REPORT OF EXP5

PB22061259 刘沛 2024/5/25

任务概述

本实验使用与实验三和实验四相同的数据集，现欲使用数据挖掘方法实现各数据集上的分类预测，请各位同学分别在实验三和实验四的两个数据集上，选择若干合适的分类算法进行训练，汇报在测试集上的分类预测结果，并完成实验报告。

实验三数据的预测变量：二值变量 **diagnosis**

实验四数据的预测变量：二值变量 **Class**

实验分析与任务大纲

之前的几次实验都有助教给出来的任务列表，做起来的时候其实目的非常清晰。然而这次的实验并没有这样一份“任务列表”，所以，打开 jupyter 的时候，我的脑子里基本上一片空白，不知道该从何下手。

于是乎，我决定先写这样一份实验报告，事先给出一个实验大致的流程和方向，也就是说，自己给自己写出一份任务列表，按照老师在课堂上讲的数据分析惯常的流程来做这次实验。

以下是任务列表，也是实验分析（当然，实验结果是后来补上的）

数据信息和预处理：

请概述所使用数据集的基本信息，并处理缺失值和异常值。

实验三对应：威斯康辛州乳腺癌数据集（Breast Cancer Wisconsin Dataset），记为 BC1。由之前的实验可以知道这个数据集中是没有异常值的。也是像之前一样，把 diagonosis 的值改为 0 和 1。

```
实验三中数据集合的基本信息如下：
总数据量（已删除缺失值）：560
所有的属性：Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
良性（B）的计数：355,恶性（M）的计数：205
```

实验四对应：乳腺癌数据集（Breast Cancer Dataset），记为 BC2。实验四中助教给出的数据集预备处理方向非常值得学习，要是我自己不看的话其实很难知道哪里是有问题的。就是说，我需要对这个数据集本身比较熟悉，才能知道 tumor-size 这些属性的值有问题，才能进行专门的修改，我觉得这个在真正的实际问题处理当中还是特别困难的。

```
所有的属性：Index(['id', 'Class', 'age', 'menopause', 'tumor-size', 'inv-nodes',
       'node-caps', 'deg-malig', 'breast', 'breast-quad', 'irradiat'],
      dtype='object')
复发的计数：81,不复发的计数：196
```

数据集划分：

使用固定比例 7/1/2 划分训练集，验证集、测试集

这个部分只需要设置一个随机种子把原来的数据集合打乱以后，再进行简单的分片即可

分类算法模型：

选择至少两种数据挖掘算法模型（可以调用算法库，也可以自主实现），作为主实验的比较方法，汇报它们的模型/算法信息，并在报告中引用相应的参考资料。

这里我选择使用支持向量机（Support Vector Machine, SVM），决策树模型进行实验分析

实验三威斯康辛州乳腺癌数据集（Breast Cancer Wisconsin Dataset）

这里我在实验三的数据分析当中，考虑到属性基本上都是连续的取值，所以我选择支持向量机（SVM）模型进行对标签 diagnosis 的预测。

支持向量机：

一开始使用最原始的 SVM 时，准确度就高达 0.96，让我十分震惊，不敢相信

```
Accuracy: 0.9642857142857143
              precision    recall  f1-score   support

           0       0.95      0.98      0.97        63
           1       0.98      0.94      0.96        49

    accuracy                           0.96       112
   macro avg       0.97      0.96      0.96       112
weighted avg       0.96      0.96      0.96       112
```

后来进行了简单的超参数调节

```
# 初始化超参数
C_values = [0.1, 1, 10, 100]
gamma_values = ['scale', 0.1, 1, 10]
best_accuracy_SVM = 0.0
best_params_SVM= {'C': None, 'gamma': None}
```

