* 项目背景
  + 绪论
    - 项目背景

自2016年以来，中国电竞行业进入了井喷期！一方面，移动互联网人口暴增的红利为电竞带来了空前的新生力量，使得电竞逐渐发展成为一种新时代的社交，也渐渐与人们的日常生活融合在了一起；另一方面，政策导向给电竞事业开了前所未有的绿灯，使得电竞的发展不断拥有着更大的舞台——电竞行业已然成为了时代的宠儿。

英雄联盟（League of Legends 简称LOL )是一个多人在线MOBA竞技游戏，也是当下电子竞技比赛中热度最高的游戏之一，其宗旨致力于推动全球电子竞技的发展，形成了自己独有的电子竞技文化！本次项目主要利用收集到的五万多条英雄联盟比赛数据进行分析预测，实现对比赛情况的相关预测和指导。

* + - 选题依据

1. 该选题采用大量真实的比赛数据，包括比赛结果，比赛组成成分，几乎全方面地覆盖了能影响每场游戏进度情况的各种因素，从而可以很好地为比赛结果的预测提供了足够依据。
2. 对于LOL这种MOBA公平竞技游戏来说，每进行一场游戏，受不同玩家，不同打法，不同英雄选择等方面的影响，其实际的发展情况都是千变万化的，但是在特定的一些发展情况下，合理的策略与打法，恰当的推进方式，都会成为影响游戏结果的重要因素，因此，能很好地根据每场比赛的实际进度，发展情况来对比赛进行相应的预测，打法指导等，将会是对游戏意识和策略培养的一个有效方法。
3. 目前，电竞事业在中国发展迅速。LOL又是发展浪潮中的重要角色之一，选题注重新颖性和时代先进性，但又不失实用性。
   * 数据集简介
     + 数据集来源

该项目中所用到的数据集来自kaggle官网，提供数据作者：Mitchell J.其内容为每场LOL比赛各项详细数据，共51490条数据。

* + - 数据集说明

1. 数据集涉及到的游戏内容简介：

a.在英雄联盟中，玩家扮演一名召唤师角色，控制着一名有独特能力的英雄，与其他玩家控制的不同英雄组成一支队伍。两只队伍在固定的地图中进行战斗。每一支队伍的目标是摧毁对方的水晶枢纽。游戏最常使用的地图是5V5召唤师峡谷，本次项目也是依据该地图进行。

b.在地图中，两边玩家不断地向对面地域进攻，摧毁路径上的敌方防御塔，最终目标为摧毁敌方基地的水晶枢纽。整个地图可以抽象成为一个正方形（如图1所示），在地图的对角线是双方各自的基地。双方基地每隔一段时间会产生一定数量的小兵，沿上、中、下三条路线向敌人基地前进。玩家操纵自己的英雄向敌方基地进攻，摧毁敌方建筑，包括防御塔（红色和蓝色圆点），敌方水晶（base），最终赢得比赛。

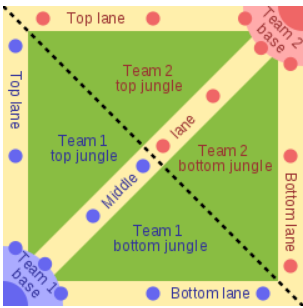


图1

c.现对地图内影响比赛进程的组成因素作介绍说明：

* 防御塔（Tower）：双方队伍的道路上都具有的强大的防御结构，用于阻碍敌方队伍前进至己方基地。每队有11座防御塔。
* 召唤水晶：每条道路上有一座召唤水晶， 当一条路上的三座防御塔被摧毁以后，该条路上的召唤水晶就会受到攻击。召唤水晶被摧毁后，摧毁方将会派出较之前能力更加强大的小兵——超级兵进行战斗推进，且召唤水晶将在被摧毁后的3min内复活，复活后停止召唤超级兵。
* 小龙，大龙，峡谷先锋：位于地图特定区域的野怪，双方队伍皆可以针争夺击杀，击杀后，会给队伍带来不同类别不同效果的增益，对增强战斗能力和兵线推进能力有非常大的影响。
* 水晶枢纽：位于敌方基地，每队拥有一座水晶枢纽，当某条路的防御塔和召唤水晶被摧毁后，水晶枢纽将会受到攻击，水晶枢纽被摧毁后则比赛结束。

1. 数据集各属性对应描述（见表1.1）

表1.1

|  |  |
| --- | --- |
| winner | 胜方 |
| firstBlood | 拿到第一滴血的一方 |
| firstTower | 拿到第一座塔的一方 |
| firstInhibitor | 拿下第一座召唤水晶的一方 |
| firstBaron | 拿到第一条大龙的一方 |
| firstDragon | 拿到第一条小龙的一方 |
| firstRiftHerald | 拿到第一只峡谷先锋的一方 |
| x\_towerKills | 某一方的总推塔数（x= t1 or t2） |
| x\_inhibitorKills | 某一方摧毁召唤水晶的数量（x= t1 or t2） |
| x\_baronKills | 某一方击杀大龙的数量（x= t1 or t2） |
| x\_dragonKills | 某一方击杀小龙的数量（x= t1 or t2） |
| t1\_riftHeraldKills | 某一方击杀峡谷先锋的数量（x= t1 or t2） |
| gameid | 队伍游戏id |
| creation | 比赛进行的时间 |
| Seasoned | 赛季时间 |
| gameDuration | 游戏持续时间 |
| x\_champid | 某一方比赛表现最佳选手(x=t1 or t2) |
| x\_champ#\_sum\* | 某一方第#位选手所使用的第\*个召唤师技能（x=t1 or t2） |
| x\_ban\* | 某个队伍禁用的第\*位英雄（x=t1 or t2） |
|  |  |

