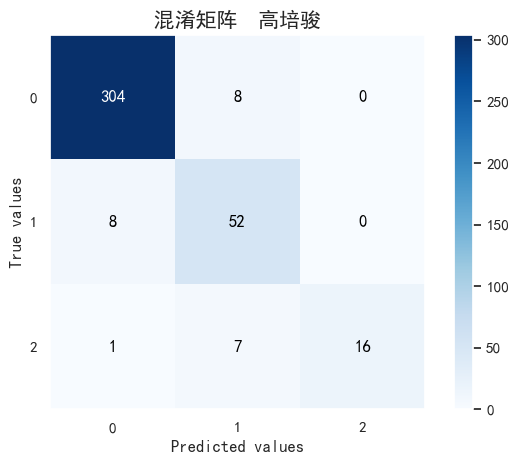
对实验结果进行分析说明，测试一些课程中演示的样例，根据结果说明为什么对或者为什么错等具体分析；对不同模型和参数的对比进行分析；最后对系统提出改进方案等。

1. 随机森林：

指标：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support | accuracy |
| 0 | 0.97 | 0.97 | 0.97 | 312 | 0.94 |
| 1 | 0.79 | 0.87 | 0.83 | 60 |
| 2 | 0.94 | 0.67 | 0.78 | 24 |

混淆矩阵：

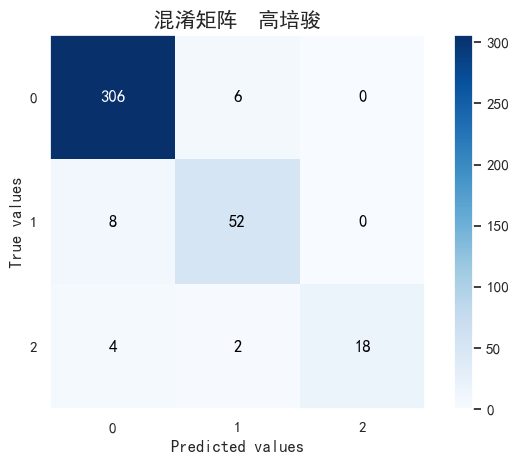


1. AdaBoost：

指标：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support | accuracy |
| 0 | 0.97 | 0.97 | 0.97 | 312 | 0.95 |
| 1 | 0.86 | 0.92 | 0.89 | 60 |
| 2 | 0.95 | 0.75 | 0.84 | 24 |

混淆矩阵：

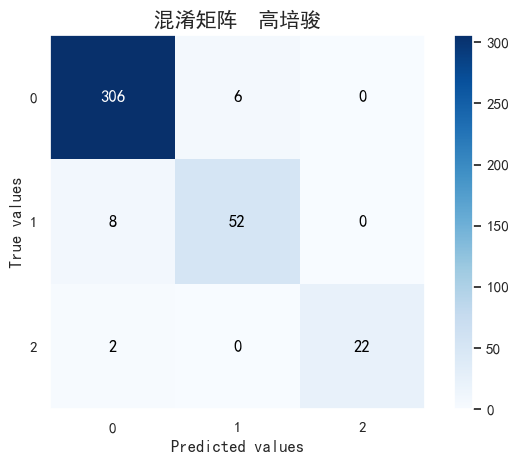


1. XGBoost：

指标：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support | accuracy |
| 0 | 0.97 | 0.98 | 0.97 | 312 | 0.96 |
| 1 | 0.90 | 0.87 | 0.88 | 60 |
| 2 | 1.00 | 0.92 | 0.96 | 24 |

混淆矩阵：

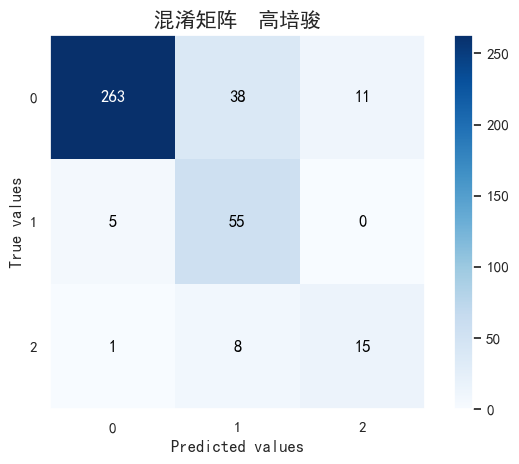


1. 高斯朴素贝叶斯：

指标：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support | accuracy |
| 0 | 0.98 | 0.84 | 0.91 | 312 | 0.84 |
| 1 | 0.54 | 0.92 | 0.68 | 60 |
| 2 | 0.58 | 0.62 | 0.60 | 24 |