这是参数调节的过程：

```
Parameters: {'C': 0.1, 'gamma': 'scale'},Accuracy: 0.9375
Parameters: {'C': 0.1, 'gamma': 0.1},Accuracy: 0.8839285714285714
Parameters: {'C': 0.1, 'gamma': 1},Accuracy: 0.5625
Parameters: {'C': 0.1, 'gamma': 10},Accuracy: 0.5625
Parameters: {'C': 1, 'gamma': 'scale'},Accuracy: 0.9642857142857143
Parameters: {'C': 1, 'gamma': 0.1},Accuracy: 0.9553571428571429
Parameters: {'C': 1, 'gamma': 1},Accuracy: 0.5625
Parameters: {'C': 1, 'gamma': 10},Accuracy: 0.5625
Parameters: {'C': 10, 'gamma': 'scale'},Accuracy: 0.9553571428571429
Parameters: {'C': 10, 'gamma': 0.1},Accuracy: 0.9464285714285714
Parameters: {'C': 10, 'gamma': 1},Accuracy: 0.5714285714285714
Parameters: {'C': 10, 'gamma': 10},Accuracy: 0.5625
Parameters: {'C': 100, 'gamma': 'scale'},Accuracy: 0.9375
Parameters: {'C': 100, 'gamma': 0.1},Accuracy: 0.9464285714285714
Parameters: {'C': 100, 'gamma': 1},Accuracy: 0.5714285714285714
Parameters: {'C': 100, 'gamma': 10},Accuracy: 0.5625
Best parameters: {'C': 1, 'gamma': 'scale'}
Best accuracy: 0.9642857142857143
```

在测试集上达到的最高准确度依旧是 0.96 附近。
但出乎意料的是，在验证集合上的准确度是 1.0！

```
验证集上的Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        35
           1       1.00      1.00      1.00        21

    accuracy                           1.00        56
   macro avg       1.00      1.00      1.00        56
weighted avg       1.00      1.00      1.00        56
```

也许，这个数据点之间确实存在着特别清晰的边界吧！

决策树：

而后面调用决策树算法的时候，结果也相对令人满意：

```
决策树在实验三测试集上的Accuracy: 0.9017857142857143
              precision    recall  f1-score   support

           0       0.93      0.89      0.91        63
           1       0.87      0.92      0.89        49

    accuracy                           0.90       112
   macro avg       0.90      0.90      0.90       112
weighted avg       0.90      0.90      0.90       112
```

后面针对决策树的最大深度和拆分内部节点所需的最小样本数这两个参数进行简单调节以后，发现在测试集上的结果好像也没有什么优化：

```
Best Accuracy: 0.9017857142857143
Best Parameters: {'max_depth': 2, 'min_samples_split': 2}
```

以下是参数调节的过程

```
Parameters: {'max_depth': 1, 'min_samples_split': 2},Accuracy: 0.8839285714285714
Parameters: {'max_depth': 1, 'min_samples_split': 3},Accuracy: 0.8839285714285714
Parameters: {'max_depth': 1, 'min_samples_split': 4},Accuracy: 0.8839285714285714
Parameters: {'max_depth': 1, 'min_samples_split': 5},Accuracy: 0.8839285714285714
Parameters: {'max_depth': 1, 'min_samples_split': 6},Accuracy: 0.8839285714285714
Parameters: {'max_depth': 1, 'min_samples_split': 7},Accuracy: 0.8839285714285714
Parameters: {'max_depth': 1, 'min_samples_split': 8},Accuracy: 0.8839285714285714
Parameters: {'max_depth': 1, 'min_samples_split': 9},Accuracy: 0.8839285714285714
Parameters: {'max_depth': 1, 'min_samples_split': 10},Accuracy: 0.8839285714285714
Parameters: {'max_depth': 2, 'min_samples_split': 2},Accuracy: 0.9017857142857143
Parameters: {'max_depth': 2, 'min_samples_split': 3},Accuracy: 0.9017857142857143
Parameters: {'max_depth': 2, 'min_samples_split': 4},Accuracy: 0.9017857142857143
Parameters: {'max_depth': 2, 'min_samples_split': 5},Accuracy: 0.9017857142857143
Parameters: {'max_depth': 2, 'min_samples_split': 6},Accuracy: 0.9017857142857143
Parameters: {'max_depth': 2, 'min_samples_split': 7},Accuracy: 0.9017857142857143
```