* + 项目目标

本次项目使用五万多长LOL比赛情况数据，进行分析挖掘，主要旨在完成以下目标：

1. 分析比赛过程中存在的各种不同的推进形式、不同的比赛进展情况下（如一血，一塔，拿下的大龙数量、小龙数量，峡谷先锋，摧毁防御塔的速度等），对双方比赛结果的影响。
2. 依据每场比赛总体情况综合分析，利用现有数据，选用适当算法进行分析训练，得到相应模型，能够根据做提供的比赛进展数据对比赛结果进行预测。
3. 在2的基础上，对模型进行进一步的完善，使得到的模型能够依据当前比赛的发展情况，针对某一方的具体情况，提供相应预测性分析和打法指导，使其能最大概率地赢得比赛。

* 数据集处理
  + 无关属性剔除

1.根据游戏规则和游戏进行模式，结合大量的游戏实际进行情况来看，游戏者的ID，游戏ID，赛季时间，以及比赛持续时间等并不能从客观情况下影响双方的游戏比赛结果，因此，可以直接忽略属性：

Gameid, creation, Seasoned, gameDuration, x\_champid（属性相关说明见表1.1），得到剩余属性47个。

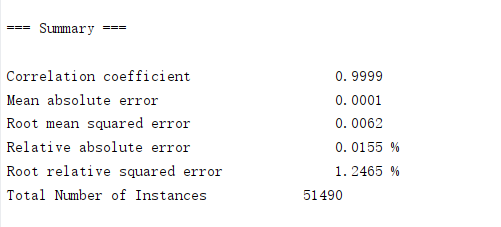
2.借助weka工具，利用属性子集选择方法（参考文献），利用RandomTree算法初步对数据集训练得到粗糙模型，利用模型数据（包括**相关系数，平均绝对误差，均方根误差，相关绝对误差，根相对平方误差**）来评估属性子集，删除不相关或冗余属性，进一步减少数据量。

将双方使用的**召唤师技能（x\_champ#\_sum\*属性）**作为一个子集（每位选手两项，共有20项），比较去除前后，所得到的模型评估数据

子集移除前得到的RandomTree model（共使用47个属性）：

可得到当前模型的评估标准（图2.1）

图2.1



子集移除后得到的RandomTree model（共使用27个属性）：

可得到当前模型的评估标准：（图2.2）

图2.2

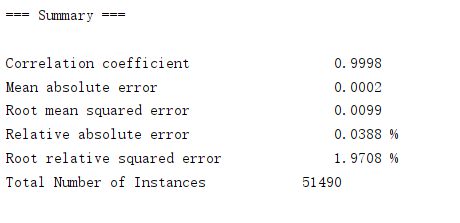
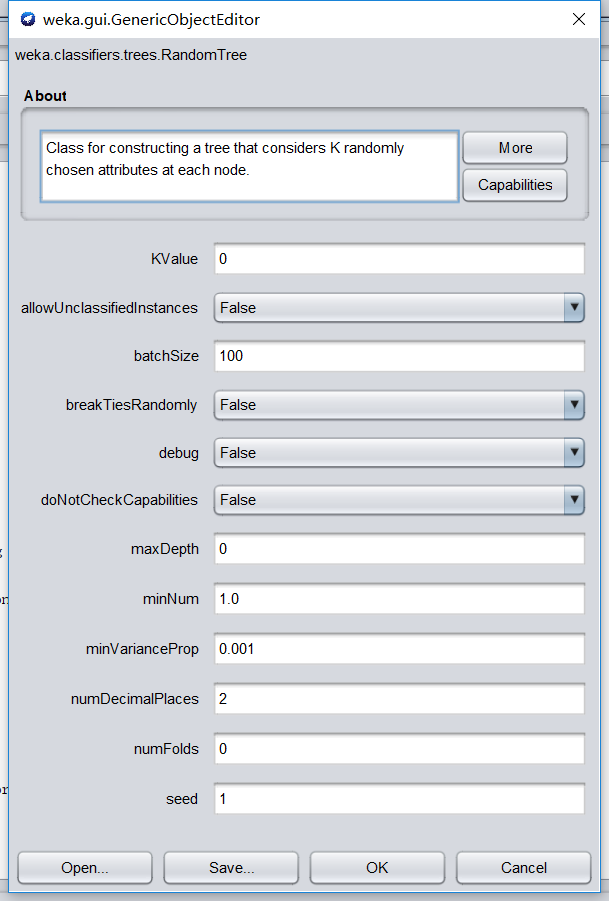


