

基于 NBA 数据的球员类型分析系统

——球探

小组成员：徐钰东 何恒朗 艾俊涛 王哲

班级：计算机 1604

摘要

本次大作业我们选择的题目是基于 NBA 数据的球员类型分析系统，主要的启发源自现在 NBA 的数据分析是比较全面的，很多球队的球探和数据分析师都会通过对球员比赛数据进行分析，来达到其提高战绩，招募适合球员等的目的。该系统目的主要可以通过输入球员的几项重要数据如得分、位置、篮板等，对球员的类型实现判断和分类，如全能巨星，主力球员等不同的类别。我们的系统一方面可以用来给专业人士做一些分析和判断，另一方面也为广大篮球爱好者提供了了解自己感兴趣球员的各方面数据能力的路径，同时我们在球员测评结果下面针对每个位置设置了代表球员的打法风格的代表人物及其打法特点，我们在每一类球员下面都设置了打法教学这一栏，帮助大家去细致地学习和吸纳你喜欢类型球员优秀代表的技巧打法，篮球是一项体育运动，虽然说实践出真知，但是如果不去仔细地钻研技巧和细节，在打球中反复去琢磨应用，为自己所用，是很难有较大的提升的，所以这一功能也是为了篮球爱好者找到正确的方向，提高球技而设计的。

我们收集并采用了 07-17 年间其中 6 个赛季的 NBA 球员的比赛数据，通过一定的标准，对其得分、进攻篮板、防守篮板、助攻、抢断、盖帽六项属性进行评级，按照得分能力，防守能力的相对强弱，将球员分为五个类型，也可以说是级别。为球员加上了专属的标签。

在训练模型时主要的思路是先利用简单的分类学习器 KNN，SVM，决策树训练模型来得到结果。之后我们主要应用到了集成学习的算法思想，这个是我们算法的核心部分，通过将多个基本分类器集成一个较复杂的分类学习器模型，我们采用的有基于 Bagging 策略的随机森林（RF），基于 Boosting 策略的 AdaBoost，还有基于 Stacking 策略的集成学习模型。我们再利用正确率，标准来比较各类模型的效果，通过衡量选定最好的模型应用在我们的系统之中。再加上 GUI 的设计，构成了我们最后的系统。

关键词：NBA 大数据、球员多分类、集成学习

1. 介绍

1.1 背景及设计目的

● 背景：

第一方面是就专业角度而言，现在是一个大数据时代，NBA 赛场也是一个产生庞大数据库的地方，对每个球员的表现分析不仅体现在临场的发挥上，还可以通过分析球员数据来对球队、球员的能力，打法各方面进行评测。也有许多的球探，数据分析师，甚至主教练都会利用数据来更好地进行球队部署、球员的挑选。在很多时候都会起到很好的作用。另一方面，其实近些年篮球运动是兴起的，很多篮球爱好者喜欢看球，喜欢打球。对职业球员的数据表现也比较感兴趣，也会去学习自己喜欢的或者想发展成为类型的优秀球员的技巧。其实很多时候人们会强调去实战，真正去分析技术动作的人可能相对并没有那么多，但篮球名宿科比·布莱恩特曾经就说过他觉得篮球水平提高关键点在于细节的把握和球场上的专

注。而细节更多的需要我们观察优秀球员的场上表现的技术动作细节去学习和领悟。

● 设计目的：

我们设计的分类系统主要是通过对球员得分、进攻篮板、防守篮板、助攻、抢断、盖帽各项属性数据进行分析判断得出球员综合能力、进攻能力、防守能力、各项属性数据的能力高低。因为球员的数据是多而庞杂的，而且专业数据分析对其都有很精密的计算公式和分析方法。只采用简单的几项数据是难以很好地得出球员的准确特征。但是他们的每一类球员综合能力强弱的特征其实依据数据具有较好的分布特点。我们只能以较为合理的标准去为其分类，虽然无法保证完全地准确，但是也是满足实际的大体分布的。

我们在每一个球员测评后得到的界面会依据球员的位置归属显示得到该位置不同打法优秀球员的代表人物，及对应的视频教学链接。这部分主要是为了让使用者能根据自己想学习的球员类型打法有针对性地去观看视频并学得经验，在欣赏了精彩的视频之余自己的球技也有所提升。