最终，用得出来的最优决策树对验证集进行预测：

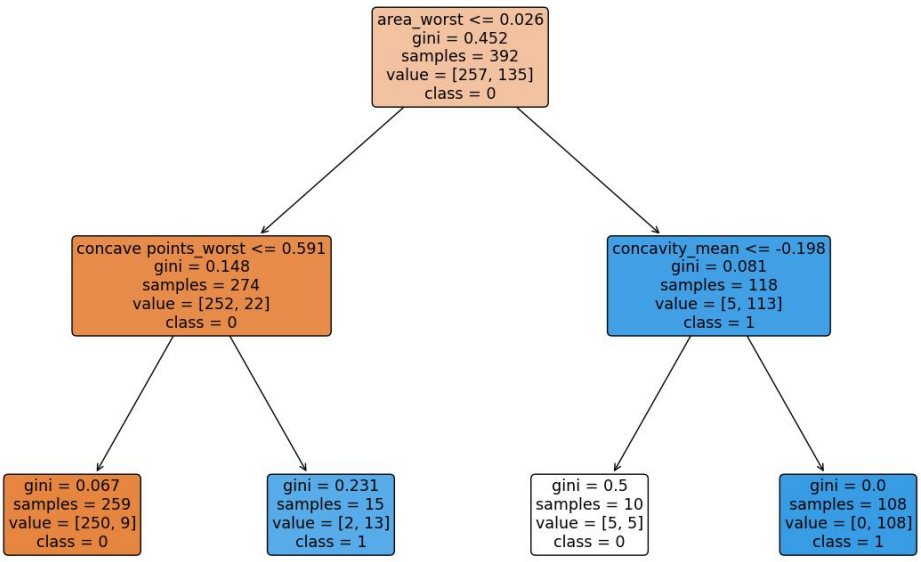
```
决策树在实验三验证集上的Accuracy: 0.9107142857142857
              precision    recall  f1-score   support

           0       0.89      0.97      0.93        35
           1       0.94      0.81      0.87        21

    accuracy                           0.91        56
   macro avg       0.92      0.89      0.90        56
weighted avg       0.91      0.91      0.91        56
```

效果其实也还行，不过比起 SVM 来，还是差了一些。

下面是决策树的可视化：



实验四乳腺癌数据集（Breast Cancer Dataset）

决策树

选择决策树，我这个准确率是针对 test 集合来进行计算得出来，0.6666 是我使用的最原始的决策树算法，没有进行任何超参数的调节。后来对 sklearn 库的深入了解，发现这个库函数中的决策树算法其实里面有很多参数可以提供调节。

第一列是我没有进行参数调节得到的最原始的结果，可以看得到，这个准确度其实并不是很好。

然后我进行了简单的手动参数调节：

```
# 定义候选超参数
max_depth_range = range(1, 11)  # 尝试从1到10的深度
min_samples_split_range = range(2, 11)  # 尝试从2到10的最小样本分割数
```

