# 云南大学软件学院期末课程报告

Final Course Report
School of Software, Yunnan University

## 个人成绩

| 序号 | 学号 | 姓名 | 成绩 |
|---|---|---|---|
| 1 | 20171120116 | 王英臣 | |
| 2 | 20171260024 | 许逸凡 | |
| 3 | 20171130195 | 王　渤 | |

学　　期：2019 学年春季学期

课程名称：数据挖掘与安全

任课教师：张云春

实践题目：英雄联盟对局获胜者分析

小 组 长：王英臣

联系电话：18183053901

电子邮件：405415501@qq.com

# 《数据挖掘与安全》期末课程报告成绩考核表

**年级：2017 专业：信息安全 学号：20171120116 姓名：王英臣**

**本人所做工作：** 项目选题，组织组员工作，随机森林算法测试，撰写文档

| 指标内容 | 分值 | 指标内涵及评估标准 | | | | 得分 |
|---|---|---|---|---|---|---|
| | | A | B | C | D | |
| **选题定位（20 分）** | | | | | | |
| 选题意义 | 5 | 意义重大 | 意义较大 | 意义一般<br>属于简单开发 | 无意义 | |
| 解决的关键技术问题 | 5 | 准确，范围合适<br>重点突出 | 基本准确 | 部分关键 | 未抓住关键 | |
| 技术路线可行程度 | 10 | 合理可行具体且有创新 | 合理可行具体 | 基本合理可行 | 不够合理 | |
| **完成情况（50 分）** | | | | | | |
| 小组成员的工作量 | 15 | 高出平均要求工作量 15%以上 | 高出平均要求工作量 | 达到平均要求工作量 | 低于平均要求工作量 | |
| 项目完成技术水平 | 15 | 难度很大超出一般本科生水平 | 难度较大达到本科毕业论文水平 | 难度一般达到普通课程要求水平 | 难度小很容易实现 | |
| 达到预期目标程度 | 10 | 完全达到 | 基本达到 | 无法预见 | 未能达到 | |
| 团队精神 | 10 | 团队合作精神强 | 合作情况良好 | 合作情况一般 | 合作不好 | |
| **总结情况（30 分）** | | | | | | |
| 报告撰写质量（30 分） | 5 | 报告非常完整 | 报告比较完整 | 完整程度一般 | 报告不完整 | |
| | 5 | 逻辑结构清晰 | 逻辑组织较好 | 逻辑组织一般 | 逻辑不清 | |
| | 5 | 内容非常丰富 | 内容较丰富 | 内容一般 | 内容欠缺 | |
| | 5 | 文字表达非常好 | 文字表达较好 | 文字表达一般 | 文字表达差意思不明了 | |
| | 5 | 图表制作非常专业化 | 图件制作良好 | 图件制作一般 | 图件制作效果差 | |
| | 5 | 整体效果很好 | 整体效果良好 | 整体效果一般 | 整体效果差 | |
| 综合得分（总分：100 分） | | | | | | |
| 评语： | | | | | | |
| 任课老师签名 | | | | | | |

# 《数据挖掘与安全》期末课程报告成绩考核表

**年级：2017 专业：信息安全 学号：20171260024 姓名：许逸凡**

**本人所做工作：** 调整参数完成随机森林算法，编写文档

| 指标内容 | 分值 | 指标内涵及评估标准 | | | | 得分 |
|---|---|---|---|---|---|---|
| | | A | B | C | D | |
| **选题定位（20 分）** | | | | | | |
| 选题意义 | 5 | 意义重大 | 意义较大 | 意义一般<br>属于简单开发 | 无意义 | |
| 解决的关键技术问题 | 5 | 准确，范围合适<br>重点突出 | 基本准确 | 部分关键 | 未抓住关键 | |
| 技术路线可行程度 | 10 | 合理可行具体且有创新 | 合理可行具体 | 基本合理可行 | 不够合理 | |
| **完成情况（50 分）** | | | | | | |
| 小组成员的工作量 | 15 | 高出平均要求工作量 15%以上 | 高出平均要求工作量 | 达到平均要求工作量 | 低于平均要求工作量 | |
| 项目完成技术水平 | 15 | 难度很大超出一般本科生水平 | 难度较大达到本科毕业论文水平 | 难度一般达到普通课程要求水平 | 难度小很容易实现 | |
| 达到预期目标程度 | 10 | 完全达到 | 基本达到 | 无法预见 | 未能达到 | |
| 团队精神 | 10 | 团队合作精神强 | 合作情况良好 | 合作情况一般 | 合作不好 | |
| **总结情况（30 分）** | | | | | | |
| 报告撰写质量（30 分） | 5 | 报告非常完整 | 报告比较完整 | 完整程度一般 | 报告不完整 | |
| | 5 | 逻辑结构清晰 | 逻辑组织较好 | 逻辑组织一般 | 逻辑不清 | |
| | 5 | 内容非常丰富 | 内容较丰富 | 内容一般 | 内容欠缺 | |
| | 5 | 文字表达非常好 | 文字表达较好 | 文字表达一般 | 文字表达差意思不明了 | |
| | 5 | 图表制作非常专业化 | 图件制作良好 | 图件制作一般 | 图件制作效果差 | |
| | 5 | 整体效果很好 | 整体效果良好 | 整体效果一般 | 整体效果差 | |
| 综合得分（总分：100 分） | | | | | | |
| 评语： | | | | | | |
| 任课老师签名 | | | | | | |