2. 准备工作

在完成本次大作业之前我们也是做了很多包括收集数据，查找文献，探讨设计的合理性和可行性，接下来将对我们的准备进行具体的阐述。

2.1 数据收集

我们的数据来源是：<http://www.stat-nba.com/>

这是一个很全面的 NBA 历史数据网站。

小贴士 点击查询结果的表头即可按此项目排序

	球员	赛季	球队	出场	首发	时间	投篮	命中	出手	三分	命中	出手	罚球	命中	出手	篮板	前场	后场	助攻	抢断	盖帽	失误	犯规	得分	胜	负
1	詹姆斯-哈登	17-18	火箭	72	72	35.4	44.9%	9.0	20.1	36.7%	3.7	10.0	85.8%	8.7	10.1	5.4	0.6	4.8	8.8	1.8	0.7	4.4	2.3	30.4	59	13
2	安东尼-戴维斯	17-18	鹈鹕	75	75	36.4	53.4%	10.4	19.5	34.0%	0.7	2.2	82.8%	6.6	8.0	11.1	2.5	8.6	2.3	1.5	2.6	2.2	2.1	28.1	45	30
3	勒布朗-詹姆斯	17-18	骑士	82	82	36.9	54.2%	10.5	19.3	36.7%	1.8	5.0	73.1%	4.7	6.5	8.6	1.2	7.5	9.1	1.4	0.9	4.2	1.7	27.5	50	32
4	达米安-利拉德	17-18	开拓者	73	73	36.6	43.9%	8.5	19.4	36.1%	3.1	8.6	91.6%	6.8	7.4	4.5	0.8	3.6	6.6	1.1	0.4	2.8	1.6	26.9	44	29
5	扬尼斯-阿德托昆博	17-18	雄鹿	75	75	36.8	52.9%	9.9	18.7	30.7%	0.6	1.9	76.0%	6.5	8.5	10.0	2.1	8.0	4.8	1.5	1.4	3.0	3.1	26.9	39	36
6	凯文-杜兰特	17-18	勇士	68	68	34.3	51.6%	9.3	18.0	41.9%	2.5	6.1	88.9%	5.3	5.9	6.8	0.5	6.4	5.4	0.7	1.8	3.0	2.0	26.4	49	19
7	拉塞尔-威斯布鲁克	17-18	雷霆	80	80	36.5	44.9%	9.5	21.1	29.8%	1.2	4.1	73.7%	5.2	7.1	10.1	1.9	8.2	10.3	1.8	0.3	4.8	2.5	25.4	47	33
8	凯里-欧文	17-18	凯尔特人	60	60	32.2	49.1%	8.9	18.1	40.8%	2.8	6.8	88.9%	3.9	4.4	3.8	0.6	3.2	5.1	1.1	0.3	2.3	2.0	24.4	41	19
9	拉玛库斯-阿尔德里奇	17-18	马刺	75	75	33.5	51.0%	9.2	18.0	29.3%	0.4	1.2	83.7%	4.5	5.3	8.5	3.3	5.2	2.0	0.6	1.2	1.5	2.1	23.1	45	30
10	维克多-奥拉迪波	17-18	步行者	75	75	34.1	47.7%	8.5	17.9	37.1%	2.1	5.8	79.9%	3.9	4.9	5.2	0.6	4.6	4.3	2.4	0.8	2.9	2.3	23.1	48	27
11	德玛尔-德罗赞	17-18	猛龙	80	80	33.9	45.6%	8.1	17.7	31.0%	1.1	3.6	82.5%	5.8	7.0	3.9	0.7	3.2	5.2	1.1	0.3	2.2	1.9	23.0	57	23
12	乔尔-恩比德	17-18	76人	63	63	30.4	48.3%	8.1	16.8	30.8%	1.0	3.4	76.9%	5.7	7.4	11.0	2.3	8.7	3.2	0.6	1.8	3.7	3.3	22.9	41	22
13	布拉德利-比尔	17-18	奇才	82	82	36.3	46.0%	8.3	18.1	37.5%	2.4	6.5	79.1%	3.6	4.5	4.4	0.7	3.7	4.5	1.2	0.4	2.6	2.0	22.6	43	39
14	路易斯-威廉姆斯	17-18	快船	79	19	32.8	43.5%	7.4	16.9	35.9%	2.4	6.6	88.0%	5.5	6.2	2.5	0.5	2.0	5.3	1.1	0.2	3.0	1.3	22.6	42	37
15	吉米-巴特勒	17-18	森林狼	59	59	36.7	47.4%	7.4	15.6	35.0%	1.2	3.4	85.4%	6.2	7.2	5.3	1.3	4.0	4.9	2.0	0.4	1.8	1.3	22.2	37	22
16	肯巴-沃克	17-18	黄蜂	80	80	34.2	43.1%	7.4	17.0	38.4%	2.9	7.5	86.4%	4.5	5.3	3.1	0.4	2.7	5.6	1.1	0.3	2.2	1.2	22.1	36	44
17	保罗-乔治	17-18	雷霆	79	79	36.6	43.0%	7.3	17.0	40.1%	3.1	7.7	82.2%	4.3	5.2	5.7	0.9	4.7	3.3	2.0	0.5	2.7	2.9	21.9	47	32
18	布雷克-格里芬	17-18		58	58	33.9	43.8%	7.5	17.2	34.5%	1.9	5.6	78.5%	4.5	5.7	7.4	1.3	6.1	5.8	0.7	0.3	2.8	2.4	21.4	28	30
19	C.J.迈克鲁姆	17-18	开拓者	81	81	36.1	44.3%	8.2	18.6	39.7%	2.3	5.9	83.6%	2.6	3.1	4.0	0.7	3.3	3.4	1.0	0.4	1.9	2.1	21.4	48	33
20	卡尔-安东尼-唐斯	17-18	森林狼	82	82	35.6	54.5%	7.8	14.3	42.1%	1.5	3.5	85.8%	4.2	4.9	12.3	2.9	9.4	2.4	0.8	1.4	1.9	3.5	21.3	47	35

我们的数据集主要是由 2007-2008、2009-2010、2011-2012、2013-2014、2015-2016、2017-2018 六个赛季共 1569 个球员的数据。我们采用的是得分、进攻篮板、防守篮板、助攻、抢断、盖帽六个属性来进行分类，得到待分类的五种标签：当家巨星、主力球员、主力轮换球员、替补球员、饮水机守护神（边缘球员）。

2.2 组内分工

- 何恒朗：数据集的收集和处理，K-means 算法和 KNN 算法的设计。
- 艾俊涛：SVM、Adaboost 算法的设计，各个模型的性能比较分析。
- 徐钰东：数据集标签的确立，决策树算法、随机森林算法、基于 Stacking 策略的集成学习分类器的设计，报告的书写。
(报告对应的技术部分由专门负责该功能的同学书写，我负责的技术部分、各部分的集合、总体框架由我完成。)
- 王哲：图形界面 GUI 的设计，GUI 和模型部分的结合。

2.3 学习准备

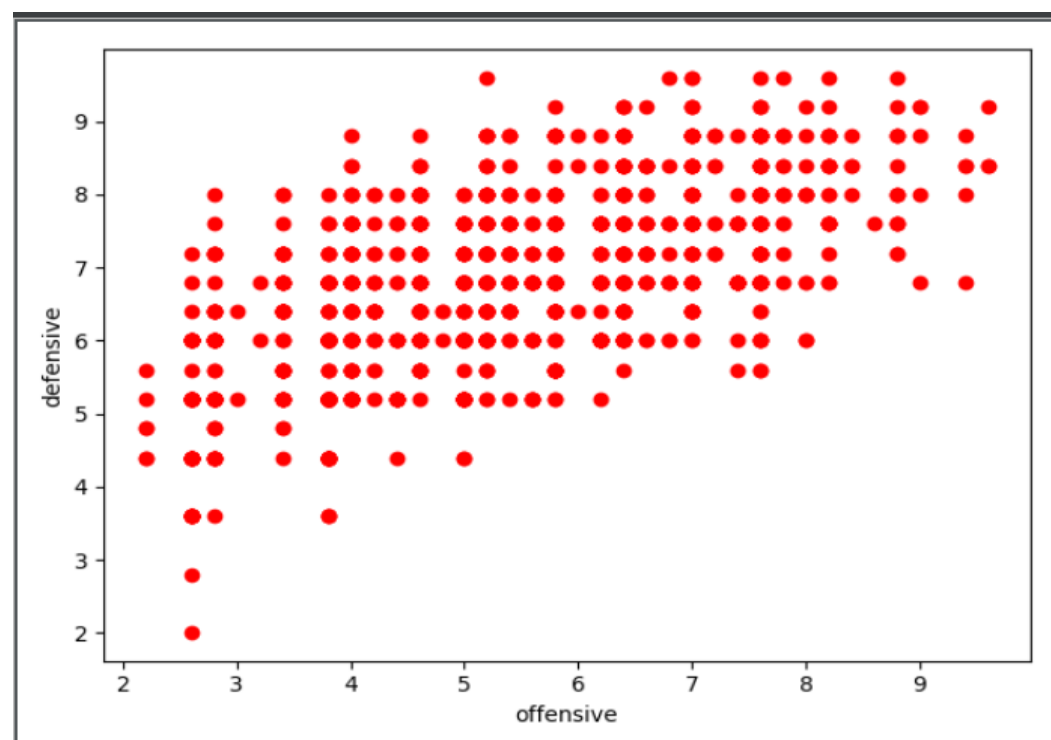
我们这次要解决的问题是一个多分类问题。初始的数据集是无标签的一个数据集，属于无监督学习。我们首先希望给球员的分类得到对应的标签。然后利用合适的分类模型来进行数据的训练。

标签的确立

我们利用每个球员的进攻属性（得分、进攻篮板、助攻），防守属性（防守篮板、抢断、盖帽），根据每一项数据的高低给球员的每一项数据划分得到对应的等级。然后依据球员场上位置的不同给球员的进攻和防守能力评级

位置	进攻能力	防守能力
内线（C 和 PF）	$P*0.6+A*0.1+ORB*0.3$	$DRB*0.4+S*0.2+B*0.4$
外线（SF、SG、PG）	$P*0.6+A*0.3+ORB*0.1$	$B*0.2+DRB*0.4+S*0.4$

(P:得分 A: 助攻 ORB: 进攻篮板 DRB: 防守篮板 S:抢断 B: 盖帽)