参数调节过程如下，数量蛮多的，不是很方便全部截图

```
测试集上的Accuracy: 0.6666666666666666
              precision    recall  f1-score   support

           0       0.74      0.79      0.77        39
           1       0.47      0.39      0.42        18

    accuracy                           0.67        57
   macro avg       0.60      0.59      0.59        57
weighted avg       0.65      0.67      0.66        57

Parameters: {'max_depth': 1, 'min_samples_split': 2},Accuracy: 0.7719298245614035
Parameters: {'max_depth': 1, 'min_samples_split': 3},Accuracy: 0.7719298245614035
Parameters: {'max_depth': 1, 'min_samples_split': 4},Accuracy: 0.7719298245614035
Parameters: {'max_depth': 1, 'min_samples_split': 5},Accuracy: 0.7719298245614035
Parameters: {'max_depth': 1, 'min_samples_split': 6},Accuracy: 0.7719298245614035
Parameters: {'max_depth': 1, 'min_samples_split': 7},Accuracy: 0.7719298245614035
```

```
Parameters: {'max_depth': 2, 'min_samples_split': 3},Accuracy: 0.8245614035087719
Parameters: {'max_depth': 2, 'min_samples_split': 4},Accuracy: 0.8245614035087719
Parameters: {'max_depth': 2, 'min_samples_split': 5},Accuracy: 0.8245614035087719
Parameters: {'max_depth': 2, 'min_samples_split': 6},Accuracy: 0.8245614035087719
Parameters: {'max_depth': 2, 'min_samples_split': 7},Accuracy: 0.8245614035087719
Parameters: {'max_depth': 2, 'min_samples_split': 8},Accuracy: 0.8245614035087719
Parameters: {'max_depth': 2, 'min_samples_split': 9},Accuracy: 0.8245614035087719
Parameters: {'max_depth': 2, 'min_samples_split': 10},Accuracy: 0.8245614035087719
Parameters: {'max_depth': 3, 'min_samples_split': 2},Accuracy: 0.7719298245614035
Parameters: {'max_depth': 3, 'min_samples_split': 3},Accuracy: 0.7719298245614035
Parameters: {'max_depth': 3, 'min_samples_split': 4},Accuracy: 0.7719298245614035
Parameters: {'max_depth': 3, 'min_samples_split': 5},Accuracy: 0.7719298245614035
Parameters: {'max_depth': 3, 'min_samples_split': 6},Accuracy: 0.7719298245614035
Parameters: {'max_depth': 3, 'min_samples_split': 7},Accuracy: 0.7719298245614035
```

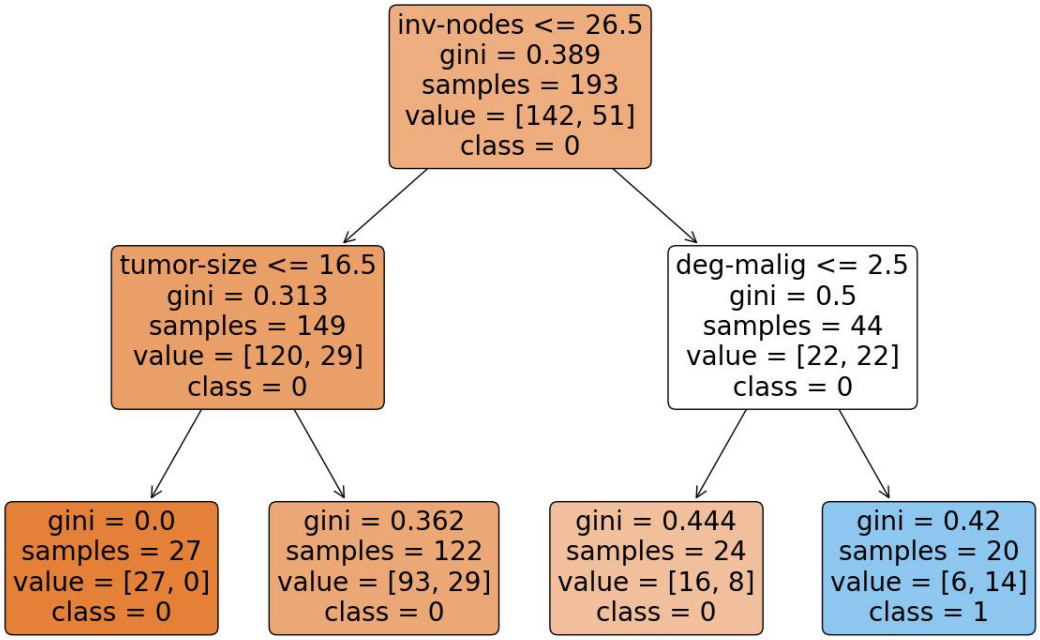
这是我设置的参数调节过程，可以看得出来，深度和分裂度并不是越高越好。

```
Parameters: {'max_depth': 10, 'min_samples_split': 6},Accuracy: 0.7017543859649122
Parameters: {'max_depth': 10, 'min_samples_split': 7},Accuracy: 0.7017543859649122
Parameters: {'max_depth': 10, 'min_samples_split': 8},Accuracy: 0.7017543859649122
Parameters: {'max_depth': 10, 'min_samples_split': 9},Accuracy: 0.7017543859649122
Parameters: {'max_depth': 10, 'min_samples_split': 10},Accuracy: 0.7017543859649122
Best Accuracy: 0.8070175438596491
Best Parameters: {'max_depth': 2, 'min_samples_split': 2}
验证集上的Accuracy: 0.8148148148148148
              precision    recall  f1-score   support

           0       0.86      0.90      0.88        20
           1       0.67      0.57      0.62         7

    accuracy                           0.81        27
   macro avg       0.76      0.74      0.75        27
weighted avg       0.81      0.81      0.81        27
```