# 《数据挖掘与安全》期末课程报告成绩考核表

**年级：2017 专业：信息安全 学号：20171130195 姓名：王渤**

**本人所做工作：** 编写代码实现利用 CART 决策树算法 、参数选择、编写文档

| 指标内容 | 分值 | 指标内涵及评估标准 | | | | 得分 |
|---|---|---|---|---|---|---|
| | | A | B | C | D | |
| **选题定位（20 分）** | | | | | | |
| 选题意义 | 5 | 意义重大 | 意义较大 | 意义一般<br>属于简单开发 | 无意义 | |
| 解决的关键技术问题 | 5 | 准确，范围合适<br>重点突出 | 基本准确 | 部分关键 | 未抓住关键 | |
| 技术路线可行程度 | 10 | 合理可行具体且有创新 | 合理可行具体 | 基本合理可行 | 不够合理 | |
| **完成情况（50 分）** | | | | | | |
| 小组成员的工作量 | 15 | 高出平均要求工作量 15%以上 | 高出平均要求工作量 | 达到平均要求工作量 | 低于平均要求工作量 | |
| 项目完成技术水平 | 15 | 难度很大超出一般本科生水平 | 难度较大达到本科毕业论文水平 | 难度一般达到普通课程要求水平 | 难度小很容易实现 | |
| 达到预期目标程度 | 10 | 完全达到 | 基本达到 | 无法预见 | 未能达到 | |
| 团队精神 | 10 | 团队合作精神强 | 合作情况良好 | 合作情况一般 | 合作不好 | |
| **总结情况（30 分）** | | | | | | |
| 报告撰写质量（30 分） | 5 | 报告非常完整 | 报告比较完整 | 完整程度一般 | 报告不完整 | |
| | 5 | 逻辑结构清晰 | 逻辑组织较好 | 逻辑组织一般 | 逻辑不清 | |
| | 5 | 内容非常丰富 | 内容较丰富 | 内容一般 | 内容欠缺 | |
| | 5 | 文字表达非常好 | 文字表达较好 | 文字表达一般 | 文字表达差意思不明了 | |
| | 5 | 图表制作非常专业化 | 图件制作良好 | 图件制作一般 | 图件制作效果差 | |
| | 5 | 整体效果很好 | 整体效果良好 | 整体效果一般 | 整体效果差 | |
| 综合得分（总分：100 分） | | | | | | |
| 评语： | | | | | | |
| 任课老师签名 | | | | | | |

# 《数据挖掘与安全》期末课程报告成绩考核表

**年级：2017　专业：信息安全　学号：20171120095　姓名：胡震晗**

**本人所做工作：** 参与选题讨论、分析选题和数据集可行性，协助完成二、三部分

| 指标内容 | 分值 | 指标内涵及评估标准 | | | | 得分 |
|---|---|---|---|---|---|---|
| | | A | B | C | D | |
| **选题定位（20 分）** | | | | | | |
| 选题意义 | 5 | 意义重大 | 意义较大 | 意义一般<br>属于简单开发 | 无意义 | |
| 解决的关键技术问题 | 5 | 准确，范围合适<br>重点突出 | 基本准确 | 部分关键 | 未抓住关键 | |
| 技术路线可行程度 | 10 | 合理可行具体且有创新 | 合理可行具体 | 基本合理可行 | 不够合理 | |
| **完成情况（50 分）** | | | | | | |
| 小组成员的工作量 | 15 | 高出平均要求工作量 15%以上 | 高出平均要求工作量 | 达到平均要求工作量 | 低于平均要求工作量 | |
| 项目完成技术水平 | 15 | 难度很大超出一般本科生水平 | 难度较大达到本科毕业论文水平 | 难度一般达到普通课程要求水平 | 难度小很容易实现 | |
| 达到预期目标程度 | 10 | 完全达到 | 基本达到 | 无法预见 | 未能达到 | |
| 团队精神 | 10 | 团队合作精神强 | 合作情况良好 | 合作情况一般 | 合作不好 | |
| **总结情况（30 分）** | | | | | | |
| 报告撰写质量（30 分） | 5 | 报告非常完整 | 报告比较完整 | 完整程度一般 | 报告不完整 | |
| | 5 | 逻辑结构清晰 | 逻辑组织较好 | 逻辑组织一般 | 逻辑不清 | |
| | 5 | 内容非常丰富 | 内容较丰富 | 内容一般 | 内容欠缺 | |
| | 5 | 文字表达非常好 | 文字表达较好 | 文字表达一般 | 文字表达差意思不明了 | |
| | 5 | 图表制作非常专业化 | 图件制作良好 | 图件制作一般 | 图件制作效果差 | |
| | 5 | 整体效果很好 | 整体效果良好 | 整体效果一般 | 整体效果差 | |
| 综合得分（总分：100 分） | | | | | | |
| 评语： | | | | | | |
| 任课老师签名 | | | | | | |

# 目　　录

# 一、绪论

## 1. 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。

## 2. 选题依据

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。

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。

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。

# 二、数据集简介

## 1. 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。

## 2. 数据集说明

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。

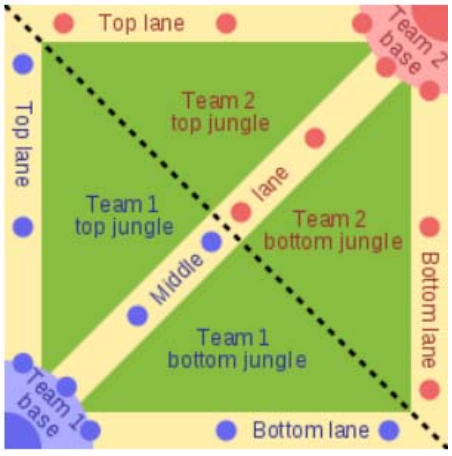



图 1

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。

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_riftHeraldK | 某一方击杀峡谷先锋的数量（x= t1 or t2） |

| | |
|---|---|
| ills | |
| 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） |