上面是我们得到的球员能力分布图（X 轴：进攻能力 Y 轴：防守能力）

可以发现球员类别或者说能力高低分布特征还是比较明显的，能力高的主要分布在右上角，从这一部分向左下依次能力降低。可以给定数据集对应的标签标准。

进攻能力	防守能力	球员等级
8-10	8-10	当家巨星
7-8	7.2-10	主力球员
6-8	6-7.6	主力轮换球员
6-7	7.2-10	主力轮换球员
4-7	5-9	替补球员
0-4	6-9	替补球员
0-4	0-6	边缘球员

就此，标签就顺利给定了。

利用 K-MEANS 做的尝试

- 我们一开始打算使用 K-Means 算法根据每个球员某两个数据通过聚簇来得到同一位置球员不同打法风格的区别。K-means 算法的基本思想是：以空间中 k 个点为形心进行聚类，对最靠近他们的对象归类。通过迭代的方法，逐次更新簇的形心的值，直至得到最好的聚类结果。（形心可以是实际的点、或者是虚拟点）。

建立该模型的原因

针对球员已分好的等级，对每个等级的球员根据所打位置的不同按其风格实现再分类（比如在第一等级的中锋球员可以分为“两双机器”、“篮下空气”、“禁区铁闸”、“潜力待挖掘”等风格），让系统更加完善和多样化，也可以引起系统的使用者更大的兴趣。

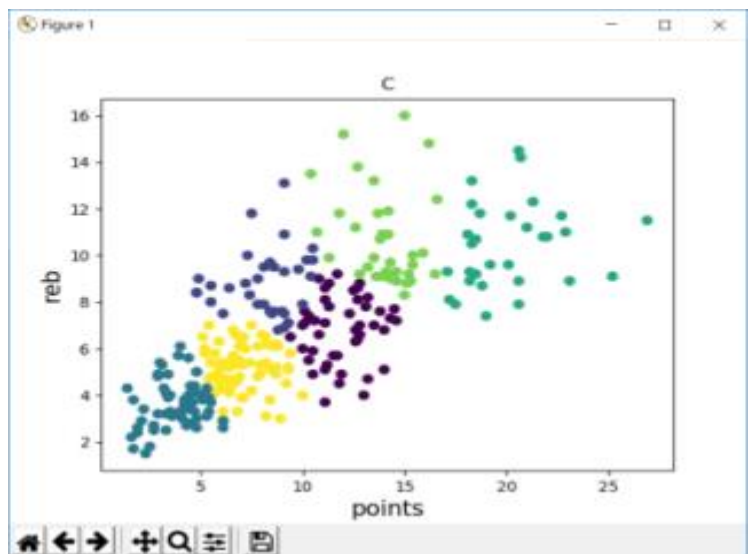
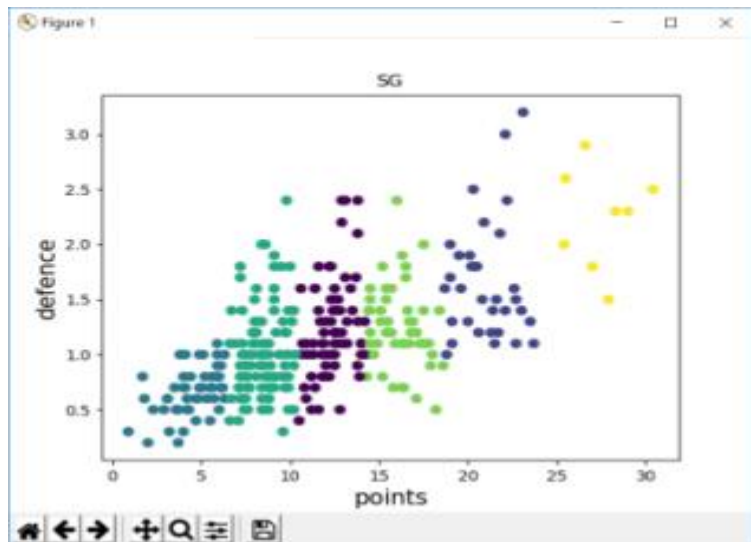
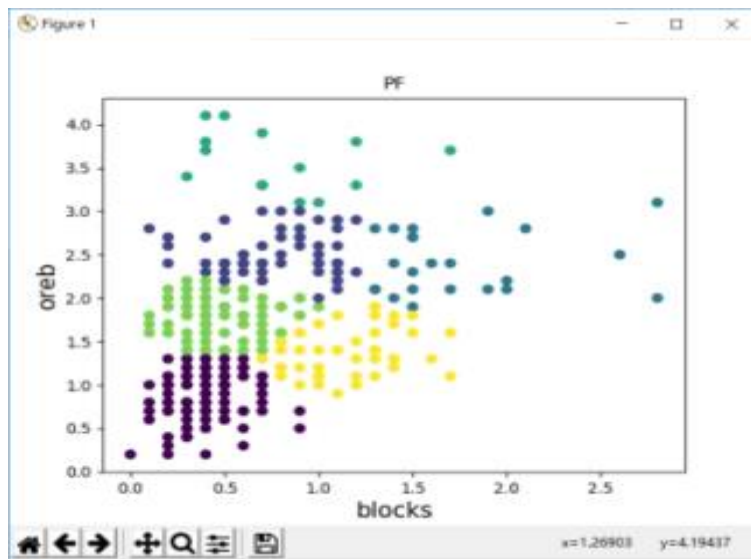
实现思路

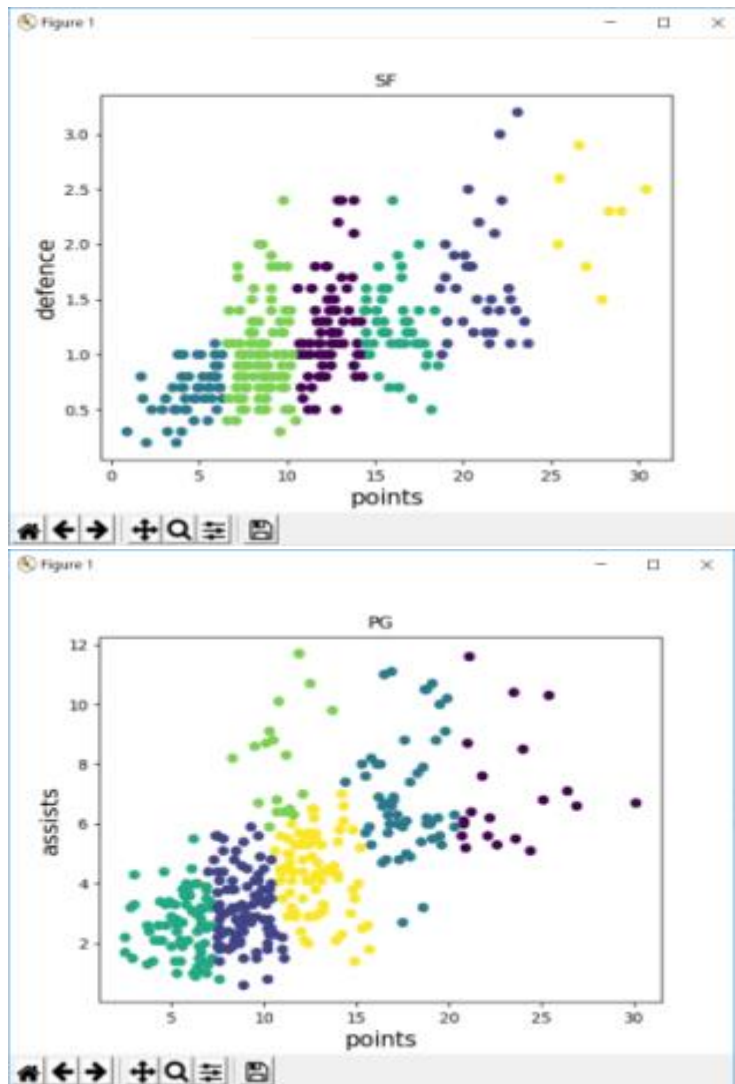
- 控球后卫：利用场均得分和场均助攻来分类；
- 得分后卫：利用场均得分和防守（场均抢断+场均盖帽）来分类；
- 小前锋：与得分后卫相同；
- 大前锋：利用场均盖帽和场均进攻篮板来分类；
- 中锋：利用场均得分和场均总篮板来分类。