最后遍历全部设置，得到的最佳参数设置，以及最佳准确度（这是在测试集上的结果）。下面是这个“最优决策树”的可视化：



在验证集上的结果如下：

```
验证集上的Accuracy: 0.8148148148148148
              precision    recall  f1-score   support

           0       0.86      0.90      0.88        20
           1       0.67      0.57      0.62         7

    accuracy                           0.81        27
   macro avg       0.76      0.74      0.75        27
weighted avg       0.81      0.81      0.81        27
```

出人意料的是，在测试集上进行参数调节后得到的决策树，在验证集上不仅没有出现效果的下降，甚至相比于测试集，准确度还提升了。

实际上，我在具体运行代码的时候，我把数据分片的部分和决策树的训练验证部分分开了。也就是说，其实我每一次建立决策树的时候，用的应该都是同一份数据（在分片的时候我进行了随机化顺序处理）。但是，我每一次运行决策树部分的代码时，得到的准确度还不一样，内部的深层次原因应该是每一次建立的决策树都不一样。**我并不知道为什么会出现这样的情况。**

支持向量机（SVM）

没有进行参数调节时，得到的对测试集的预测结果评估如下：

```
SVM在实验四数据集上的Accuracy: 0.7017543859649122
              precision    recall  f1-score   support

           0       0.70      1.00      0.82        40
           1       0.00      0.00      0.00        17

    accuracy                           0.70        57
   macro avg       0.35      0.50      0.41        57
weighted avg       0.49      0.70      0.58        57
```

出人意料的是，对 Class = 0 情况的预测全部出错。模型效果并不好。

后来针对 SVM 里面的一些超参数进行调节

```
# 初始化超参数
C_values = [0.1,1,2,3,10, 100]
gamma_values = ['scale', 0.1, 1, 10]
best_accuracy_SVM = 0.0
best_params_SVM= {'C': None, 'gamma': None}
```

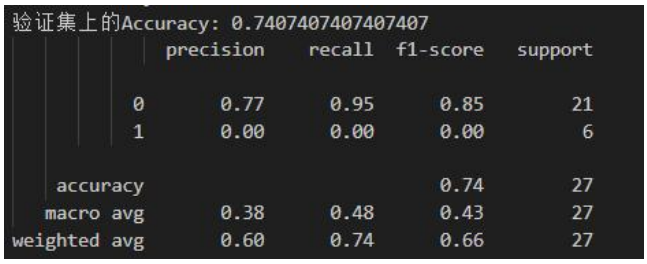
调节过程如下

```
Parameters: {'C': 0.1, 'gamma': 'scale'},Accuracy: 0.7017543859649122
Parameters: {'C': 0.1, 'gamma': 0.1},Accuracy: 0.7017543859649122
Parameters: {'C': 0.1, 'gamma': 1},Accuracy: 0.7017543859649122
Parameters: {'C': 0.1, 'gamma': 10},Accuracy: 0.7017543859649122
Parameters: {'C': 1, 'gamma': 'scale'},Accuracy: 0.7017543859649122
Parameters: {'C': 1, 'gamma': 0.1},Accuracy: 0.7368421052631579
Parameters: {'C': 1, 'gamma': 1},Accuracy: 0.7368421052631579
Parameters: {'C': 1, 'gamma': 10},Accuracy: 0.7368421052631579
Parameters: {'C': 2, 'gamma': 'scale'},Accuracy: 0.7017543859649122
Parameters: {'C': 2, 'gamma': 0.1},Accuracy: 0.7192982456140351
```

