**驭风计划：实验1**

**实验名称： 基于决策树的英雄联盟游戏胜负预测**

**姓名：王灿**

**UID：11031310**

**联系方式：13145318073**

**报告正文**

1. **数据集**
   1. **数据集简介**
      1. 样本数量：数据集League of Legends Diamond Ranked Games (10 min) 来自Kaggle，包含了9879场钻一到大师段位的单双排对局，对局双方几乎是同一水平。每条数据是前10分钟的对局情况，每支队伍有19个特征，红蓝双方共38个特征，涉及了金钱，视野，塔防等多个方面，具体特征和说明可见参考1。
      2. 特性：根据官网介绍和绘制了箱图（图1）进行数据分析，可以了解到其具有以下4条特性。

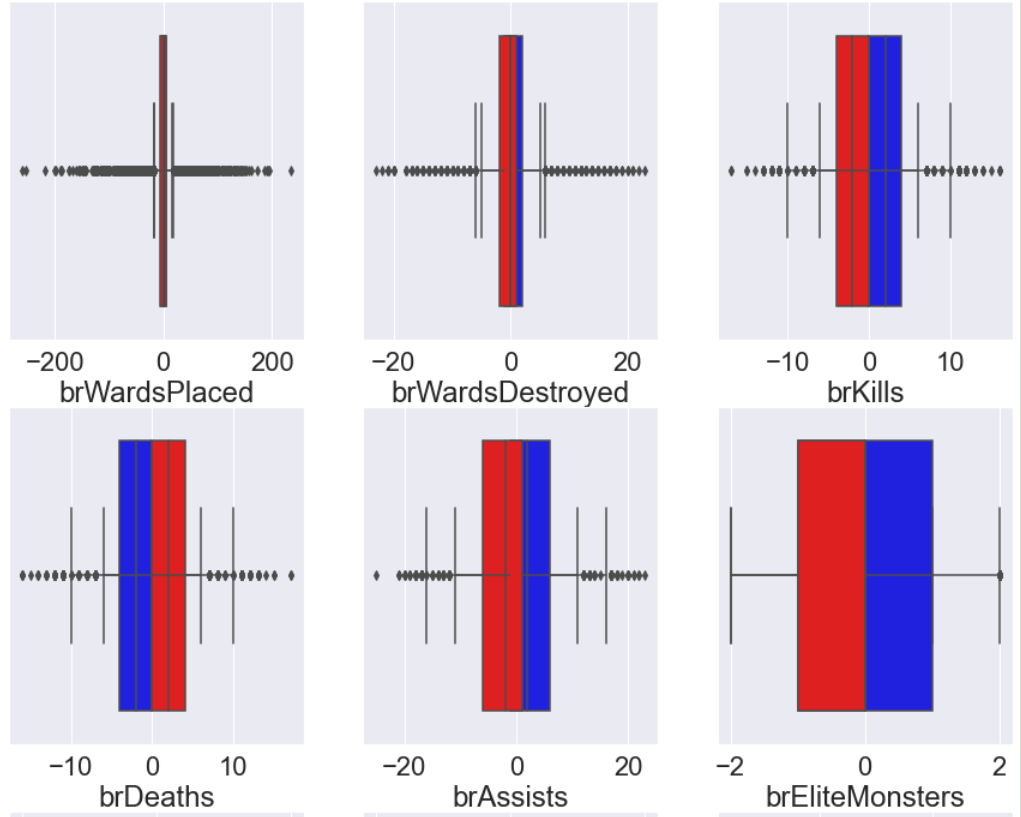


图1 不同特征分布箱图（部分）

* + - 1. 存在冗余特征，如每分钟击杀数和击杀总数，可以通过与时间相乘得到等式，所以在开始要进行**特征的筛选和删除**。
      2. 部分数据是连续的，而决策树算法一般是基于离散特征的，所以预处理部分将对特征进行离散化，即**将一定范围内的值映射成一个值**。
      3. 数据是对称的，可以知道每局游戏红蓝双方的表现是无关的，也就说明可以通过做差来获得新的特征，并且因为对称，存在部分数据是均匀分布于正负样本之间的，所以**对特征的奇数划分**将更为合理。