## 3. 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。

# 三、数据集处理

## 1. 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

```
=== Summary ===

Correlation coefficient                  0.9999
Mean absolute error                      0.0001
Root mean squared error                  0.0062
Relative absolute error                  0.0155 %
Root relative squared error              1.2465 %
Total Number of Instances            51490
```

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

```
=== Summary ===

Correlation coefficient                  0.9998
Mean absolute error                      0.0002
Root mean squared error                  0.0099
Relative absolute error                  0.0388 %
Root relative squared error              1.9708 %
Total Number of Instances            51490
```

图 2.3 RandomTree 算法参数设置面板

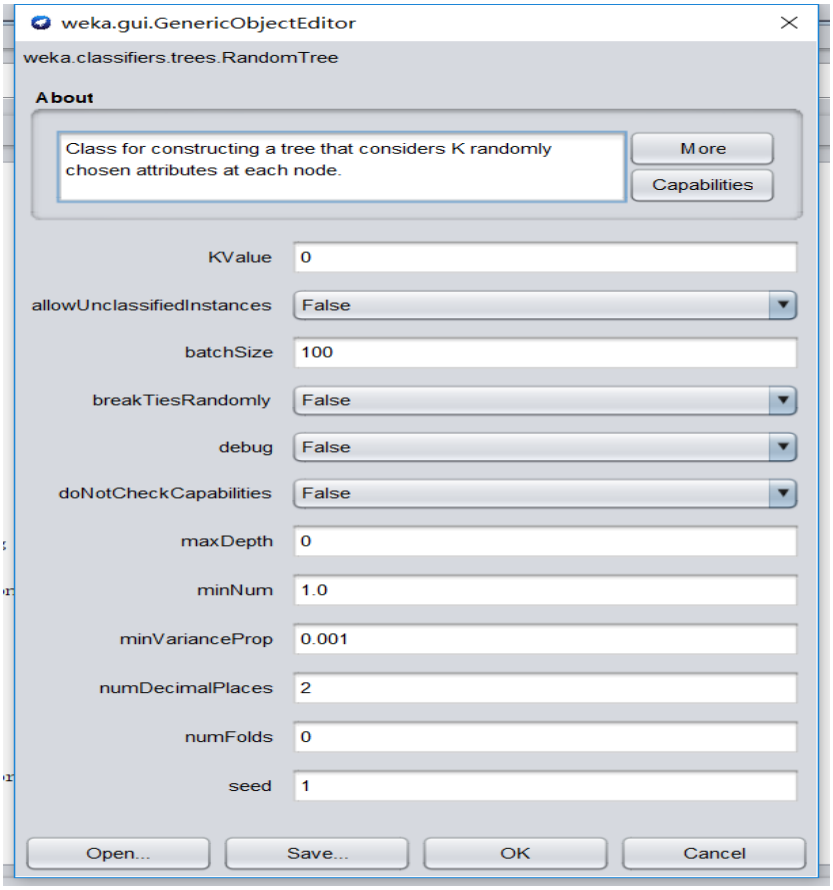



图 2.3

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。

## 2. 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

```
winner                0
firstBlood            0
firstTower            0
firstInhibitor        0
firstBaron            0
firstDragon           0
firstRiftHerald       0
t1_champ1_sum1        0
t1_champ1_sum2        0
t1_champ2_sum1        0
t1_champ2_sum2        0
t1_champ3_sum1        0
t1_champ3_sum2        0
t1_champ4_sum1        0
t1_champ4_sum2        0
t1_champ5_sum1        0
t1_champ5_sum2        0
t1_towerKills         0
t1_inhibitorKills     0
t1_baronKills         0
t1_dragonKills        0
t1_riftHeraldKills    0
t1_ban1               0
t1_ban2               0
t1_ban3               0
t1_ban4               0
t1_ban5               0
t2_champ1_sum1        0
t2_champ1_sum2        0
t2_champ2_sum1        0
t2_champ2_sum2        0
t2_champ3_sum1        0
t2_champ3_sum2        0
t2_champ4_sum1        0
t2_champ4_sum2        0
t2_champ5_sum1        0
t2_champ5_sum2        0
t2_towerKills         0
t2_inhibitorKills     0
t2_baronKills         0
t2_dragonKills        0
t2_riftHeraldKills    0
t2_ban1               0
t2_ban2               0
t2_ban3               0
t2_ban4               0
t2_ban5               0
dtype: int64
```

图 2.4

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。

### 3. 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）

```
In [13]: lol[(lol['t1_baronKills']>0) & (lol['gameDuration']<20*60)]
Out[13]:
```

| gameId | creationTime | gameDuration | seasonId | winner | firstBlood | firstTower | firstInhibitor | firstBaron | firstDragon | ... | t2_towerKills | t2_inhibitorKills | t2_baror |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

0 rows × 61 columns

```
In [17]: lol[(lol['t1_towerKills'])>11]
Out[17]:
```

| gameId | creationTime | gameDuration | seasonId | winner | firstBlood | firstTower | firstInhibitor | firstBaron | firstDragon | ... | t2_towerKills | t2_inhibitorKills | t2_baror |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