结果图

每个位置球员经过多次聚类的情況如下

实际上我们通过聚类结果没有得到合适的风格打法的分类情况。因为对于打法这一特点的描述其实是很难通过几个简单属性就将其界定的。因此我们在最后的模型就没有采用其分类结果，取而代之的是在对应球员评测界面添加了我们熟知的几位代表球员。我们希望之后能找到合适的方法去实现风格打法的划分。

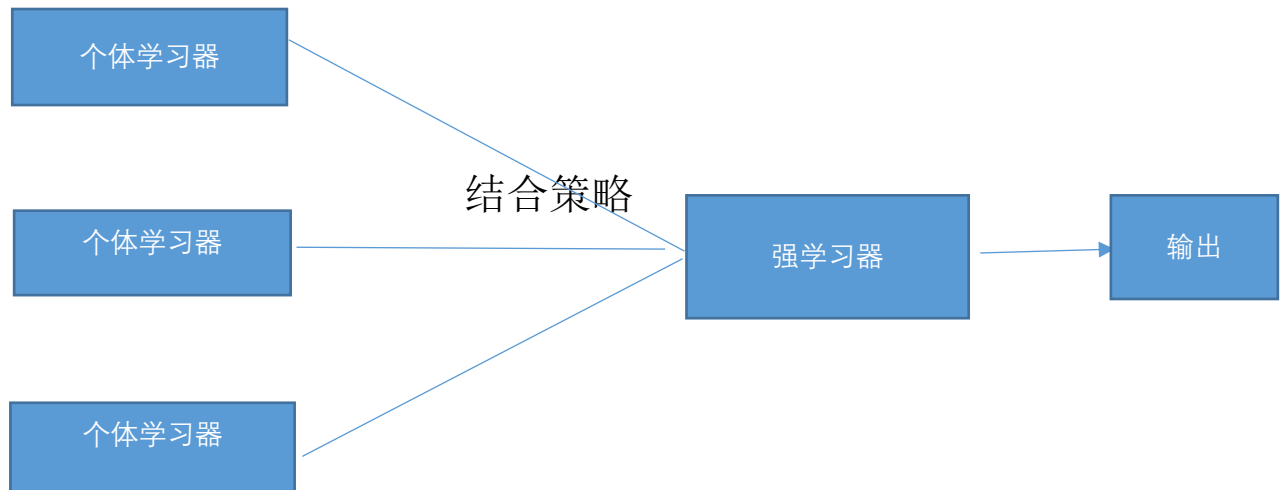




3. 建立模型

3.1 基本思想

- 我们的重点是在建立模型这一部分上。我们主要利用的是 Python 中的一个机器学习框架：Scikit-learn。Scikit-learn 的基本功能主要被分为六大部分：分类，回归，聚类，数据降维，模型选择和数据预处理。目前 Scikit-learn 已经实现的监督分类算法包括：支持向量机（SVM），最近邻，逻辑回归，随机森林，决策树以及多层感知器（MLP）神经网络等等。利用这个学习框架帮助我们将重点放在了寻找最好的分类器模型上。
- 我们首先建立了几个单个分类器模型来进行训练，之后又采用了集成学习的算法来将单个模型进行集成得到更加复杂的分类器。集成学习就是通过构建结合多个学习器来完成学习任务，有时候也被称为多分类器系统。通过训练若干个体学习器，通过一定的结合策略，最终形成一个强学习器。以得到泛化性能更好的分类器。



主要分类：

根据个体学习器生成方式分为：

- 个体学习器间存在强依赖关系、必须串行化生成的序列化方法，代表：Boosting 系列算法
- 个体学习器之间不存在强依赖关系、可同时生成的并行化方法，代表：Bagging 系列算法、随机森林
- （其中个体学习器又分为同质和异质）

按结合策略分：假定得到的 T 个弱分类器是 $\{h_1, h_2, \dots, h_T\}$

- 平均法：常用来得到回归问题的结果。对若干个弱分类器的输出进行平均得到最终的预测输出。
 - $H(x) = \left(\frac{1}{T}\right) \sum W_i h_i(x)$ （若权重 w 为 1 则为简单平均）
- 投票法：主要用于分类问题。假设我们的预测类别是 $\{c_1, c_2, \dots, c_K\}$ ，对于任意一个预测样本 x ，我们的 T 个弱学习器的预测结果分别是 $(h_1(x), h_2(x) \dots h_T(x))$ 。
 - 最简单的投票法是相对多数投票法，也就是我们常说的少数服从多数，也就是 T 个弱学习器的对样本 x 的预测结果中，数量最多的类别 c_i 为最终的分类类别。如果不止一个类别获得最高票，则随机选择一个做最终类别。
 - 稍微复杂的投票法是绝对多数投票法，也就是我们常说的要票过半数。在相对多数投票法的基础上，不光要求获得最高票，还要求票过半数。否则会拒绝预测。
 - 更加复杂的是加权投票法，和加权平均法一样，每个弱学习器的分类票数要乘以一个权重，最终将各个类别的加权票数求和，最大的值对应的类别为最终类别。

学习法：以上两种结合方法比较简单，可能会造成较大的学习误差，学习法就应运而生了。对于学习法，代表方法是 stacking，当使用 stacking 的结合策略时，我们不是对弱学习器的结果做简单的逻辑处理，而是再加上一层学习器，也就是

说，我们将训练集弱学习器的学习结果作为输入，将训练集的输出作为输出，重新训练一个学习器来得到最终结果。

因为各种个体分类器难免会存在一些局限性，可能会陷入局部最优，我们希望能通过集成学习得到更复杂的分类器来解决这一问题。

最后，我们希望通过比较各种分类器的效果，来找出一种最适合的分类器。

3.2 分类模型设计

3.2.1 简单分类器

● K 最近邻(k-Nearest Neighbor, KNN)分类算法

工作原理：

在监督学习中，我们知道样本集中每一数据与所属分类的对应关系。输入没有标签的新数据后，将新数据的每个特征与样本集中数据对应的特征进行比较，算法提取样本集中特征最相似数据（最近邻）的分类标签。（前 k 个最相似的数据，这就是 k -近邻算法中 k 的出处，通常 k 是不大于 20 的整数。）