      4. 数据是正确且合理的，即使存在着离群值。通过代码进行检验了非法条件（比如摧毁视野数大于放置视野数的情况），发现并没有非法数据存在。且可以发现离群值分布松散但连续，每个特征中约有1000个为离群值，说明非噪点所为。那么这部分数据必须被保留并考虑，因为未知空间也可能存在这样的数据。同时，**对于这类数据，按照密度划分特征明显比按照区间划分更为合理。**
    1. 训练集与测试集划分：已有训练样本和测试样本比例为4：1。后文将使用5折交叉验证，将训练样本再随机划分为5份，每次取不同的1份作为验证集，用于超参数的调节和模型的改进，**最终得到比例构成为4：1：1.25的数据集**。
  1. **预处理与特征表示**
     1. **无用特征的剔除：**首先去除无用特征，共有两部分。第一部分是可以通过换算得来已有特征的，比如每分钟金币，经验，击杀。另一部分是根据游戏规则可以判断无影响或影响极小的特征，比如对局编号，第一滴血。
     2. **特征转换：**将剩余28个特征进行蓝红做差，得到为以前特征值一半数量14的数据。绘制热力图（图2）查看其相关性。

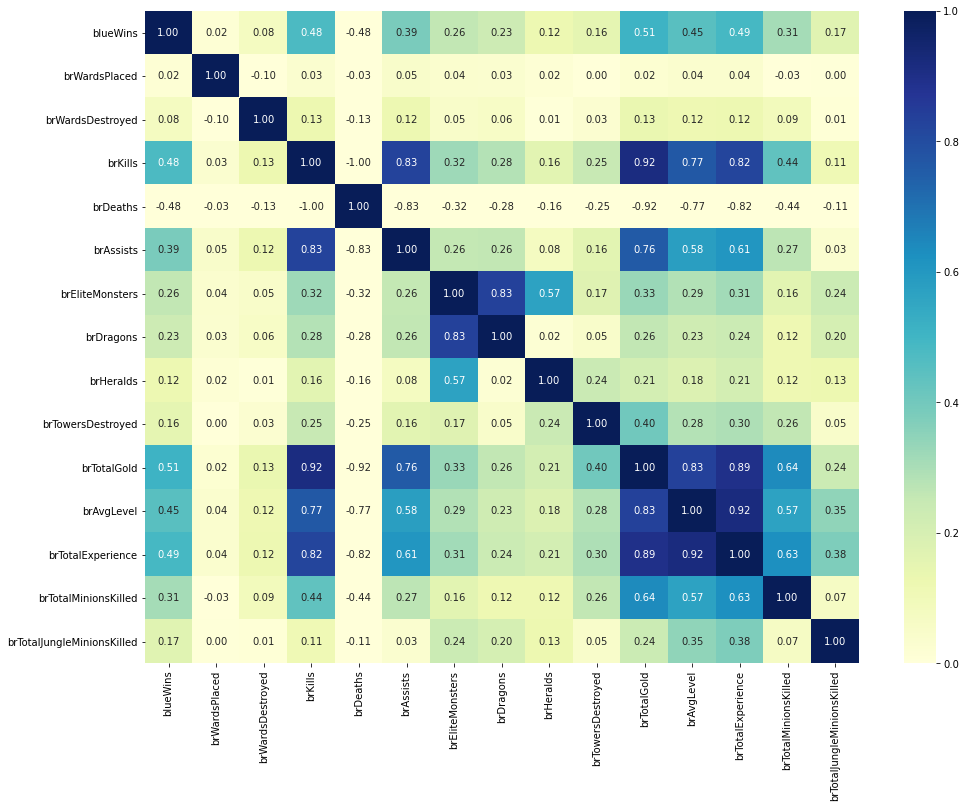


图2 特征之间的相关性

值得一提的是，最开始凭借线性回归的经验，为了消除相关性使用了**PCA主成分分析法进行降维**，结果发现效果并不尽人意，准确率没有提升。查阅资料（参考2）发现决策树和随机森林的预测能力不受多重共线性影响，有以下2点原因：