0 rows × 61 columns

图 2.5

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。

## 4. 相关属性单项分析与属性值关联分析

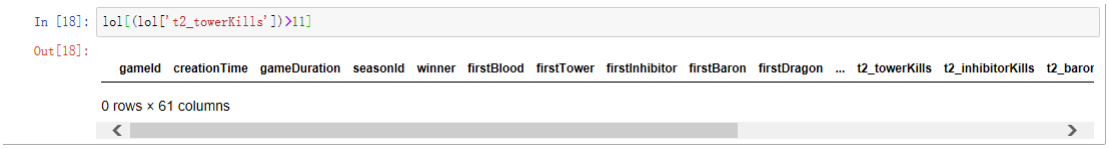



图 2.6

# 四、算法的设计与参数选择

## 1. 基于 Python sklearn 库实现的 CART 算法

### 1.1 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：

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。

### 1.2 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：

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 | 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 |

| | | |
|---|---|---|
| | 本数 | 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。

### 1.3 算法相关参数取值测试

splitter 参数测试：

取’best’时测试结果：

```
当前路径下文件：['www', 'flask2.py', 'ssl', '24dian', 'jupyterenv']
当前参数：
 criterion=gini,
splitter=best
min_impurity_decrease=默认
min_samples_split=默认
min_impurity_split=默认
测试所得准确率：0.959862756522302
```

取’random’时测试结果：

```
当前路径下文件：['www', 'flask2.py', 'ssl', '24dian', 'jupyterenv']
当前参数：
 criterion=gini,
splitter=random
min_impurity_decrease=默认
min_samples_split=默认
min_impurity_split=默认
测试所得准确率：0.958179581795818
```

| 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’.

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

```
当前参数：
 criterion=gini,
splitter=best
max_depth=1
min_impurity_decrease=默认
min_samples_split=默认
min_impurity_split=默认
测试所得准确率：0.8874862432834855
```

max_depth 取 3

```
当前参数：
 criterion=gini,
splitter=best
max_depth=3
min_impurity_decrease=默认
min_samples_split=默认
min_impurity_split=默认
测试所得准确率：0.9524179452320839
```

max_depth 取 5

```
当前参数：
 criterion=gini,
splitter=best
max_depth=5
min_impurity_decrease=默认
min_samples_split=默认
min_impurity_split=默认
测试所得准确率：0.9663365054703179
```

max_depth 取 7

```
当前参数：
 criterion=gini,
splitter=best
max_depth=7
min_impurity_decrease=默认
min_samples_split=默认
min_impurity_split=默认
测试所得准确率：0.9666601929177187
```

max_depth 取 9

```
当前参数：
 criterion=gini,
splitter=best
max_depth=9
min_impurity_decrease=默认
min_samples_split=默认
min_impurity_split=默认
测试所得准确率：0.9665307179387583
```

max_depth 取 11

```
当前参数：
 criterion=gini,
splitter=best
max_depth=11
min_impurity_decrease=默认
min_samples_split=默认
min_impurity_split=默认
测试所得准确率：0.9621933061435878
```

max_depth 取 13

```
当前参数：
 criterion=gini,
splitter=best
max_depth=13
min_impurity_decrease=默认
min_samples_split=默认
min_impurity_split=默认
测试所得准确率：0.960251181459183
```

max_depth 取 15

```
当前参数：
 criterion=gini,
splitter=best
max_depth=15
min_impurity_decrease=默认
min_samples_split=默认
min_impurity_split=默认
测试所得准确率：0.9592153816275005
```

max_depth 取 17

```
当前参数：
 criterion=gini,
splitter=best
max_depth=17
min_impurity_decrease=默认
min_samples_split=默认
min_impurity_split=默认
测试所得准确率：0.9584385317537386
```

max_depth 取 19

```
当前参数：
 criterion=gini,
splitter=best
max_depth=21
min_impurity_decrease=默认
min_samples_split=默认
min_impurity_split=默认
测试所得准确率：0.9580501068168576
```

max_depth 取 21

```
当前参数：
 criterion=gini,
splitter=best
max_depth=21
min_impurity_decrease=默认
min_samples_split=默认
min_impurity_split=默认
测试所得准确率：0.9580501068168576
```

max_depth 取 23

```
当前参数:
 criterion=gini,
splitter=best
max_depth=23
min_impurity_decrease=默认
min_samples_split=默认
min_impurity_split=默认
测试所得准确率: 0.9585032692432187
```

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.

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）

```
当前参数:
 criterion=gini,
splitter=best
max_depth=7
min_impurity_decrease=0
min_samples_split=默认
min_impurity_split=默认
测试所得准确率: 0.9669838803651194
```

min_impurity_decrease 取值为 0.05

```
当前参数:
 criterion=gini,
splitter=best
max_depth=7
min_impurity_decrease=0.05
min_samples_split=默认
min_impurity_split=默认
测试所得准确率: 0.9249692496924969
```

min_impurity_decrease 取值为 0.1

```
当前参数:
 criterion=gini,
splitter=best
max_depth=7
min_impurity_decrease=0.1
min_samples_split=默认
min_impurity_split=默认
测试所得准确率: 0.8874862432834855
```

min_impurity_decrease 取值为 0.15

```
当前参数：
 criterion=gini,
splitter=best
max_depth=7
min_impurity_decrease=0.1500000000000000
min_samples_split=默认
min_impurity_split=默认
测试所得准确率: 0.8874862432834855
```

min_impurity_decrease 取值为 0.2

```
当前参数：
 criterion=gini,
splitter=best
max_depth=7
min_impurity_decrease=0.2
min_samples_split=默认
min_impurity_split=默认
测试所得准确率: 0.8874862432834855
```

min_impurity_decrease 取值为 0.25

```
当前参数：
 criterion=gini,
splitter=best
max_depth=7
min_impurity_decrease=0.25
min_samples_split=默认
min_impurity_split=默认
测试所得准确率: 0.8874862432834855
```

min_impurity_decrease 取值为 0.3

```
当前参数：
 criterion=gini,
splitter=best
max_depth=7
min_impurity_decrease=0.3
min_samples_split=默认
min_impurity_split=默认
测试所得准确率: 0.8874862432834855
```

min_impurity_decrease 取值为 0.35

```
当前参数：
 criterion=gini,
splitter=best
max_depth=7
min_impurity_decrease=0.35
min_samples_split=默认
min_impurity_split=默认
测试所得准确率: 0.5099372046352042
```

（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.