混淆矩阵：

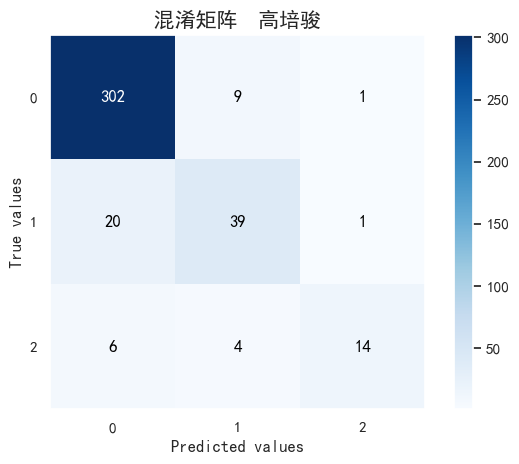


1. K最近邻：

指标：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support | accuracy |
| 0 | 0.92 | 0.97 | 0.94 | 312 | 0.90 |
| 1 | 0.75 | 0.65 | 0.70 | 60 |
| 2 | 0.88 | 0.58 | 0.70 | 24 |

混淆矩阵：

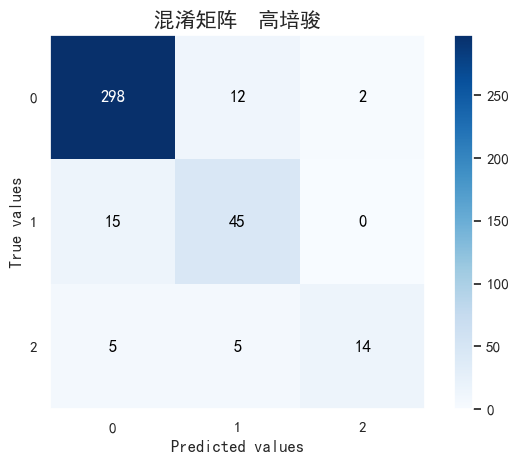


1. 多层感知机：

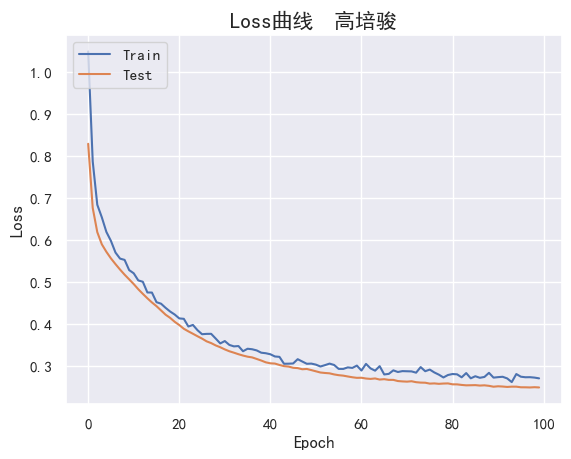
指标：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support | accuracy |
| 0 | 0.94 | 0.95 | 0.95 | 312 | 0.90 |
| 1 | 0.72 | 0.77 | 0.74 | 60 |
| 2 | 0.88 | 0.58 | 0.70 | 24 |