* + - 1. 统计分析中，作推断时，如果自变量存在共线性，将无法区分它们对因变量的影响，因此无法对结果进行清楚的解释。但是作预测时，我们并不关心如何解释自变量对因变量的影响。
      2. 决策树用贪婪算法进行变量选择，只有新变量对结果影响比较大时，才会被加入到模型中。在决策树模型中，每一个树的构建都是贪婪的，因此，它们决定分裂时，树只会选择一个完全相关的特征。
    1. **弱相关性特征剔除：**设置阈值将与是否获胜（标签）为弱相关性的特征剔除，阈值设定为0.1，即相关性低于0.1时认为影响很小，共舍弃了 'brWardsDestroyed','brWardsPlaced', 'brTowersDestroyed', 'brTotalMinionsKilled', 'brHeralds' 五个弱相关特征。**最后获得数据为9879×9的特征。**
    2. **数据离散化：**如上文1.1.2中所提，通过将连续的特征映射到不同区间来进行离散化，首先区间划分数目一定为奇数，也就是1，3，5……其次，决策树的前几层倾向于与结果相关性最高的特征，也就是总金币和经验，这样可以尽可能的保证树的当前路径最短。通过遍历1~11之间的奇数，观察每一层树的结构，**取层数和特征相关性的加权平均得分（记树的第i层权重为2^(总层数-当前层数)）最高的一颗树即可。**实验结果为3，也就是每个特征将被离散化为3段，每个结点有3个子结点来构建一颗多叉树，实验代码如下。



1. **实验环境**
   1. **计算机配置**

使用计算机为Win10系统，内存大小16G，64位8核处理器，其它配置如下图3所示。

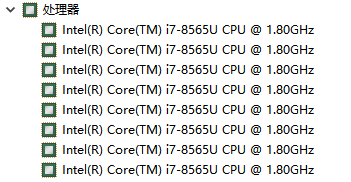


图3 计算机配置

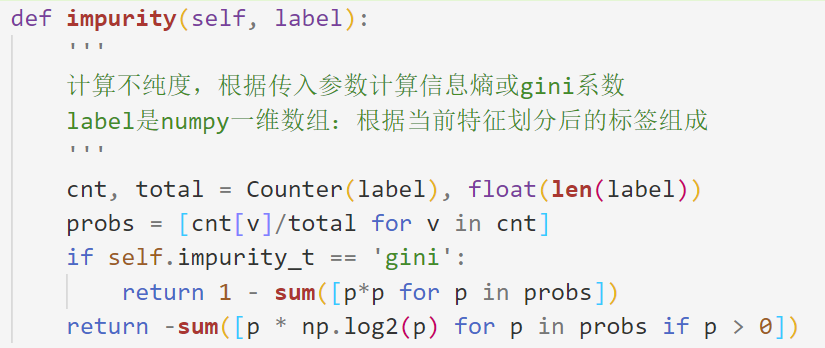
* 1. **编程语言与平台**

采用python编写，版本为3.8.5。编程平台为网页交互式编程环境jupyter notebook。

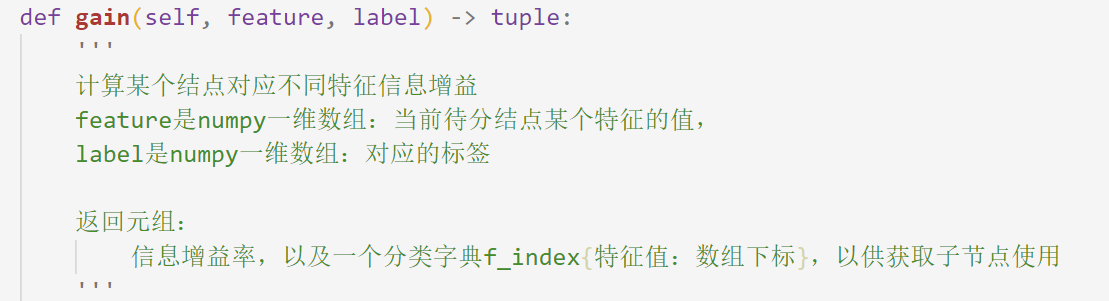
1. **实验过程（此处仅给出部分重点方法的说明）**
   1. **决策树设计**