最后，选择 k 个最相似数据中出现次数最多的分类，作为新数据的分类。

特点：

优点

- 1) 简单，易于理解，易于实现，无需估计参数，无需训练；
- 2) 适合对稀有事件进行分类；
- 3) 特别适合于多分类问题(multi-model, 对象具有多个类别标签)

缺点：

当样本不平衡时，如一个类的样本容量很大，而其他类样本容量很小时，有可能导致当输入一个新样本时，该样本的 K 个邻居中大容量类的样本占多数。

由于 KNN 方法主要靠周围有限的邻近的样本，而不是靠判别类域的方法来确定所属类别的，因此对于像本次这种类域的交叉或重叠较多的待分样本集来说，KNN 方法较为合适，实现也较为简单。

● One vs one 和 One vs rest

工作原理：

在 one vs rest 策略中，假设有 n 个类别，那么就会建立 n 个二项分类器，每个分类器针对其中一个类别和剩余类别进行分类。进行预测时，利用这 n 个二项分类器进行分类，得到数据属于当前类的概率，选择其中概率最大的一个类别作为最终的预测结果。

在 one-vs-one 策略中，同样假设有 n 个类别，则会针对两两类别建立二项分类器，得到 $k=n*(n-1)/2$ 个分类器。对新数据进行分类时，依次使用这 k 个分类器进行分类，每次分类相当于一次投票，分类结果是哪个就相当于对哪个类投了一票。在使用全部 k 个分类器进行分类后，相当于进行了 k 次投票，选择得票最多的那个类作为最终分类结果

特点:

One vs rest 这种方法有种缺陷, 因为训练集是 1:M, 这种情况下存在 biased. 因而不是很实用。

为了使训练集中各类的样本数目相同, 使用了 smote

One vs one

相对于 one vs rest, 这个方法的代价有些大, 需要 $n*(n-1)/2$ 个 model

现实表现 one vs one 好于 one vs rest

● 决策树 (Decision-Tree)

工作原理: 在这里采用的是 CART 树 (只构建二叉树)。决策树是基于树结构来进行决策的。树中每个节点表示某个对象, 而每个分叉路径则代表的某个可能的属性值, 而每个叶结点则对应从根节点到该叶节点所经历的路径所表示的对象的值。我们的目的是让每个节点的纯度越来越高。

在这里有两个重要公式。

熵: 信息的度量方式称为香农熵或者简称为熵。计算熵, 我们需要计算所有类别所有可能值包含的信息期望值, 通过下面的公式得到:

$$\text{Ent}(D) = -\sum_{i=1}^n p(x_i) \log p(x_i)$$

Ent(D) 的值越小, D 的纯度就越高。

信息增益: 在一个条件下, 信息复杂度 (不确定性) 减少的程度。决策树中每一个节点最优属性的选择也是根据这一特点来选择的。选择一个特征后, 信息增益最大 (信息不确定性减少的程度最大), 那么我们就选取这个特征作为该节点的划分标准。

信息增益公式: $\text{Gain}(D, a) = \text{Ent}(D) - \sum (|D^v|/|D|) \text{Ent}(D^v)$

特点:

优点: 计算复杂度不高, 输出结果易于理解, 对中间值的缺失不敏感, 可以处理不相关特征数据。

缺点: 可能会产生过度匹配问题。

训练过程:

如果是使用 sklearn 库的决策树生成的话, 剪枝方法有限, 仅仅只能改变其中参数来进行剪枝。

主要调参的部分

criterion: “gini” or “entropy” (default=“gini”) 是计算属性的 gini (基尼不纯度) 还是 entropy (信息增益), 来选择最合适的节点。

通过测试集训练, 采用 criterion='entropy', 效果会更好。

class_weight: 对于输入的样本, 平衡类别之间的权重。指定样本各类别的权重

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。
`class_weight="balanced"`

`min_samples_split`:是指当前节点允许分裂的最小样本数。当前节点样本数小于这个值时决策树终止。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。默认是 1, 可以输入最少的样本数的整数，或者最少样本数占样本总数的百分比。如果样本量不大，不需要管这个值。如果样本量数量级非常大，则推荐增大这个值。
在这里我设置 `min_samples_split=5`

3.2.2 集成分类器

● Adaboost

工作原理:

AdaBoost 方法是一种迭代算法，在每一轮中加入一个新的弱分类器，直到达到某个预定的足够小的错误率。每一个训练样本都被赋予一个权重，表明它被某个分类器选入训练集的概率。如果某个样本点已经被准确地分类，那么在构造下一个训练集中，它被选中的概率就被降低；相反，如果某个样本点没有被准确地分类，那么它的权重就得到提高。通过这样的方式，AdaBoost 方法能“聚焦于”那些较难分（更富信息）的样本上。在具体实现上，最初令每个样本的权重都相等，对于第 k 次迭代操作，我们就根据这些权重来选取样本点，进而训练分类器 C_k 。然后就根据这个分类器，来提高被它分错的的样本的权重，并降低被正确分类的样本权重。然后，权重更新过的样本集被用于训练下一个分类器 C_k [\[2\]](#)。整个训练过程如此迭代地进行下去。

特点:

adaBoost 是 boosting 方法中最流行的一种算法。它是以弱分类器作为基础分类器，输入数据之后，通过加权向量进行加权，；在每一轮的迭代过程中都会基于弱分类器的加权错误率，更新权重向量，从而进行下一次迭代。并且会在每一轮迭代中计算出该弱分类器的系数，该系数的大小将决定该弱分类器在最终预测分类中的重要程度。显然，这两点的结合是 adaBoost 算法的优势所在。

优点：泛化错误率低，容易实现，可以应用在大部分分类器上，无参数调整

缺点：对离散数据点敏感

训练过程:

调参操作：在训练集确定的情况下，调节 `learning_rate` ($0 < \text{learning_rate} < 1$) 每次增 0.001。找出最合适的 `learning_rate`。

● 随机森林 (Random Forest)

工作原理：随机森林是 Bagging 的一个扩展变体，RF 在以决策树为基学习器构建 Bagging 集成的基础上，进一步引入了随机属性选择。在每个节点的属性集合中随机选择一个包含 K 个属性的子集，再从中选择一个最优属性划分。

特点：

优点：1) 简单，容易实现，计算开销小

2) 随机森林与 Bagging 中基学习器的“多样性”仅通过样本扰动不同，其还增加了属性扰动，使得最终集成的泛化性能可通过个体学习器中差异度的增加而进一步提升。

3) 采用了随机取样，训练出模型的方差较小。

缺点：

1) 随机森林的起始性能往往相对较差，因为其添加了属性扰动。不过随着个体学习器树木的增加，随机森林会收敛到更低的泛化误差。

2) 在某些噪声较大的样本集，RF 模型容易陷入过拟合。

训练过程：

```
RandomForestClassifier(criterion='gini',bootstrap=True,random_state=15,oob_score=True,n_estimators=15)
```

采用 gini 作为衡量的标准，

n_estimators：建立的子树数目，调节发现 15 是本次实验最理想的值。

bootstrap：是统计学中的一种重采样技术，可以简单理解成是有放回地抽样，默认是 True，即采取有放回抽样这种策略。

oob_score=True：由于随机决策树生成过程采用的 Bootstrap，所以在一棵树的生成过程并不会使用所有的样本，未使用的样本就叫（Out_of_bag）包外样本，通过包外样本，可以评估这个树的泛化性能，辅助剪枝。默认为 False，这里将其设置为 True。

Bagging 泛化误差的包外估计为：

$$e^{oob} = \frac{1}{|D|} \sum_{(x,y) \in D} \text{sum}(H^{oob}(x) \neq y)$$

● 基于 Stacking 策略的集成分类器

原理：相对于投票法，“学习法”是一种更加强大的结合策略，将另一个学习器（次级学习器）与个体学习器（初级学习器）进行结合。次级学习器用于训练的数据叫做次级训练集，这个训练集是在初级学习器上学习得到的。这里用周志华老师《机器学习》中的流程来帮助理解一下该算法。

输入: 训练集 $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$;
初级学习算法 $\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_T$;
次级学习算法 \mathcal{L} .