图2.3 RandomTree算法参数设置面板

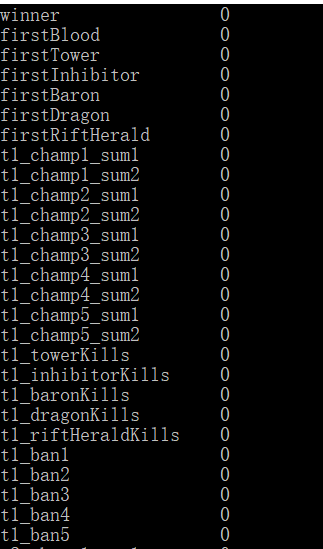
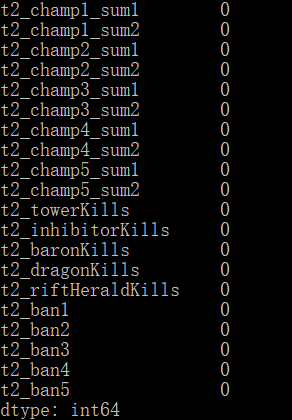


按照前后模型评估数据对比可知，属性子集去除前后相关系数相差0.0001，其余评估系数相差也不超过0.8%，因此，可以可以去除选中的属性子集。

* + 缺失值检测

使用Python检测数据集是否有缺失值（参考文献）：

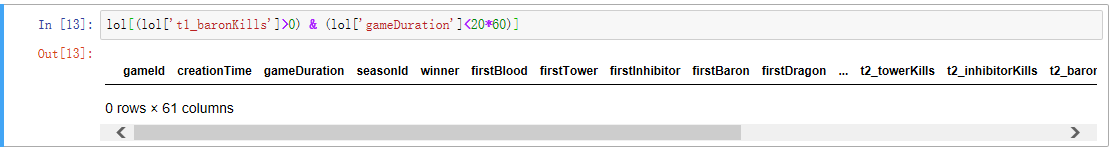
图2.4每列数据缺失值数量

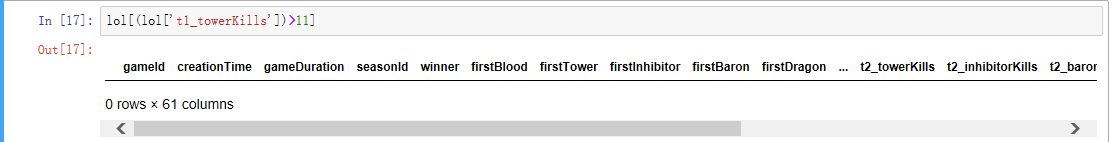
 

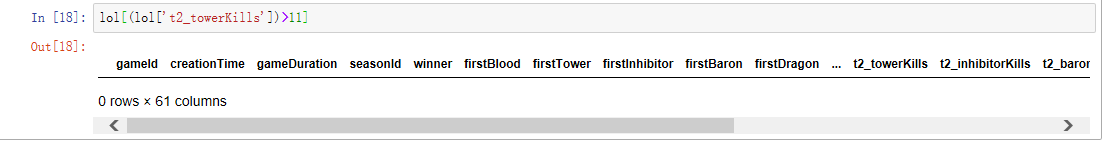
上图列出每列数据属性的数据量，结合输出结果显示，在数据集中不存在缺失值。

* + 噪声数据检测与剔除

1. 根据游戏的实际情况，游戏中存在一些客观规则，不符合规则的则为噪声数据：

* 游戏进行时间(gameDuration)不到20min时，双方的男爵击杀数量(x\_baronKills)都应该为0.
* 双方的它摧毁防御塔数量不应该超过11座（总共每方11座，x\_towerKills>=11）



根据上述检查结果，可以看出数据集中不存在违背客观情况的数据，即不存在噪声数据。

* + 相关属性单项分析与属性值关联分析

**第二节 算法的设计与参数选择**

1. 基于Python sklearn库实现的CART算法

1.CART算法设计思想与实现介绍：

CART（classification and regression tree, 分类与回归树）假设决策树是二叉树，内部结点特征的取值为“是”和“否”，左分支是取值为“是”的分支，右分支是取值为“否”的分支。这样的决策树等价于递归地二分每个特征，将输入空间即特征空间划分为有限个单元，并在这些单元上确定预测的概率分布，也就是在输入给定的条件下输出的条件概率分布。

CART算法由以下两步组成：【2】

决策树生成：基于训练数据集生成决策树，生成的决策树要尽量大；

决策树剪枝：用验证数据集对已生成的树进行剪枝并选择最优子树，这时损失函数最小作为剪枝的标准。

CART决策树的生成就是递归地构建二叉决策树的过程。CART决策树既可以用于分类也可以用于回归。在此次项目中，我们仅讨论用于分类的CART。对分类树而言，CART用Gini系数最小化准则来进行特征选择，生成二叉树。 CART生成算法如下

输入：训练数据集DD，停止计算的条件：

输出：CART决策树。

根据训练数据集，从根结点开始，递归地对每个结点进行以下操作，构建二叉决策树：

1. 设结点的训练数据集为DD，计算现有特征对该数据集的Gini系数。此时，对每一个特征A，对其可能取的每个值a，根据样本点对A=a的测试为“是”或 “否”将D分割成D1和D2两部分，计算A=a时的Gini系数。
2. 在所有可能的特征A以及它们所有可能的切分点a中，选择Gini系数最小的特征及其对应的切分点作为最优特征与最优切分点。依最优特征与最优切分点，从现结点生成两个子结点，将训练数据集依特征分配到两个子结点中去。
3. 对两个子结点递归地调用步骤1~2，直至满足停止条件。
4. 生成CART决策树。
5. 算法停止计算的条件是结点中的样本个数小于预定阈值，或样本集的Gini系数小于预定阈值（样本基本属于同一类），或者没有更多特征。