整个多叉树将以递归的形式创建，采用贪心的方法。因经过特征的删减，解空间并不庞大，可以遍历搜索。**贪心策略为：通过不纯度的做差得到信息增益，优先贪心增益最大的特征。**

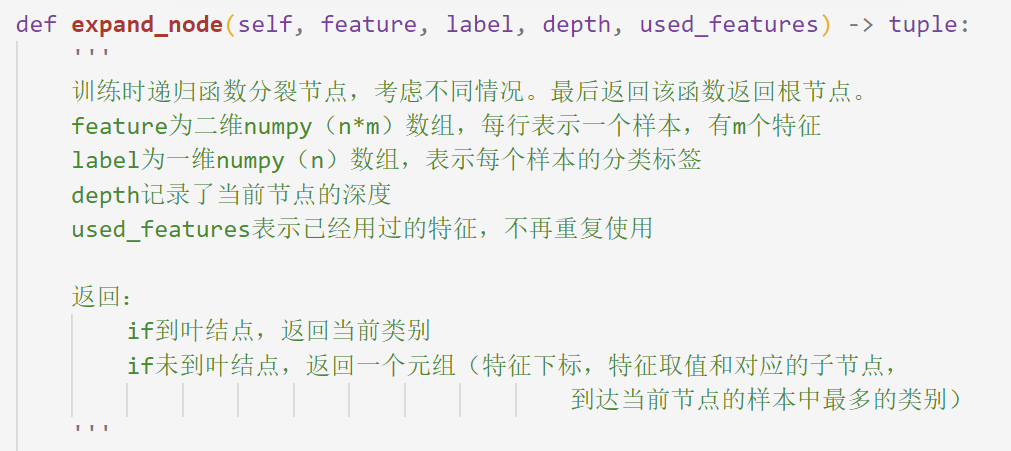
* + 1. 不纯度的计算：根据传入参数的不同，可以采用信息熵和基尼系数两种做法。都是先获取每个取值的比例，再根据公式计算，代码如下。



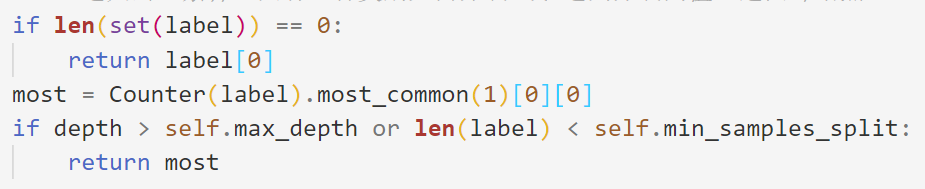
* + 1. 增益的计算：计算当前结点未分裂时的不纯度，和依据每个 “尚未使用” 过的特征来分裂后的不纯度。计算增益时，考虑了信息增益率，即在标签无关时的不纯度，以防止对特征取值多的天然偏置带来的问题。但因为上文离散化时将特征和最大需要划分阈值设为同一个，所以此处并未用到。



* + 1. 递归结点函数：将结点定义为两类，第一类是叶结点，只需要返回当前类别即可。第二类为可以继续分裂的结点，此时其结构为一个元组，元组内容为（特征下标，特征取值和对应的子节点，到达当前节点的样本中最多的类别）。



* + - 1. 递归终止条件：只有一种类别无需分裂或达到分裂阈值，此时返回叶结点，取当前结点最多的类别。



* + - 1. 遍历所有未使用特征，调用gain()找到最佳分裂特征。如果找不到有用的分裂特征，也结束递归。遍历特征的每种取值，递归调用expand\_node进行建树。同时记录当前结点的层数。