过程:

```
1: for  $t = 1, 2, \dots, T$  do
2:    $h_t = \mathcal{L}_t(D)$ ;
3: end for
4:  $D' = \emptyset$ ;
5: for  $i = 1, 2, \dots, m$  do
6:   for  $t = 1, 2, \dots, T$  do
7:      $z_{it} = h_t(x_i)$ ;
8:   end for
9:    $D' = D' \cup ((z_{i1}, z_{i2}, \dots, z_{iT}), y_i)$ ;
10: end for
11:  $h' = \mathcal{L}(D')$ ;
输出:  $H(x) = h'(h_1(x), h_2(x), \dots, h_T(x))$ 
```

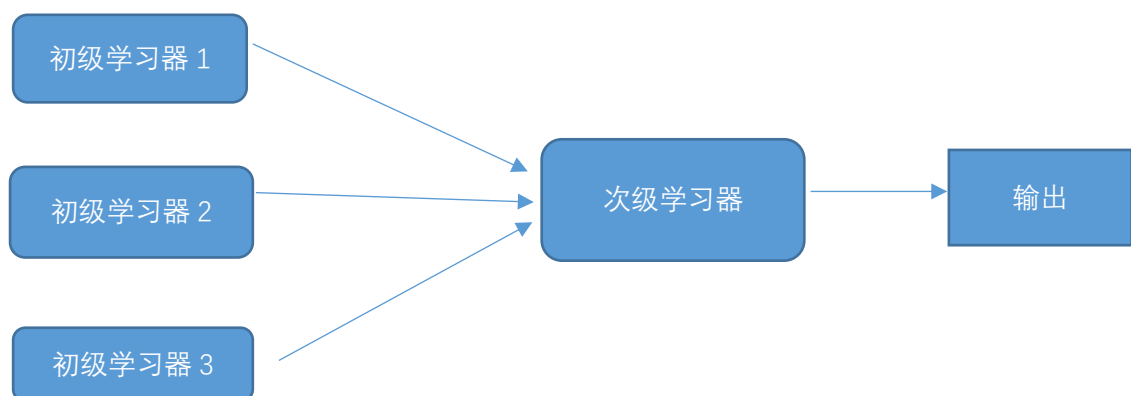
图 8.9 Stacking 算法

过程 1-3 是训练出来个体学习器，也就是初级学习器。

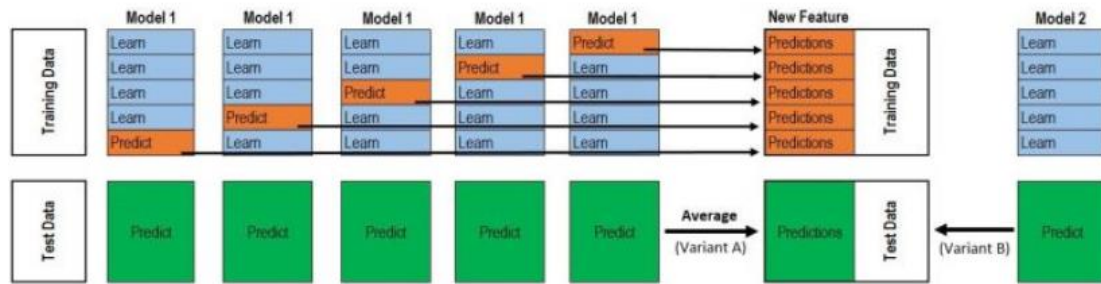
过程 5-9 是使用训练出来的个体学习器来得预测的结果，这个预测的结果当做次级学习器的训练集。

过程 11 是用初级学习器预测的结果训练出次级学习器，得到我们最后训练的模型。

如果想要预测一个数据的输出，只需要把这条数据用初级学习器预测，然后将预测后的结果用次级学习器预测便可。



这里采取交叉验证的思想实现 stacking 模型。



如上图，次级训练集的构成不是直接由模型在训练集 D 上面预测得到，而是使用交叉验证的方法，将训练集 D 分为 k 份，对于每一份，用剩余数据集训练模型，然后预测出这一份的结果。重复上面步骤，直到每一份都预测出来。这样就不会出现上面的过拟合这种情况。并且在构造次级训练集的过程当中，顺便把测试集的次级数据也给构造出来了。

对于我们所有的初级分类器，都要重复上面的步骤，才能构造出来最终的次级训练集和次级测试集。

特点：

优点：使用多个分类器集合起来得到一个强分类器，会提高学习模型的泛化性能。

缺点：在训练速度上比较其他简单模型会有所减慢。

训练过程：

初级分类器

我采用了随机森林 (RF)，自适应增强 (Adaptive Boosting)，梯度提升决策树，SVC 模型来做初级分类器。

```
rf_model=RandomForestClassifier(criterion='gini',bootstrap=True,random_state=15)
```

```
adb_model=AdaBoostClassifier(learning_rate=0.42)
```

```
gdbc_model=GradientBoostingClassifier()
```

```
et_model=ExtraTreesClassifier(criterion='gini',bootstrap=True)
```

```
svc_model= SVC()
```

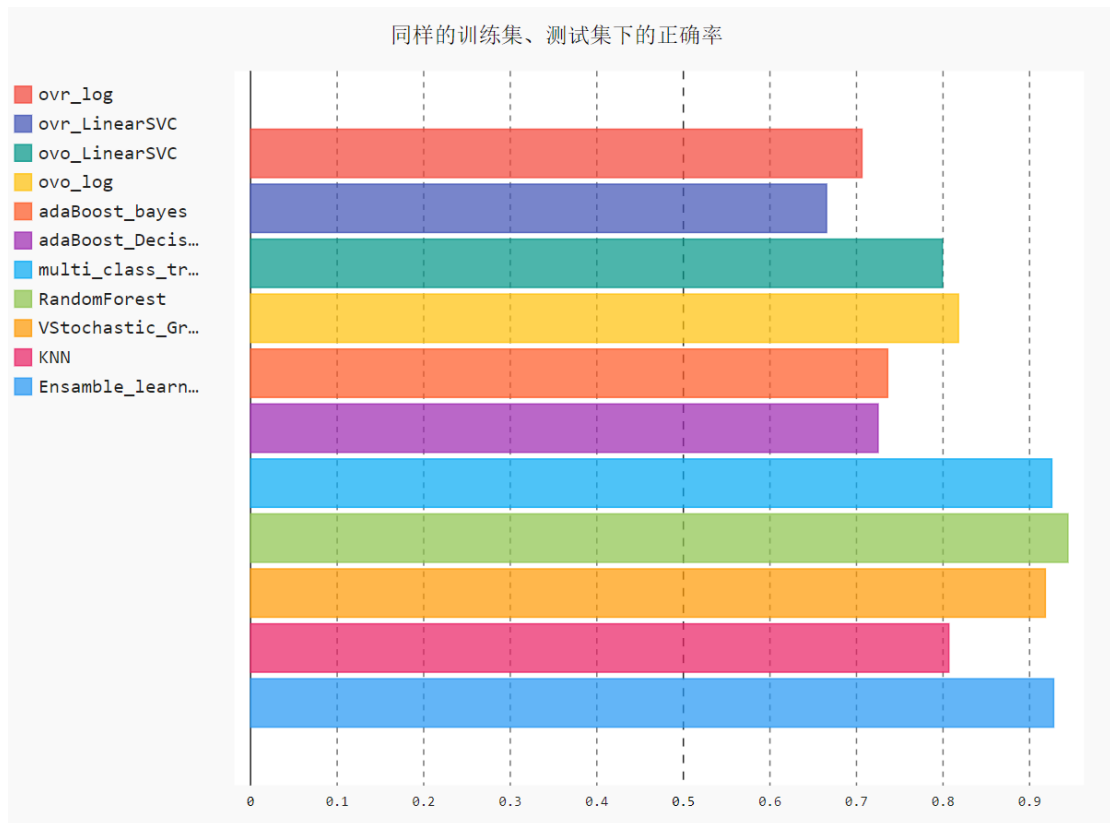
采用的交叉验证方法为十折验证法，通过初级分类器中每个基学习器迭代训练得到次级训练集和测试集。