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

```
当前参数:
 criterion=gini,
splitter=best
max_depth=7
min_impurity_decrease=0
min_samples_split=2
测试所得准确率: 0.966789667896679
```

min_samples_split 取 82

```
当前参数:
 criterion=gini,
splitter=best
max_depth=7
min_impurity_decrease=0
min_samples_split=82
测试所得准确率: 0.9663365054703179
```

……(中间部分测试结果图略)

min_samples_split 取 802

```
当前参数:
 criterion=gini,
splitter=best
max_depth=7
min_impurity_decrease=0
min_samples_split=802
测试所得准确率: 0.9600569689907426
```

min_samples_split 取 882

```
当前参数:
 criterion=gini,
splitter=best
max_depth=7
min_impurity_decrease=0
min_samples_split=882
测试所得准确率: 0.9593448566064608
```

min_samples_split 取 882

```
当前参数:
 criterion=gini,
splitter=best
max_depth=7
min_impurity_decrease=0
min_samples_split=962
测试所得准确率: 0.9519647828057228
```

（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.

### 1.4 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图 3.1 所示：

```python
#参数字典
pram={
    'criterion':'gini',
    'splitter':'best',
    'max_depth':7,
    'min_impurity_decrease':0,#节点划分最小不纯度
    'min_samples_split':802,#内部节点再划分所需最小样本数

    }
```

图 3.1

其余参数均使用默认参数。

```
测试所得准确率: 0.9600569689907426
模型详细参数: {'class_weight': None, 'min_impurity_split': None, 'splitter': 'best',
'presort': False, 'min_samples_split': 802, 'criterion': 'gini', 'min_weight_fraction_l
eaf': 0.0, 'max_features': None, 'min_impurity_decrease': 0, 'min_samples_leaf': 1, 'ma
x_leaf_nodes': None, 'max_depth': 7, 'random_state': None}
```