2. Python sklesrn库DecisionTreeClassifier对象实现CART算法

sklearn中使用sklearn.tree.DecisionTreeClassifier类来实现决策树分类算法，sklearn决策树算法类库内部实现使用了调优过的CART算法。涉及到的典型参数解释如下：【3】

表

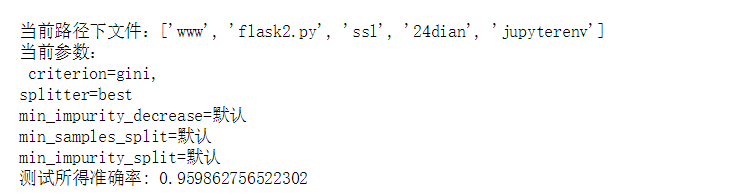
|  |  |  |
| --- | --- | --- |
| **参数名** | **功能** | **描述** |
| criterion | 特征选择标准 | ‘gini’or’entropy’(default=’gini’)，前者是基尼系数，后者是信息熵。两种算法差异不大,对准确率无影响，信息墒运算效率较低。 |
| splitter | 特征划分标准 | ‘best’or ‘random’ (default=”best”) 前者在特征的所有划分点中找出最优的划分点。后者是随机的在部分划分点中找局部最优的划分点。 |
| max\_depth | 决策树最大深度 | int or None, optional (default=None)常用的可以取值10-100之间。常用来解决过拟合。 |
| min\_impurity\_decrease | 节点划分最小不纯度 | float, optional (default=0.) 这个值限制了决策树的增长，如果某节点的不纯度(基尼系数，信息增益，均方差，绝对差)小于这个阈值，则该节点不再生成子节点。 |
| min\_samples\_split | 内部节点再划分所需最小样本数 | nt, float, optional (default=2) 如果是 int,则取传入值本身作为最小样本数；如果是 float,则取ceil(min\_samples\_split \* 样本数量) 的值作为最小样本数，即向上取整。 |
| min\_samples\_leaf | 叶子节点最少样本数 | 如果是 int，则取传入值本身作为最小样本数； 如果是 float，则取ceil(min\_samples\_leaf \* 样本数量) 的值作为最小样本数，即向上取整。 这个值限制了叶子节点最少的样本数，如果某叶子节点数目小于样本数，则会和兄弟节点一起被剪枝。 |
| max\_leaf\_nodes | 最大叶子节点数 | int or None, optional (default=None) 通过限制最大叶子节点数，可以防止过拟合，默认是”None”，即不限制最大的叶子节点数。如果加了限制，算法会建立在最大叶子节点数内最优的决策树。 |
| min\_impurity\_split | 信息增益的阀值 | 决策树在创建分支时，信息增益必须大于这个阀值，否则不分裂 |
| min\_weight\_fraction\_leaf | 叶子节点最小的样本权重和 | float, optional (default=0.) 这个值限制了叶子节点所有样本权重和的最小值，如果小于这个值，则会和兄弟节点一起被剪枝。默认是0，就是不考虑权重问题。 |
| class\_weight | 类别权重 | dict, list of dicts, “balanced” or None, default=None 指定样本各类别的的权重，主要是为了防止训练集某些类别的样本过多，导致训练的决策树过于偏向这些类别。这里可以自己指定各个样本的权重，或者用“balanced”，如果使用“balanced”，则算法会自己计算权重，样本量少的类别所对应的样本权重会高。如果样本类别分布没有明显的偏倚，则可以不管这个参数，选择默认的”None” . |

* Criterion参数选择‘gini’,即使用CART.
* 不推荐使用min\_impurity\_split参数。 它的默认值将在版本0.23中从1e-7变为0，并且将在0.25中删除。改用min\_impurity\_decrease参数。

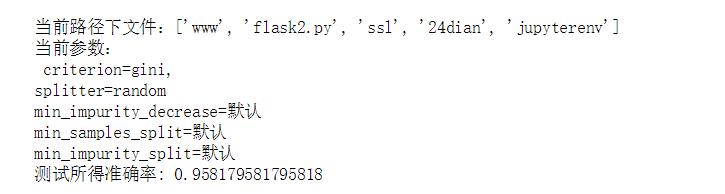
3.2算法相关参数取值测试

* + 1. splitter参数测试：

取’best’时测试结果：



取’random’时测试结果：



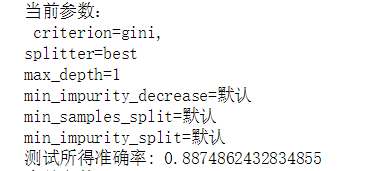
|  |  |
| --- | --- |
| 参数取值 | 测试准确率 |
| best | 0.959862756522302 |
| random | 0.958179581795818 |

由此可看出其他参数保持默认情况下，splitter参数取值对训练结果的影响并不大，两者检验准确率相近。取‘best’时准确率稍高，因此将splitter参数取为‘best’.