次级分类器采用随机森林模型对次级数据集进行训练得到最后的分类标签。

3.2.3 分类器性能的比较和分析

以上是我们为解决该分类问题所设计的全部分类算法。我们在最后将其汇总，通过绘制正确率对比图以及混淆矩阵来分析比较各分类器的性能。

正确率：衡量模型性能的最直接标准



（该图在文件夹中保存为 correct_rate.svg 文件）

该图体现了各种分类器在相同数据集下的正确率表现。可以看出随机森林的正确率最高，为 94.4%。

最低的为 ovr_linearSVC, 为 66.5%

Stacking 策略集成得到的分类模型、VStochastic_Gra_boosting、决策树都有较高的正确率。

和其他算法相比具有较好的泛化性能。

我们的分类问题较适合通过基于决策树的分类器来进行实现。

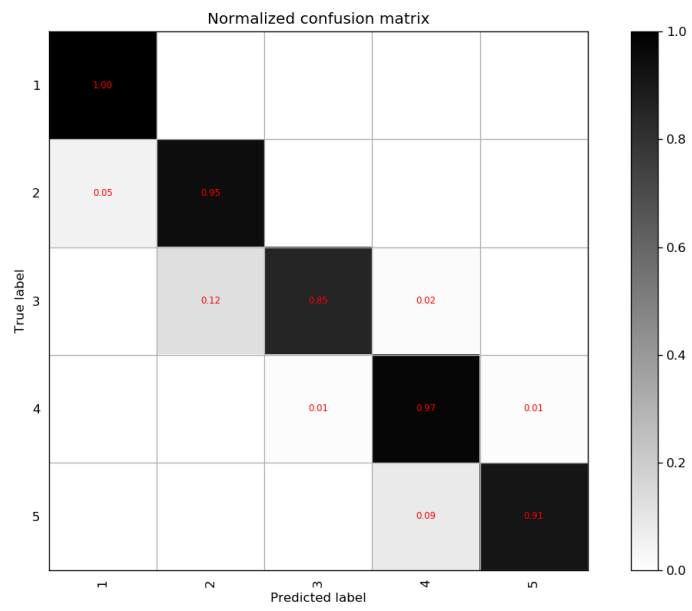
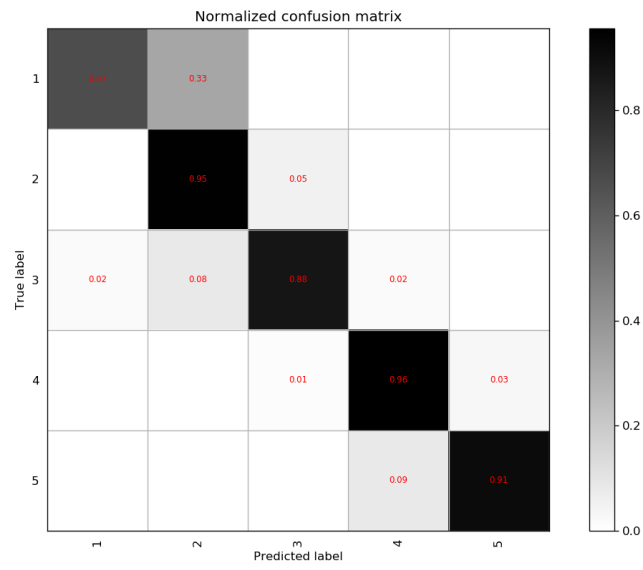
我们还绘制了混淆矩阵来得到各种分类器的分类情况。

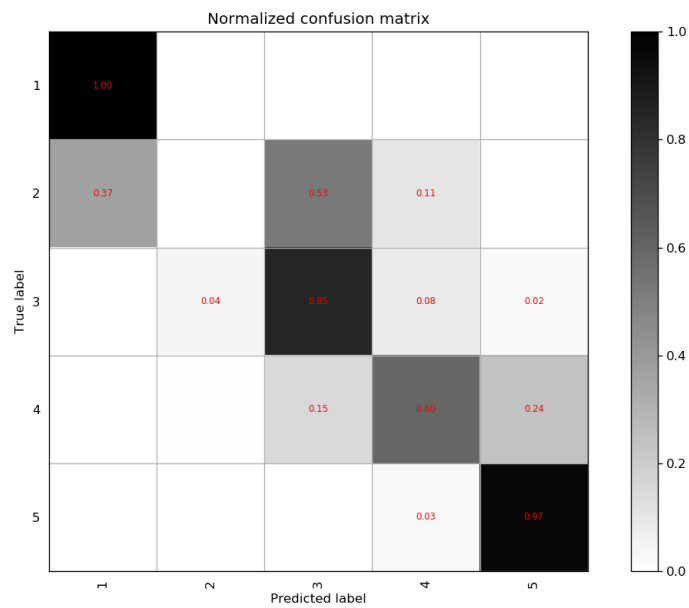
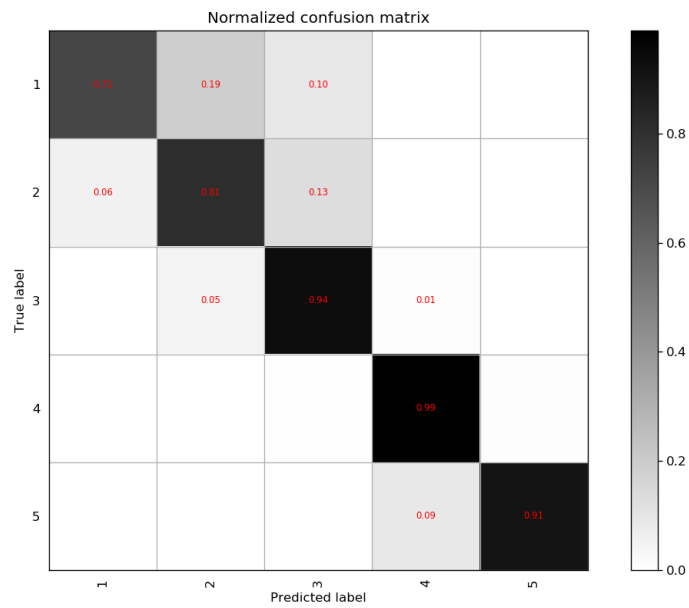
我们展示其中几个代表性的：