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 3.2 所示：

决策树：（为便于显示，将决策树可视化结果分割成两部分）

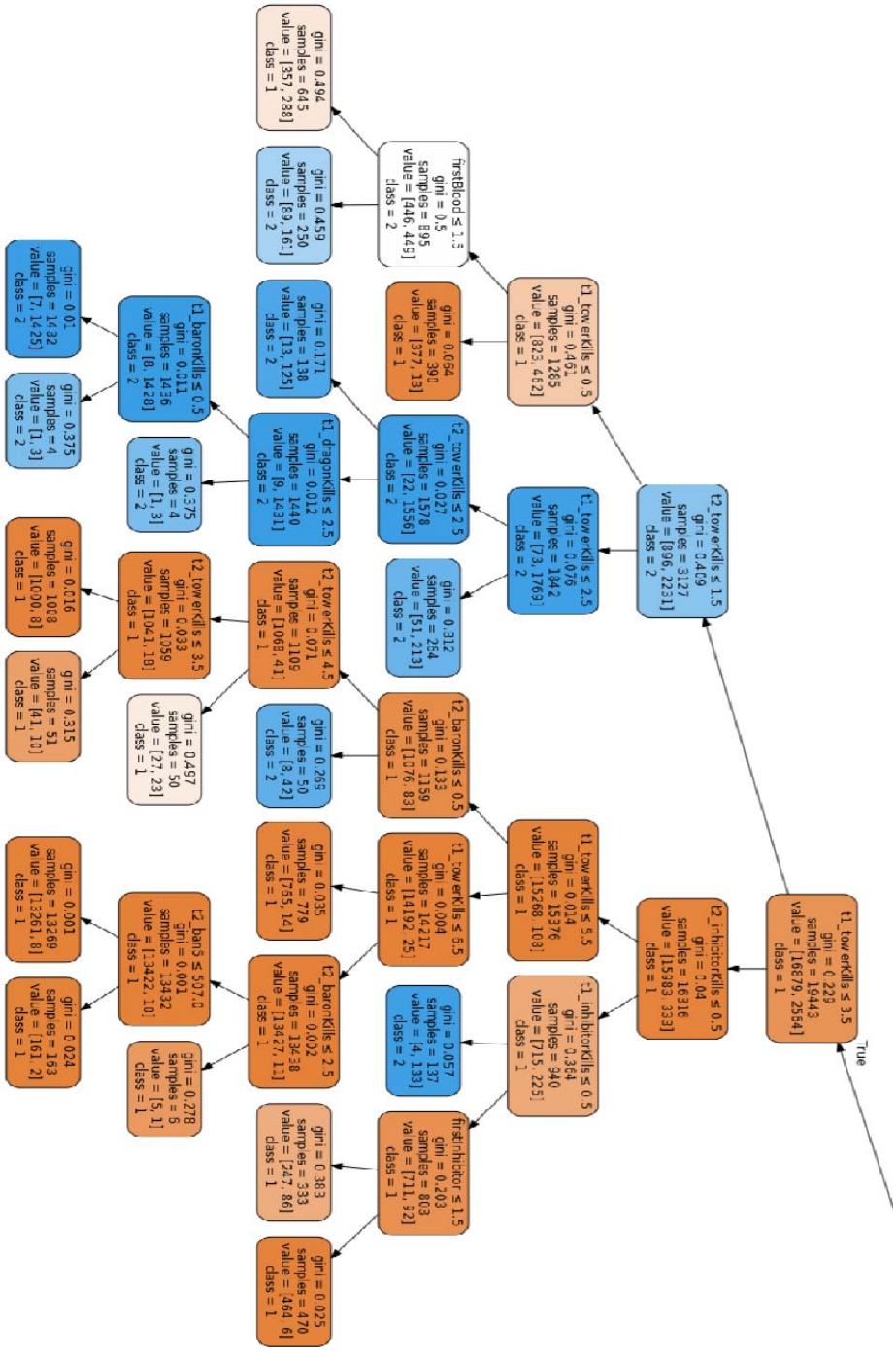



图 3.2

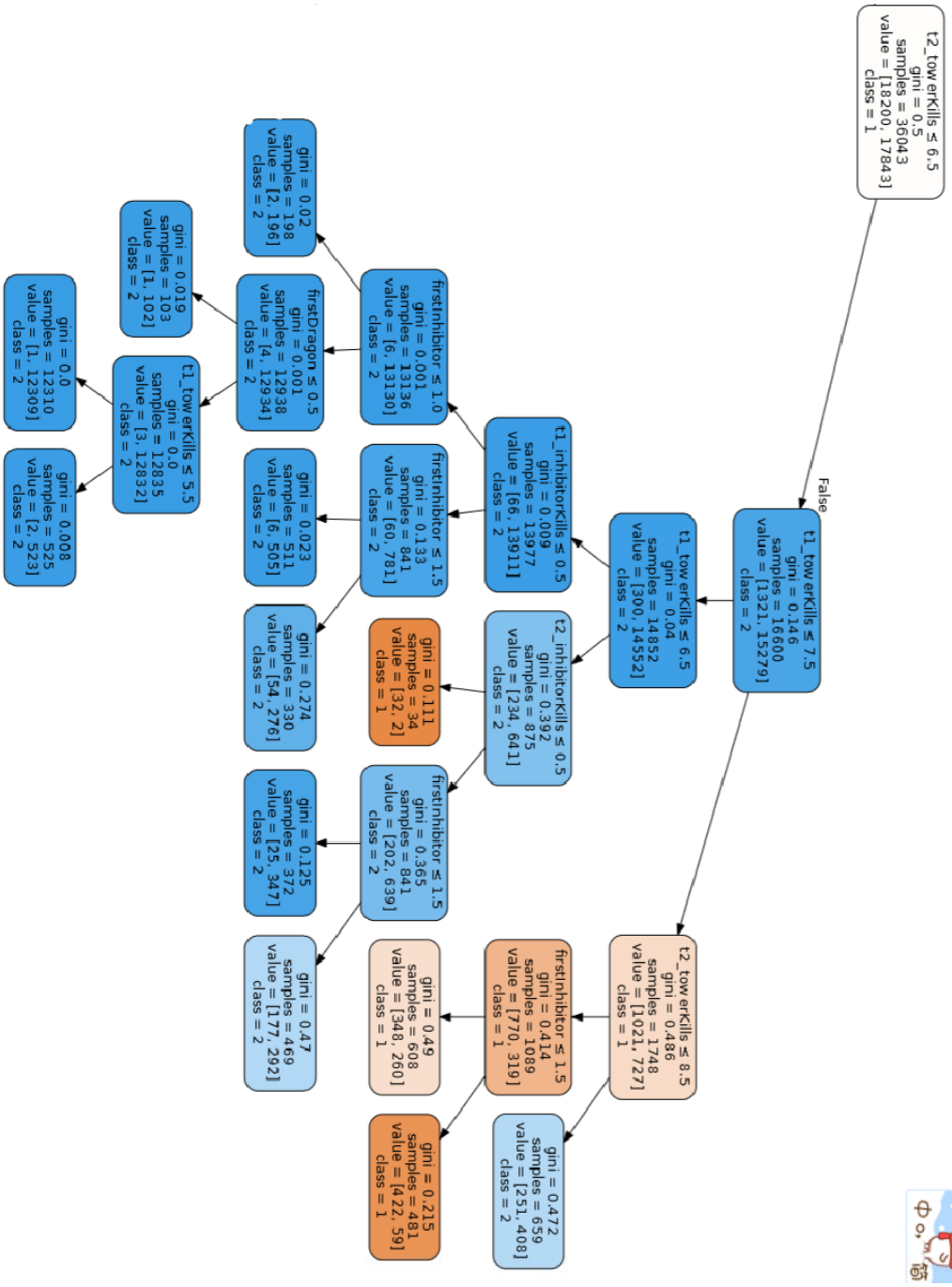



图 3.3

模型测试：

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：

1队获胜的概率 0.9872340425531915
2队获胜的概率 0.01276595744680851
获胜队伍是： 1

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：

Classification report :

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 1 | 0.96 | 0.96 | 0.96 | 7877 |
| 2 | 0.96 | 0.96 | 0.96 | 7570 |
| accuracy | | | 0.96 | 15447 |
| macro avg | 0.96 | 0.96 | 0.96 | 15447 |
| weighted avg | 0.96 | 0.96 | 0.96 | 15447 |