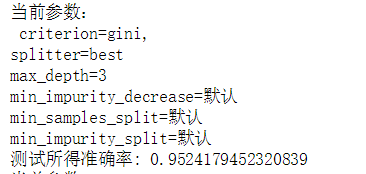
* + 1. max\_depth参数测试

该参数通常用于解决过度拟合问题，在选择参数时，结合数据集实际情况，考虑参数取[1,25]进行测试选取，测试结果如下：

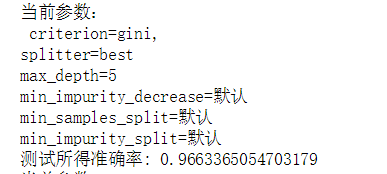
max\_depth取1



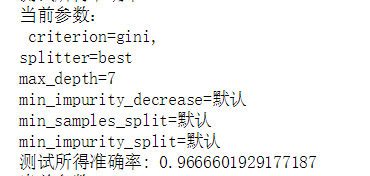
max\_depth取3



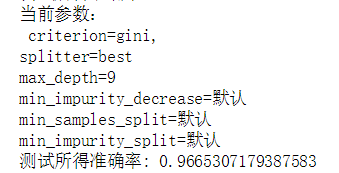
max\_depth取5



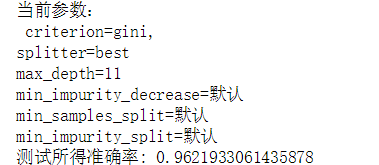
max\_depth取7



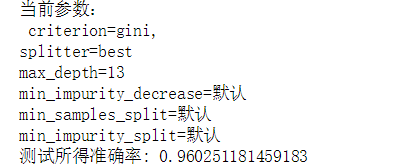
max\_depth取9



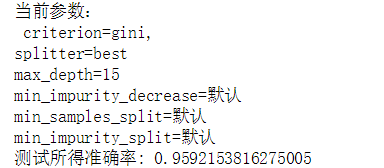
max\_depth取11



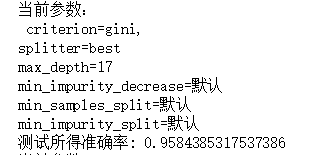
max\_depth取13



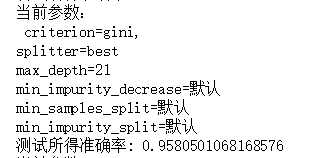
max\_depth取15



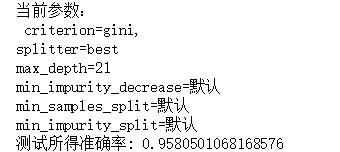
max\_depth取17



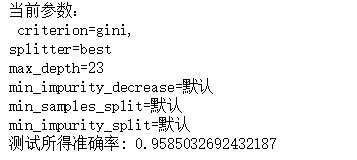
max\_depth取19



max\_depth取21



max\_depth取23



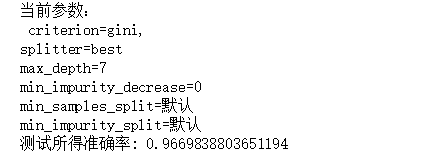
结合以上结果，可以看出，测试结果准确率存在极大值，在max\_depth取值为7时取得。因此选择max\_depth最佳参数取值为7.

* + 1. min\_impurity\_decrease(节点划分最小不纯度)参数测试

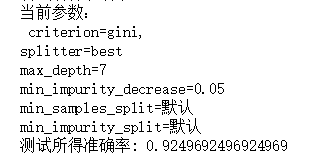
用该参数限制决策树生长，当不纯度小于改参数则终止子结点的生成，该值越大，则会使决策树的子结点越少，但相应的准确度可能会随之降低，具体选择测试结果如下：

在[0,0.5]范围内取值，每次增加0.05，得出模型测试结果

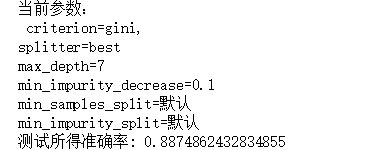
min\_impurity\_decrease取值为0（默认取值）



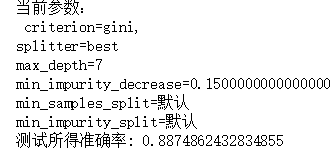
min\_impurity\_decrease取值为0.05



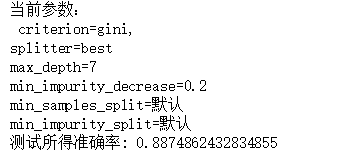
min\_impurity\_decrease取值为0.1



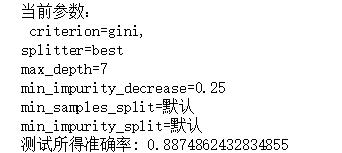
min\_impurity\_decrease取值为0.15



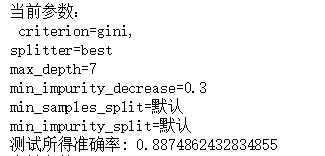
min\_impurity\_decrease取值为0.2



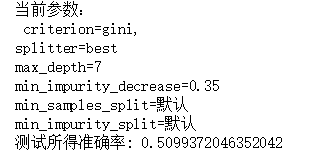
min\_impurity\_decrease取值为0.25



min\_impurity\_decrease取值为0.3



min\_impurity\_decrease取值为0.35



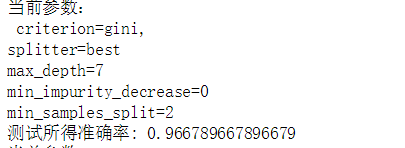
（后续图略）

当取值大于0.3之后准确率明显下降。当参数取值增加到0.05时，虽然相比于0时对决策树有一定简化效果，但模型准确率下降到0.92，效果不理想，因此取min\_impurity\_decrease参数最有取值为0.