1. 决策树
2. 随机森林
3. 基于 Stacking 的分类模型
4. Ovr_linearSVC

可以发现较好的模型在每一类的准确率都相对较高。我们的分类数据对应的属性模型其实分布较为明显，所以各种学习器的效果都相对不错。其中基于决策树的个体和集合分类器效果最好。

（所有模型的混淆矩阵结果存在文件夹图片中）





4. 图形化界面（GUI 设计）

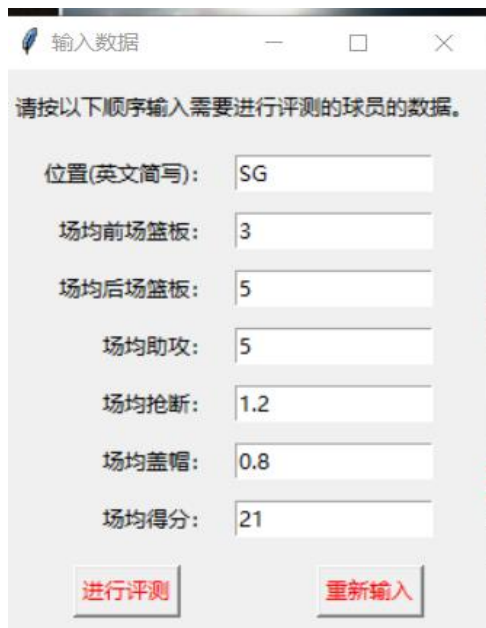
- 引入 Tkinter 与 PIL 模块，使用了其中的 label, button, image, entry 等组件以及网络布局管理器：grid 创建了该 GUI，通过 button 组件的 command 来调用函数实现页面的转换等各种操作。
- 引入 webbrowser 模块来实现 gui 中加入超链接进一步丰富界面功能的目的。

5. 结果展示

● 初始界面



● 球员测评界面

The image displays a '球员测评' (Player Evaluation) form within a window titled '输入数据' (Enter Data). The form instructs the user to '请按以下顺序输入需要进行评测的球员的数据。' (Please enter the data for the player to be evaluated in the following order). It contains seven input fields, each with a label and a value: '位置(英文简写):' (Position (English abbreviation)) with 'SG', '场均前场篮板:' (Average frontcourt rebounds) with '3', '场均后场篮板:' (Average backcourt rebounds) with '5', '场均助攻:' (Average assists) with '5', '场均抢断:' (Average steals) with '1.2', '场均盖帽:' (Average blocks) with '0.8', and '场均得分:' (Average points) with '21'. At the bottom of the form, there are two buttons: '进行评测' (Evaluate) and '重新输入' (Re-enter).

● 测评结果界面

球员属性

球员位置：

该球员综合能力：主力球员

前场篮板能力：B

后场篮板能力：C

得分后卫

进攻能力：B

抢断能力：B

助攻能力：B

防守能力：C

得分能力：B

盖帽能力：C

打法风格：

组织分位：哈登

突破精英：韦德

中距离：德罗赞

攻防全能：乔丹









撒盐撒步信手来
禁区内外任我行

一点浩然气
千里快哉风

北境之王赤胆忠心

十步杀一人
千里不留行

打法教学：

哈登视频

韦德视频

德罗赞视频

乔丹视频

● 点击打法教学链接到的视频：



6. 系统的不足

经过完善，系统仍存在几点不足之处如下：

- 1、最初的想法是在分完球员等级的情况下，进一步对每个等级每个位置的球员进行他们各自打球风格的再分类，但由于整体考虑欠妥当，造成了只分出了各种风格而没有和之前的等级进行联系的结果，最终只能放弃这一项进展；
- 2、由于采用的数据集较小，导致分类结果不是特别精确，相关结果只能作为大概参考；
- 3、结果的显示（GUI）对于 python 的环境依赖较大，没有形成一个网站或应用程序的独立显示，导致每次运行都得在代码环境下，可移植性比较差。继续发展我们希望把系统做成一个专门的网站。

7. 心得体会

- 艾俊涛：掌握基础、多看模型、好好理解模型原理、尝试着推推公式。
- 王哲：使用 python 创建 gui 界面感觉还是比较吃力，不够直观化，如 AS 在创建界面时可以直观的看到已经写好的界面或直接对界面进行添加功能，移位等操作，而 python 目前没有这个功能，比较僵硬。
- 徐钰东：本次是我们最重要的一次大作业，也是一次团队协作完成的作业，我们从前期的准备工作以及协作中付出了很多的时间，从一开始的毫无头绪到最后的确定从简单学习器到集成学习器的思路转换，我觉得算是本次实验中比较突出的要点。这次题目是我们比较感兴趣的话题。做起来可能也相对比较有热情。在查阅资料和学习的过程中，也扩展了很多自己的知识面。我主要负责的数据标签，也算巩固了一遍自己的 python 基础了。在决策树和随机森林，基于 Stacking 策略的集成学习器的设计中，对于提高自己对多分类问题的认识和理解有很多帮助。虽然这次系统在一些方面还存在着不足之处。如果时间充足，我相信我们能做得更好。机器学习还是有很多的实用处，希望自己能在以后的学习实践过程中将其付诸实际应用之中。
- 何恒朗：经过我们四人的分工协作，终于完成了第三次的实验，本次实验不同以往，从一开始实验的构思、实现的步骤、要用的算法都是未知数，到最后做出来具体的应用，这个过程学习和收获了很多：1、理论知识方面：本次实验涉及的分类型算法，在实验初期算法的挑选阶段，因为不知道各个算法对于本次实验的效果如何，所以对于各类算法都或多或少地学习了一些，在了解了它们的基本原理和优劣以及适用对象等方面的情况后，我选择了 KNN 和 K-MEANS 算法，但是用 K-MEANS 算法分类球员的风格这一工作以失败告终，心里有些遗憾；2、实验构思方面：一开始接到任务书时毫无头绪，想过很多方面的应用，最终还是以我们几个的共同爱好——篮球为切入点，先定了“分类球员”这一大的思路，然后具体化到球员的等级和风格，再逐渐细分最终“成型”，深刻体会到兴趣是最好的老师，确定的这个方向和我们的兴趣有关使得在实验中也多了一些动力和乐趣；3、分工合作方面：从开始的数据集都没有，到最终“球探”的产生，从无到有，从萌芽到一棵树的开枝散叶，深刻感受到了团队共同为一件事而努力的力量，无论结果如何，过程充实就好；4、个人方面：从之前对这门课的态度到现在的学习情况，感觉自己进步了不少，在 python 方面也有了较大的提升，但相比较周围同学来说，水平还是差了几个档，希望在以后的学习中能抓紧时间查缺补漏，争取迎头赶上。

附录：

- 主要参考文献《机器学习》周志华
《机器学习实战》Peter Harrington
《随机森林算法在空气质量评价中的应用》杭琦，杨敬辉，黄国荣
skit-learn 参考文档