图 3.4

## 2. 基于 weka 实现的 Random Forest 算法

### 2.1 随机森林算法介绍

随机森林（Random Forest，简称 RF）是通过集成学习的思想将多棵树集成的一种算法，它的基本单元是决策树。是用随机的方式建立一个森林，森林里面有很多的决策树组成，随机森林的每一棵决策树之间是没有关联的。在得到森林之后，当有一个新的输入样本进入的时候，就让森林中的每一棵决策树分别进行一下判断，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。

随机森林的基本参数类型如下：

| 参数 | 类型 | 说明 |
|---|---|---|
| class_weight | 目标 | 类别的权值 |

| | | |
|---|---|---|
| n_estimators | 性能 | 子模型的数量 |
| max_features | 性能 | 节点分裂时参与的最大分裂数 |
| max_depth | 性能 | 最大深度 |
| min_samples_split | 性能 | 分裂所需的最小样本数 |
| min_sample_leaf | 性能 | 叶子节点最小样本数 |
| max_leaf_nodes | 性能 | 最大叶子节点数 |

## 2.2 相关参数测试

**max_depth 参数测试**

结合数据集的情况，我们将 max_depth 的取值分别设为 1，10,11,12,13,14,15,20，0（0 代表不限最大深度）。测试结果如下：

```
=== Summary ===

Correlation coefficient                  0.8725
Mean absolute error                      0.2692
Root mean squared error                  0.2986
Relative absolute error                 53.8421 %
Root relative squared error             59.7272 %
Total Number of Instances           51485
```

max_depth = 1

```
=== Summary ===

Correlation coefficient                  0.9695
Mean absolute error                      0.0432
Root mean squared error                  0.1232
Relative absolute error                  8.6443 %
Root relative squared error             24.6488 %
Total Number of Instances           51485
```

max_depth = 10

```
=== Summary ===

Correlation coefficient                  0.9872
Mean absolute error                      0.025
Root mean squared error                  0.0807
Relative absolute error                  4.9935 %
Root relative squared error             16.1446 %
Total Number of Instances           51485
```

max_depth = 15

max_depth = 20

```
=== Summary ===

Correlation coefficient                  0.993
Mean absolute error                      0.0193
Root mean squared error                  0.0605
Relative absolute error                  3.8647 %
Root relative squared error             12.1    %
Total Number of Instances            51485
```

```
=== Summary ===

Correlation coefficient                  0.9945
Mean absolute error                      0.0174
Root mean squared error                  0.0535
Relative absolute error                  3.488  %
Root relative squared error             10.7102 %
Total Number of Instances            51485
```

max_depth = 0

可以看到，在由小到大测试中并没有出现极大值，总体趋势是 Correctly Classified Instances 的比例随 max_depth 的增大而增大，故在之后的测试中为了得到正确率最高的结果取 max_depth = 0

**num_features 参数测试**

num_features 代表节点分裂时参与判断的特征数，在 weka 中其默认值是 0（不限特征数）我们取 num_features 值为 2,3,5,0 测试正确率。测试结果如下：

```
=== Summary ===

Correlation coefficient                  0.9939
Mean absolute error                      0.0235
Root mean squared error                  0.0579
Relative absolute error                  4.705  %
Root relative squared error             11.5887 %
Total Number of Instances            51485
```

num_features = 2

num_features = 5

```
=== Summary ===

Correlation coefficient                  0.9945
Mean absolute error                      0.0159
Root mean squared error                  0.0536
Relative absolute error                  3.1838 %
Root relative squared error             10.7202 %
Total Number of Instances           51485
```

num_features = 10

```
=== Summary ===

Correlation coefficient                  0.9945
Mean absolute error                      0.0174
Root mean squared error                  0.0535
Relative absolute error                  3.488  %
Root relative squared error             10.7102 %
Total Number of Instances           51485
```

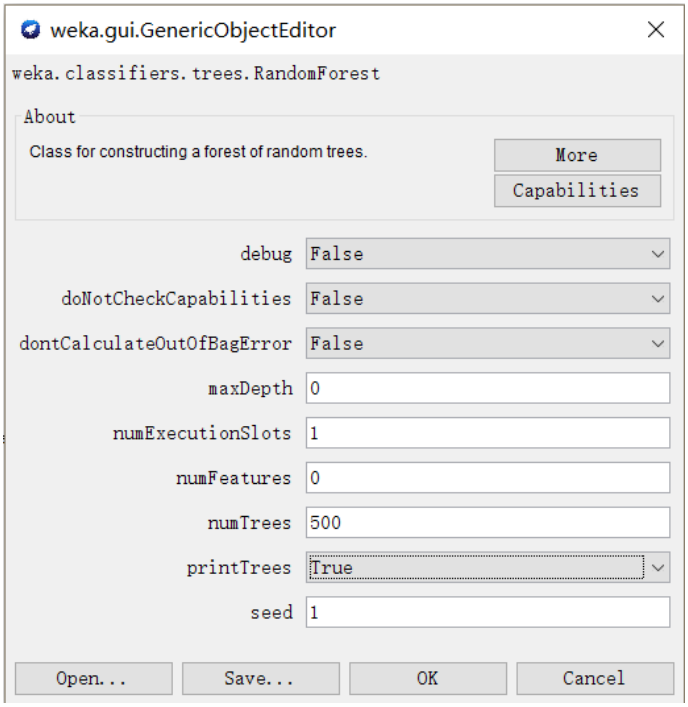
```
=== Summary ===

Correlation coefficient                  0.9945
Mean absolute error                      0.0174
Root mean squared error                  0.0535
Relative absolute error                  3.488  %
Root relative squared error             10.7102 %
Total Number of Instances           51485
```

num_features = 0

同 max_depth 参数一样，num_features 从小到大取值没有拐点，总体趋势是正确率随 num_features 增大而增大，故取其值为 0

### 2.3 最终测试




各参数调整完毕后进行最后的测试，最终测试的参数选择如图 3.5：

图 3.5

测试结果如图 3.6：

```
=== Summary ===

Correlation coefficient                  0.9947
Mean absolute error                      0.0176
Root mean squared error                  0.0527
Relative absolute error                  3.5252 %
Root relative squared error             10.5494 %
Total Number of Instances           51485
```