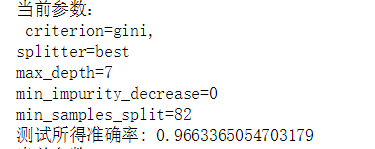
* + 1. min\_samples\_split（内部节点再划分所需最小样本数）参数测试

该参数决定决策树某一内部0节点是否继续划分，能从很大程度上决定决策树的规模。默认值为2，即划分到每一个节点的最小子集（所含元素=1），但是这样会增大决策树规模，造成模型可视化的极大不便，因此需要对此参数进行相应测试，挑选出即能保证准确率又能相应减小决策树规模的参数。选择参数区间[2,1522],每次测试增长间隔依据数据集规模（超过50000条数据）取80，测试结果如下：

min\_samples\_split取2

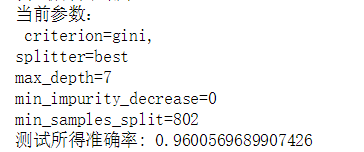


min\_samples\_split取82

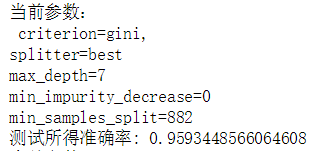


……(中间部分测试结果图略)

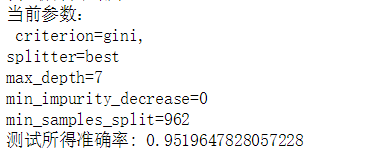
min\_samples\_split取802



min\_samples\_split取882



min\_samples\_split取882



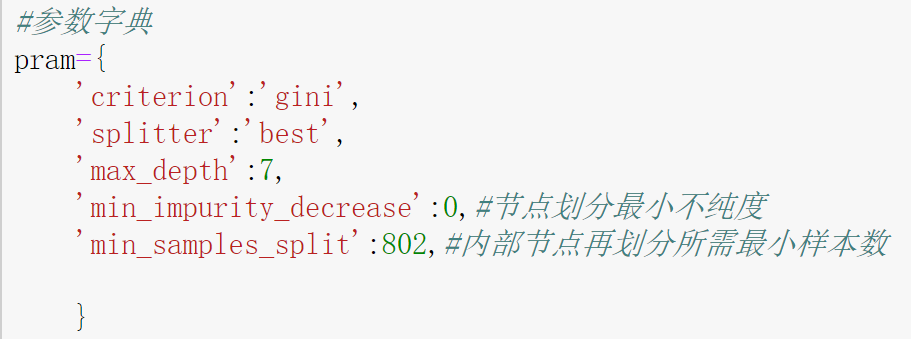
（后续测试图略）

由以上测试结果可见，在[2,802]范围内，测试准确率基本保持在0.966左右，在802以后，准确率有所下降，因此，为在保证测试准确率前提下达到适当减小决策树规模的选参目的，选择min\_samples\_split最优参数为802.