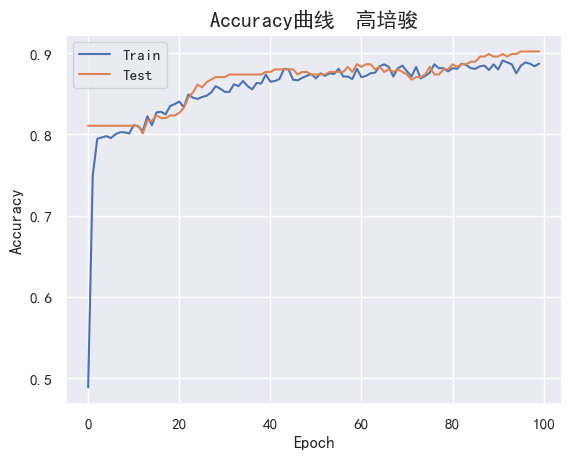
混淆矩阵：



LOSS曲线：



Accuracy曲线：

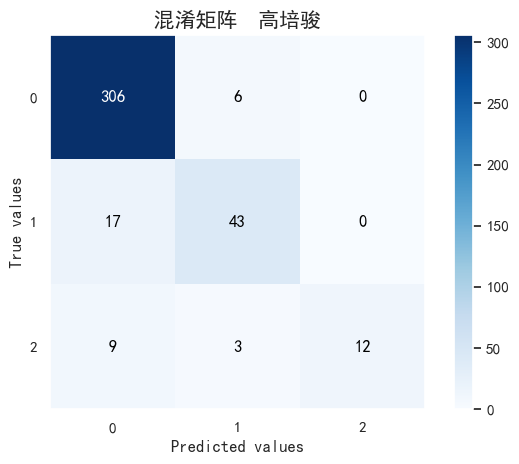


1. 支持向量机：

指标：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support | accuracy |
| 0 | 0.92 | 0.98 | 0.95 | 312 | 0.91 |
| 1 | 0.83 | 0.72 | 0.77 | 60 |
| 2 | 1.00 | 0.50 | 0.67 | 24 |

混淆矩阵：



经过计算得到的指标我们可以看出，准确度从高到低排名分别是：**XGBoost、AdaBoost、随机森林、支持向量机、多层感知机、K最近邻、高斯朴素贝叶斯**。对比来看，XGBoost模型的准确度最高，高斯朴素贝叶斯模型的准确度最低。

六．其他

从上面的结果对比来看，XGBoost模型的表现最优秀，与它的原理离不开，因为它是一个集成学习算法，通过迭代地训练一系列弱分类器（决策树），每一次迭代都试图拟合前一次迭代的残差。最终将这些弱分类器进行组合，得到一个更强大的模型。而高斯朴素贝叶斯模型的准确度相对较低，可能是由于数据集不够多样化造成的，因为朴素贝叶斯模型需要多样化的数据来训练，如果数据集中的样本不够多样化，模型的精确率也会受到影响。另外，需要知道先验概率，因此可能会由于假设的先验模型的原因导致预测效果不佳。

七、对本门课的感想、意见和建议

在本次的大作业中，我接触到了很多的方法，比如随机森林、XGBoost、支持向量机等等，并且将具体的原理也都认真的理解了一遍。比如随机森林，就是一个将很多决策树并行，并将每一棵树的决策结果综合考虑的算法；XGBoost就是使用很多CART回归树，第二棵树的创建需要考虑到第一棵树的结果，第三棵树的创建需要考虑到第二棵树的结果，以此类推，最终构造出一个相对优秀的模型；支持向量机就是寻找到距离支持向量最远的线或平面，从而实现对数据分类的最优化。这些只是举个例子，为了能够相对优秀的完成本次的大作业，仅考虑三种方法是不够的，所以我还学了很多其他方法。

在对机器学习这门课程的学习过程中，我收获到了很多知识。从最开始对于抽象的模型的难以理解，不知道模型真正是用来干什么的，到现在能够看懂模型的原理，并能够运行出来，只有自己才能真正理解其中的困难以及成功实现后的喜悦。我认为，机器学习这门课的应用是十分广泛的，小到自己学校中的大创项目，大到生活中的各种功能，都离不开机器学习这个领域。因此，我觉得学好这门课是至关重要的。并且人工智能是现在比较热门的领域，而机器学习作为人工智能相对基础的一部分，其重要性也是不言而喻的。因此，尽管机器学习这门课程已经告一段落了，但是机器学习这个领域，仍然值得我去好好花一番功夫深究。我也相信自己能在未来的学习路上，不会抛下这门课程，仍然有学习这门课程的热情。

我对这门课程的整体的想法就是老师讲的很通俗易懂，讲很多抽象的知识用一种简单的表达方式讲给我们听，我也确实收获到了很多知识。但是有些美中不足的是，发给我们的ppt和老师讲的ppt内容不太一样，且是全英的，可能对于英语不太好的同学不太友好。但整体而言，我还是很喜欢刘老师讲的机器学习这门课。