在测试集上的最佳结果：

```
Best parameters: {'C': 2, 'gamma': 1}
Best accuracy: 0.7543859649122807
```

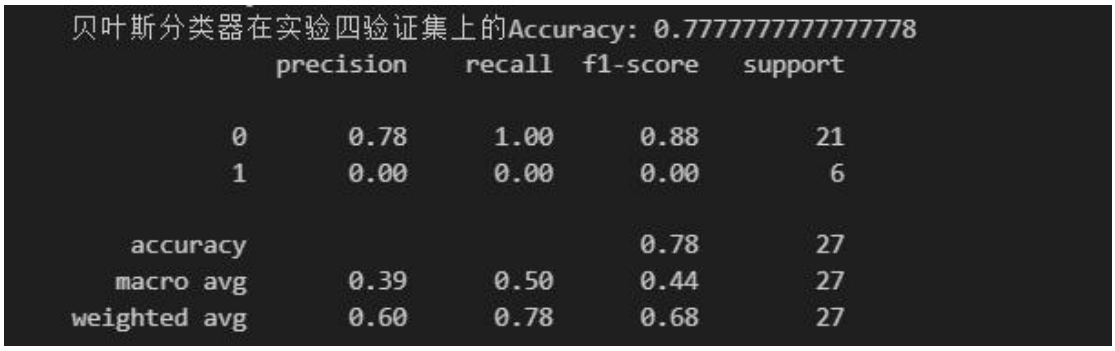
用最佳的 SVM 模型对验证集进行预测：



其实效果不是很好，甚至对 Class = 0 的预测没有一个是对的，或许这个数据集并不适合使用 SVM 进行分类预测。

多项式朴素贝叶斯 (MultinomialNB):

后来我想在这个数据集上实现比较好的预测效果，于是又尝试了一下贝叶斯分类器（反正也只是调包 bushi）



效果和前面的两个方法差不多。

实验总结

1. 不同的预测算法，结果针对同一个数据集合，应该也不一样。可惜这次的时间不太充裕，要不然其实我想要试一试集成的方法，用几个不同的算法同时进行预测，效果应该比单一算法优秀。
2. 实验三中的数据集无论用 SVM 还是决策树算法，效果其实都比较好。而相比之下，实验四数据集上用了三种不同的分类方法，每一个方法分类效果其实都不太好。我觉得，很有可能是因为实验四的数据集中只给出了 9 个有效属性来对 Class 这个标签进行预测，而且每一个属性的取值还实现进行了离散化处理，进一步减小了信息密度。也许，实验四数据集预测准确率上限也就是那样吧！
3. 最终的预测模型的建立和训练其实工作量不大，事实上，确实像老师说的那样，大部分时间都用来做数据的预处理了（在这个实验当中，直接拿前几次实验的预处理结果，所以工作量才比较小的），更何况实验给出的数据集本身就比较漂亮整齐干净了。

这是本课程的最后一个实验了，我感觉在这门课程当中收获非常之多。

首先，作为一个科大学生，深受“数理基础”的影响，在大一大二学期的课程当中，关注的重点都放在了课本上的那些理论，C 语言和数据结构课程学习也是传统的课堂教学模式，那两门课程的实验并没有让我感觉自己学习的知识和做的工作对社会有什么帮助。

但是，在这次的课程当中，老师布置的实验不再关注与算法的细枝末节（我觉得在如今 ai 日益强大的情况下，并没有必要太纠结于和 ai 卷编程能力），而是站在数据分析本身的角度，用来源于实际的数据，去进行一些贴近生活的数据工作。这种实验模式本身，就特别吸引我，哪怕课程开始的时候，我基本上并没有接触过 python，开头的那两次实验做的也是非同寻常的差劲，但是到了后面有关于乳腺癌数据集的分析，我渐渐的掌握了这门课程的核心，也开始运用课程中教授的那些方法去进行实际工作，自我感觉能力发生了非常大的变化！

感谢老师有耐心地对待我们这群一问三不知的呆学生，感谢助教不厌其烦的回复我实验过程中遇到的憨批问题，希望科大以后开出更多像数据分析这样真材实料、贴近实际的课程！