图 3.6

# 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）

```
import numpy as np
import pandas as pd #用于 csv 数据读入及后续处理
from plotly.offline import init_notebook_mode,iplot
init_notebook_mode(connected=True)
import warnings
warnings.filterwarnings("ignore")
import os
print('当前路径下文件：'+str(os.listdir("/data")))#列出当前路径下的文件列表
lol=pd.read_csv("/data/jupyterenv/games.csv")
```

```
y = lol["winner"].values
x = lol.drop(["winner"],axis=1)
from sklearn.model_selection import train_test_split
#生成测试集合训练集
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.3,random_state=1)
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

#设置参数字典
pram={'criterion':'gini',
      'splitter':'best',
      'max_depth':7,
      'min_impurity_decrease':0,#节点划分最小不纯度
      'min_samples_split':802,#内部节点再划分所需最小样本数

                          }
#用于参数 max_depth 的测试
'''
for i in range(2,22,2):
      pram['max_depth']=i
      dt2=DecisionTreeClassifier(criterion=pram['criterion'],
max_depth=pram['max_depth'],splitter=pram['splitter'])
      dt2.fit(x_train,y_train)
      print('当前参数：\n
criterion={},\nsplitter={}\nmax_depth={}\nmin_impurity_decrease={}\nmin_samples_split={}\nmin_impurity_split={}'.format(pram['criterion'],pram['splitter'],pram['ma
```

```
x_depth'],pram['min_impurity_decrease'],pram['min_samples_split']))
            print("测试所得准确率:", dt2.score(x_test,y_test))
        '''
        #用于参数 min_impurity_decrease 的测试
        '''
        min_impurity_decrease=0
        for i in range(1,30):
            pram['min_impurity_decrease']=min_impurity_decrease
            dt2=DecisionTreeClassifier(criterion=pram['criterion'],
max_depth=pram['max_depth'],splitter=pram['splitter'],min_impurity_decrease=pram
['min_impurity_decrease'])
            dt2.fit(x_train,y_train)
            print('当前参数:\n
criterion={},\nsplitter={}\nmax_depth={}\nmin_impurity_decrease={}\nmin_sample
s_split={}\nmin_impurity_split={}'.format(pram['criterion'],pram['splitter'],pram['ma
x_depth'],pram['min_impurity_decrease'],pram['min_samples_split']))
            print("测试所得准确率:", dt2.score(x_test,y_test))
            min_impurity_decrease+=0.1
        '''
        '''
        #用于测试 min_samples_split 参数
        min_samples_split=2
        for i in range(20):
                pram['min_samples_split']=min_samples_split
                dt2=DecisionTreeClassifier(criterion=pram['criterion'],
max_depth=pram['max_depth'],splitter=pram['splitter'],min_impurity_decrease=pram
['min_impurity_decrease'],min_samples_split=pram['min_samples_split'])
                dt2.fit(x_train,y_train)
```

```
            print('当前参数：\n criterion={},\nsplitter={}\nmax_depth={}\nmin_impurity_decrease={}\nmin_samples_split={}'.format(pram['criterion'],pram['splitter'],pram['max_depth'],pram['min_impurity_decrease'],pram['min_samples_split']))
            print("测试所得准确率:", dt2.score(x_test,y_test))
            min_samples_split+=80
    '''
    dt2=DecisionTreeClassifier(criterion=pram['criterion'], max_depth=pram['max_depth'],splitter=pram['splitter'],min_impurity_decrease=pram['min_impurity_decrease'],min_samples_split=pram['min_samples_split'])
    dt2.fit(x_train,y_train)
    print('当前参数:\ncriterion={},\nsplitter={}\nmax_depth={}\nmin_impurity_decrease={}\nmin_samples_split={}'.format(pram['criterion'],pram['splitter'],pram['max_depth'],pram['min_impurity_decrease'],pram['min_samples_split']))
    print("测试所得准确率:", dt2.score(x_test,y_test)) #返回给定测试集和对应标签的平均准确率
    #print(dt2.decision_path(x))  #返回 X 的决策路径
    #使用 graphviz 生成可视化决策树
    from sklearn import tree
    import graphviz
    dot_data = tree.export_graphviz(dt2, out_file=None, feature_names=x.columns,class_names=["1","2"],
                                    filled=True, rounded=True, special_characters=True)
    graph = graphviz.Source(dot_data)
    graph
    #作模型测试
    testone=dt2.predict__proba([[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]])
```

```
#提供一个测试集 返回测试结果为每个类别的可能百分比，所有类别百分比构成一个数组，概率和为 1
    print('1 队获胜的概率',testone[0][0])
    print('2 队获胜的概率',testone[0][1])
    result=dt2.predict([[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]]) #返回数组，由每个较高概率标签组成
    print('获胜队伍是：',result[0])
    #生成分类报告
    from sklearn.metrics import confusion_matrix,classification_report
    predicted_values = dt2.predict(x_test)
    cr=classification_report(y_test,predicted_values)
    print('Classification report : \n',cr)
```