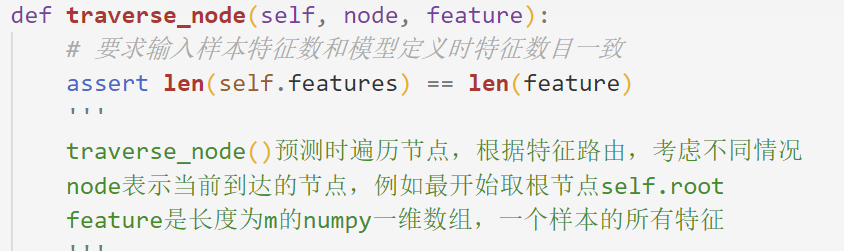


* 1. **数据预测**

数据的预测也将以递归的形式遍历。如果输入是一维numpy（m）数组，则是一个样本，包含m个特征，返回一个类别值即可。如果是二维numpy（n\*m）数组，则表示n个样本，，每个样本包含m个特征，分别调用递归函数，返回一个numpy一维数组。



从根节点开始递归，如果已经到达叶节点，则返回分类结果，否则依据特征取值进入相应子节点，递归调用traverse\_node。注意，如果特征取值在训练集中未出现过，返回训练时到达当前节点的样本中最多的类别。



1. **参数设置和优化**
   1. **待调参数**

决策树分类中共有三类超参数需要选择：

* + 1. 最大深度max\_depth。也是第一个可以确定的参数，随着树的深度变大，模型得分一般随着max\_depth单调递增，之后会趋于稳定。**这个参数和下一个超参数存在着相互制约的关系，随着最大深度变大和叶子结点最小包含样本数变小，模型将趋于复杂。**
    2. 叶子结点最小包含个数min\_samples\_split。⼀个结点必须要包含至少min\_samples\_split 个训练样本，这个结点才允许被分⽀，否则分⽀就不会发⽣。利用暂定的max\_depth参数，绘制曲线，观察得分随着min\_samples\_split（分割内部节点所需的最小样本数）的变化规律，从而确定min\_samples\_split参数的大概范围。
    3. impurity\_t不纯度计算方式。这是最后确定的一个参数，通过遍历确定即可。
  1. **优化上界**

一方面，在实验中调用了sklearn封装好的决策树算法，得到了0.72的准确率，另一方面，在kaggle官网找到了一份使用了多种机器学习方法进行训练的NoteBook（参考3），并进行运行，得到如下图4结果。**可以初步判断，在不对数据进行较大加工的前提下，该算法的上界约为73%。**

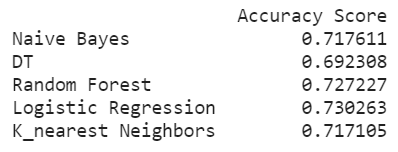


图4 本数据集上其它算法准确率

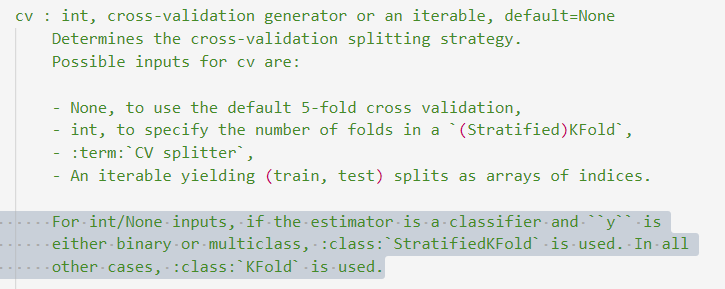
* 1. **优化方法**
     1. K折交叉验证法。即将初始采样分割成K个子样本，一个子样本被保留作为验证模型的数据，其他K-1个样本用来训练。交叉验证重复K次，每个子样本验证一次，结合K次结果得到单一估测。查看源码后，发现StratifiedKFold是用于存在多分类时保证验证集和测试集的类比配比相同，如图6标注部分所示。本数据集并没有这样的要求，所以直接使用cross\_validate即可。

图5 cross\_val\_score分类方式与KFlod方式区别

* + 1. 为获得较好性能且避免过拟合，采取**调参策略如下：首先使用树的深度来限制，不断增加树的深度，选取趋于稳定的最小深度。**然后通过不断减小min\_samples\_split来控制精度。最后通过遍历计算不纯度方法，来确定最终取值。

1. **实验结果**
   1. **各种参数配置下的实验结果**
      1. 如下图6所示，首先固定分裂直至一个结点，来观察拟合程度随着树最大深度的变化。可以发现在深度为3，采用基尼系数的时候准确率最高，为73.10%。