* 结果分析与测试

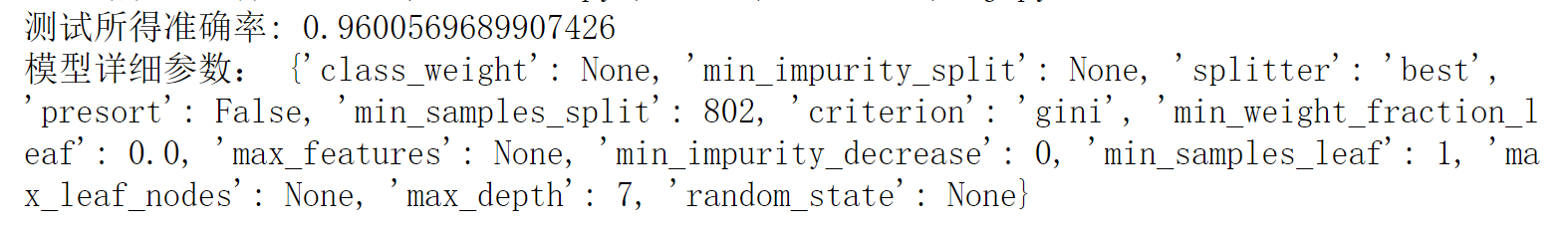
参数：

进过上述选参分析，最终可以确定sklearn.tree.DecisionTreeClassifier决策树算法的参数选用如下图所示：

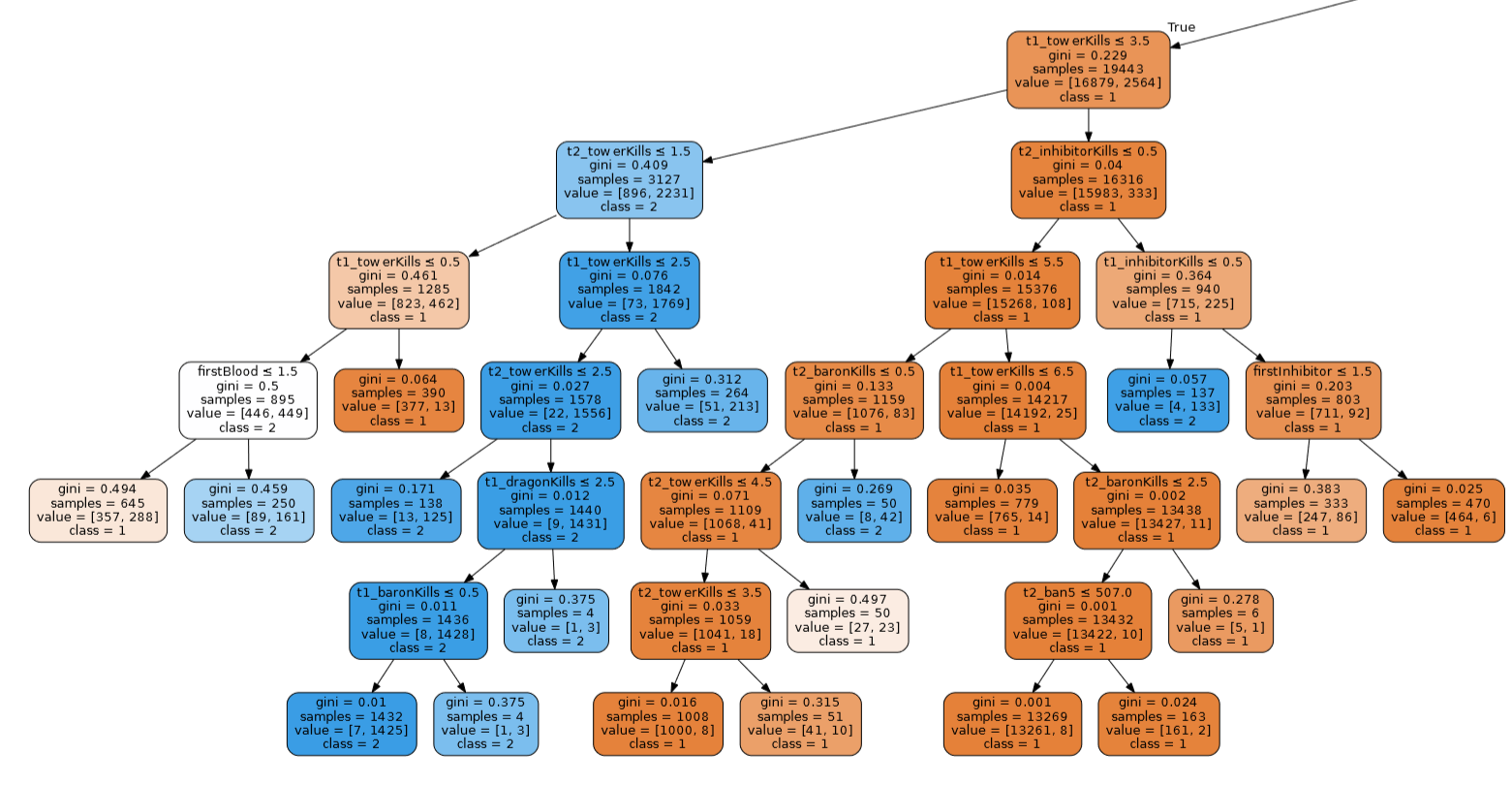


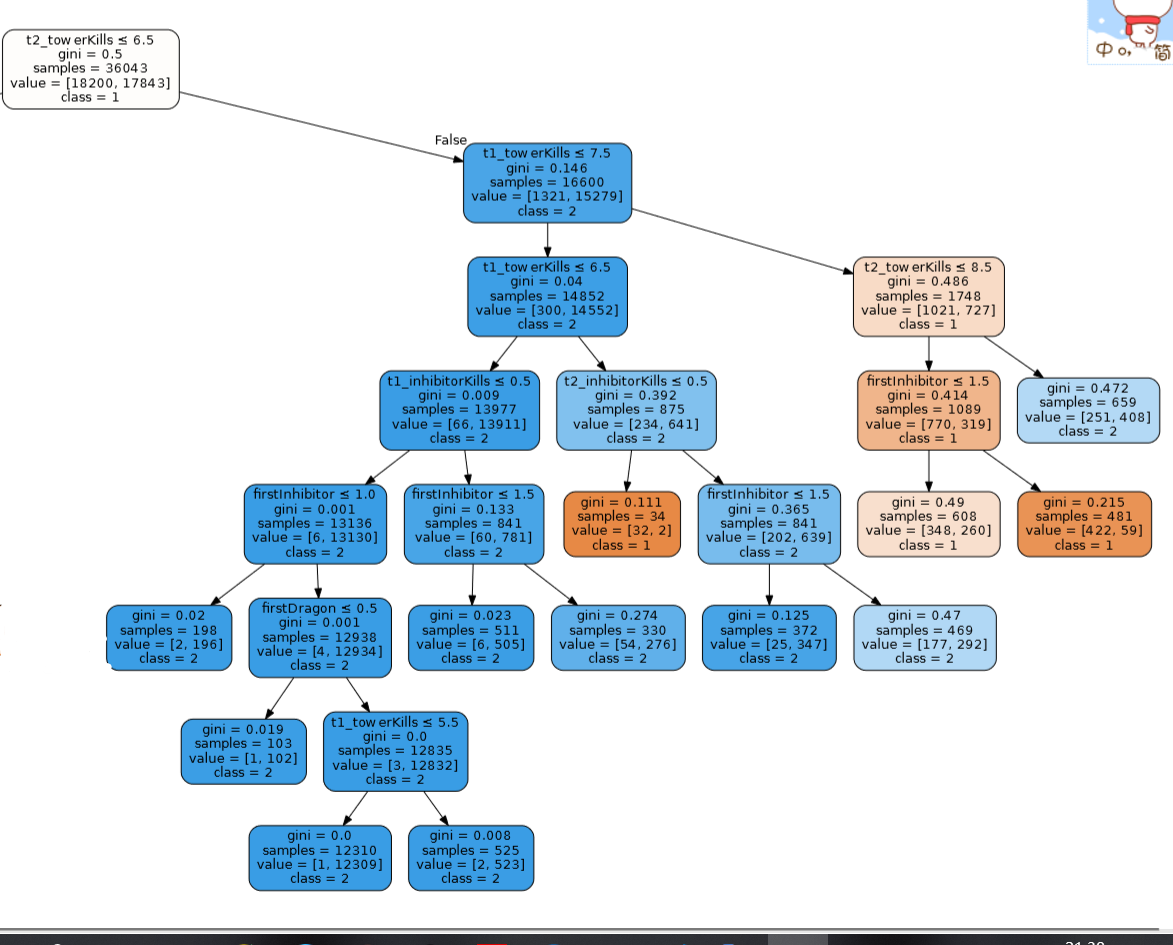
其余参数均使用默认参数。

应用所选参数，采用sklearn中train\_test\_split方法来随机划分训练集和测试集（划分方法详见附件3），运行CART决策树算法，训练结果如图所示：



决策树：

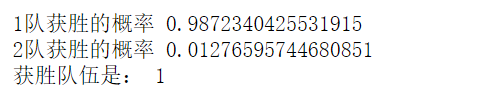




模型测试：【5】

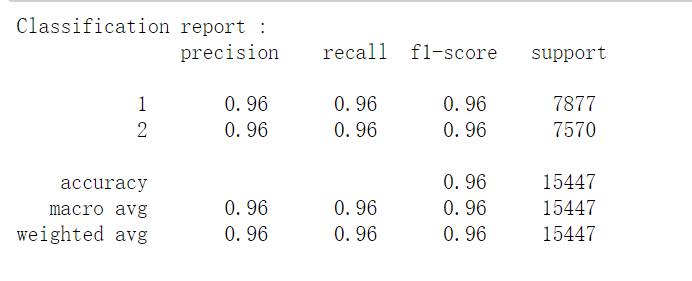
利用predict\_proba()方法，提供相应的测试数据，返回利用模型测试得到的双方获胜的概率，利用predict()方法直接得出获胜概率较高的队伍，如下：

利用测试数据（26个标签）：[1,2,5,4,8,3,5,4,1,2,5,2,1,5,4,1,2,5,4,5,2,3,3,5,4,5] ，测试结果：



模型评估：

使用sklearn库classification\_report函数【4】【5】生成分类报告，可以直观地看出模型测试所得到的相关评估参数，包括精确度，召回率和F1值。（代码详见附件3），得到分类报告如下：



* 参考文献

【1】郭维维. 数据挖掘中属性选择算法的分析与研究[D].北京交通大学,2009.

【2】数据挖掘十大算法之CART详解. (2016年11月23日). 检索来源: 中国软件开发网

链接：https://blog.csdn.net/baimafujinji/article/details/53269040

【3】DecisionTree决策树大全.(2018-04-07).检索来源：刘知行-机器学习

链接：http://ihoge.cn/2018/DecisionTree.html

【4】机器学习笔记－－classification\_report&精确度/召回率/F1值.检索来源：软件开发网

链接：https://blog.csdn.net/akadiao/article/details/78788864

【5】scikit-learn官方文档--API Reference

链接：https://scikit-learn.org/stable/modules/classes.html#module-sklearn.tree

附录1：数据预处理代码

缺失值检测：

import pandas as pd

import missingno as msno

import matplotlib.pyplot as plt

data= pd.read\_csv('games.csv')

print(data.isnull().sum()) #统计每一列中空值的数目

噪声检测：

import pandas as pd

import missingno as msno

import matplotlib.pyplot as plt

import numpy as np

data= pd.read\_csv('games.csv')

#print(data.describe())

plt.scatter(data['winner'], data['t1\_towerKills'])

plt.show();#绘制散点图

附录2：参数选择，算法实现，模型检测代码（详见注释）