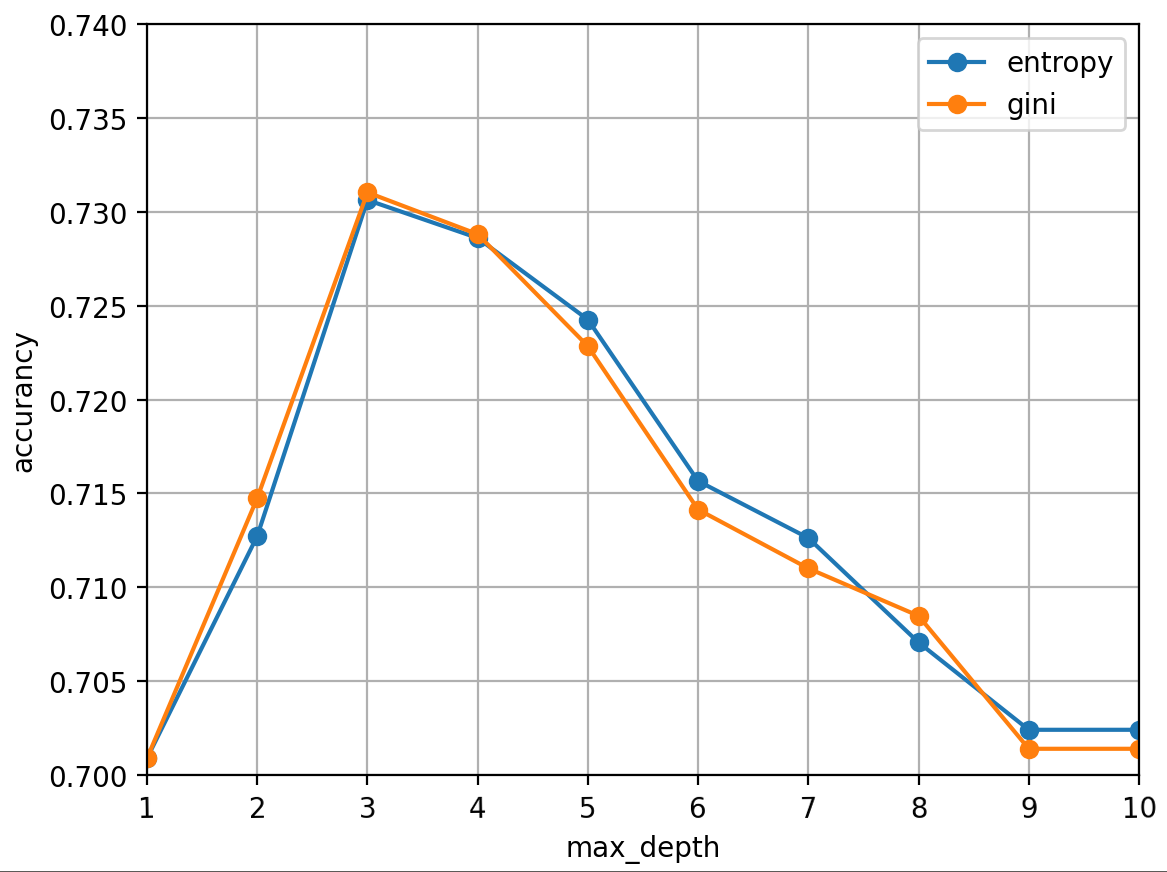


图6 准确率随着最大深度的变化

* + 1. 固定深度后，通过调节最小包含样本数来提高精度K值。可以发现，在样本数为450，采用基尼系数的时候准确率最高，为73.12%。

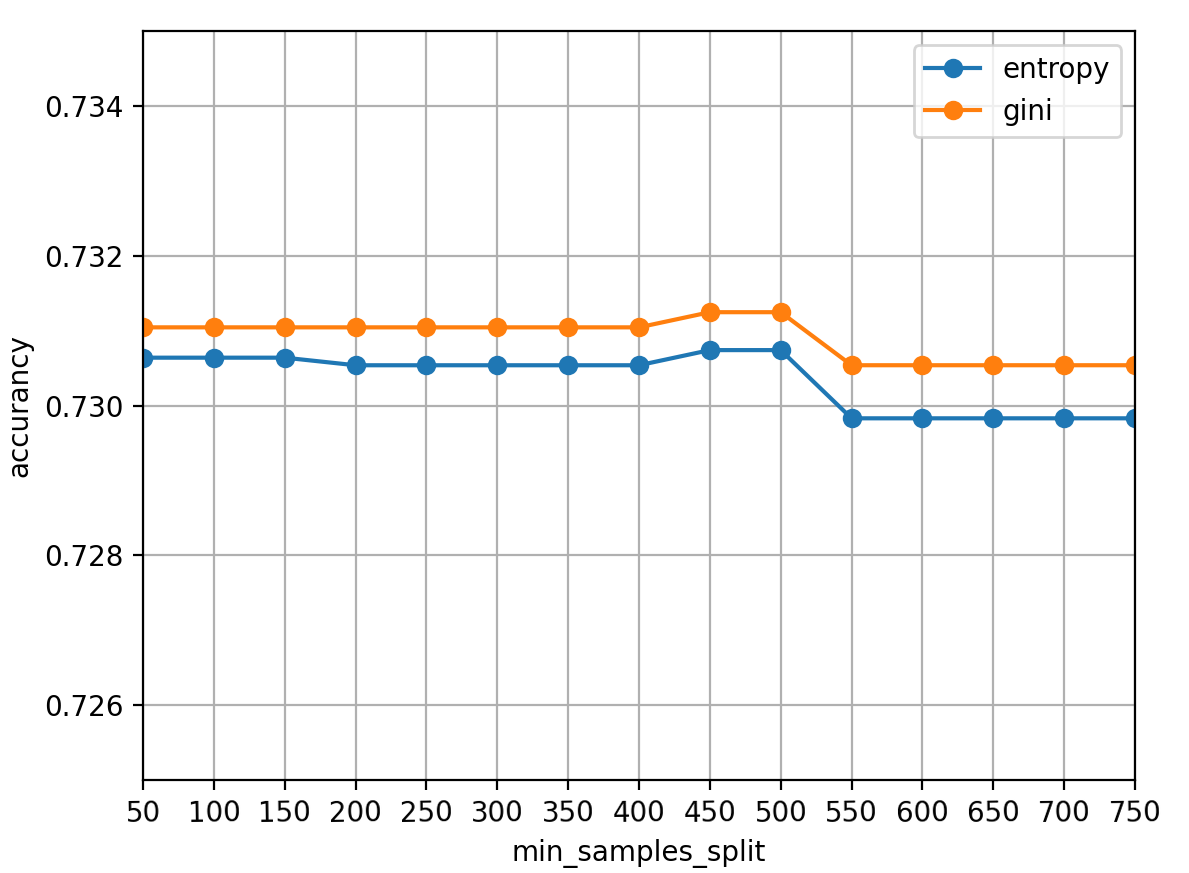
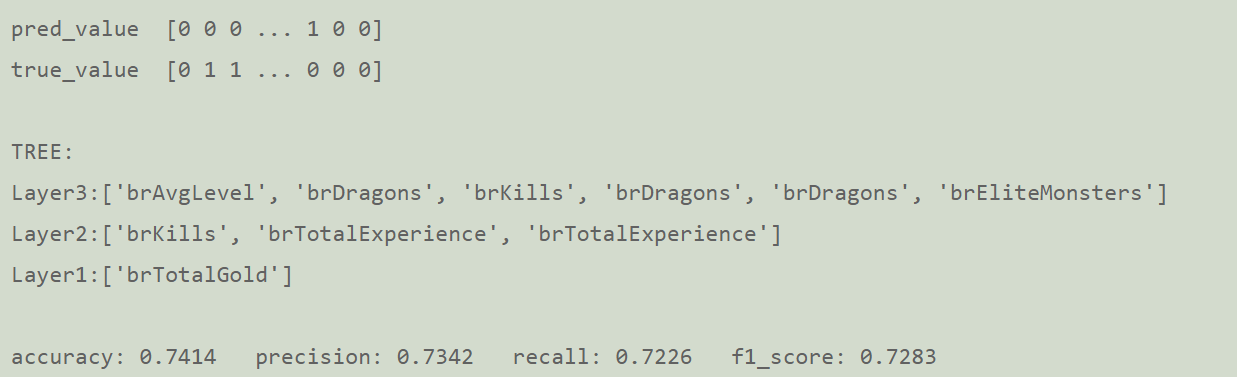


图7 准确率随着最小包含样本数的变化

* 1. **最终准确率**

将该结果（特征个数9，特征划分区间3，不纯度为基尼系数计算，最大深度3，最小包含样本450）应用于测试集，根据测试集是未知领域数据需要模型更强的泛化性，对参数进行微调，得到最终准确率74.14%，树的结构，召回率等其余信息如下图8所示。

图8 测试集上的准确率

* 1. **收获和展望**
     1. 算是第一次比较完整的实践了机器学习的方法。之前总是感觉似懂非懂，查阅了很多资料，尤其是python封装好的决策树给了我很多思路。然后自己一点点改代码报错，一点点的写出来，成就感很足。深刻的感受到了“绝知此事要躬行”的道理，大大加深了我对决策树的理解。比如课上说的根据密度划分的具体方法，还有树可以不仅仅是二叉树等等很多。
     2. 代码能力得到了提高，很多库函数的调用需要查阅文档细节，比如cut和qcut的区别，取的区间开闭问题。调用cross\_validate和GridSearchCV需要重写的方法等。
     3. 因为是代码补全工作，所以顺着做差的思路进行了数据处理。做的途中考虑到另一种思路。可以使用双方特征，根据单独一方的数据进行获胜概率预测，比较哪个概率更大不失为另一种选择。
     4. 关于数据的特征划分：这次实验非常深刻的体会到了数据的重要性，当优化模型参数可提高的程度到了极限，那就应该从数据入手。比如这里我多筛去了无关特征，才使得优化结果能逼近优化上界。同时，推测存在一种更合理的做法，即筛选两类标签的最大值和最小值，这4个点一定是分界点，然后再根据训练结果调整中间的特征个数。后来尝试了这种方法，准确率仍是逼近上界，没有实现突破。
     5. 本实验中直接将叶节点定义为包含最多类别的标签，忽略了概率问题。因为本身竞技预测可能就存在“极限翻盘”或者不可预测现象，所以根据类别概率取可能会更加合理。
     6. 本实验中使用的是预剪枝，如果使用后剪枝可能有更好的表现。

**参考资料**

[1] [League of Legends Diamond Ranked Games (10 min) | Kaggle](https://www.kaggle.com/datasets/bobbyscience/league-of-legends-diamond-ranked-games-10-min/code?datasetId=600276&searchQuery=tree)

[2] [为什么特征相关性非常的重要？ - 云+社区 - 腾讯云 (tencent.com)](https://cloud.tencent.com/developer/article/1537582)

[3] [LoL - how to win | Kaggle](https://www.kaggle.com/code/xiyuewang/lol-how-to-win)

[4] [机器学习超详细实践攻略(9)：决策树算法使用及小白都能看懂的调参指南 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/103136609)

[5] [决策树手动实现 | tian-feng (innovation64.github.io)](https://innovation64.github.io/_posts/2021-06-18-%E5%86%B3%E7%AD%96%E6%A0%91%E6%89%8B%E5%8A%A8%E5%AE%9E%E7%8E%B0/)

[6] [GitHub - RRdmlearning/Decision-Tree: CART Decision Tree](https://github.com/RRdmlearning/Decision-Tree)

[7] [【Python机器学习实战】决策树和集成学习（二）——决策树的实现 - 云+社区 - 腾讯云 (tencent.com)](https://cloud.tencent.com/developer/article/1875200)

[8] [驭风计划：培养人工智能青年人才【第三期】 - 学堂在线 - 学堂在线 (xuetangx.com)](https://www.xuetangx.com/learn/XT0809202106302/ML08091003374/10130922/article/18870662)