数学建模期末报告

——对 NBA 常规赛战绩、总决赛表现以及选秀情况的试探性预测

2017 年 6 月 6 日

陈旭鹏（2014012882）

王复英（2015011175）

马聿伯（2015012295）

摘要

本文主要分为三个部分对 NBA 常规赛、总决赛成绩以及选秀情况作出试探性的预测。首先我们通过基本的静态数据，运用逻辑回归建立了一个比较简单的机器学习模型来预测 NBA 常规赛各支球队的战绩与排名。在该模型中我们对各支球队计算“等级分”，以此预测每场比赛两队的胜负概率，并最终给出我们的预测。该预测在一定程度上与本赛季的真实排名吻合，但也暴露出了一定的局限性。同时我们还用等级分模型预测了总决赛的胜负情况。第二部分我们对 NBA 选秀进行了分析：NBA 选秀是球队挑选 NBA 球员的重要过程，新秀的实力很大程度上决定球队在未来的实力。本文对传统的球员顺位预测方法进行了改进，希望以球员在 NBA 的表现为评价标准，建立一个更有实际意义的模型来根据球员大学数据对其在 NBA 表现做预测，使用了因子分析法和随机森林等方法。最后一部分我们做了一些微小的工作，探讨利用基于位置的大数据进行更精准预测的可能性。

关键词： 等级分模型，逻辑回归，选秀模型，因子分析法，随机森林，SportVu

目录

一.常规赛与总决赛的战绩分析预测

首先我们希望建立一个传统模型对本赛季 NBA 常规赛的战绩与排名以及最后的总决赛情况做出预测。尽管此时 NBA 常规赛已经落下帷幕，对其做出预测看似毫无意义，其实不然。由于我们使用的数据均来自上一个赛季的球队战绩，因此我们的模型没有受到本赛季球队表现的任何影响。更加重要的是，由于预测的事件已经发生，因此这个模型的精确度可以在客观世界中进行检验，我们也可以依据检验结果在接下来的时间里继续对其进行修正。

之所以称之为“传统模型”，主要有以下两点原因：一是该模型使用的数据类型比较基本，这些数据都是通过人工方式从各种数据分析网址上获取的；二是该模型的核心思想是对每个球队的“等级分”进行逻辑斯蒂回归，是一个比较基本的机器学习模型，已经有一套较为完善的算法体系和求解方法供我们参考使用。我们相信对它的熟悉与应用有助于接下来一系列更加复杂、更加精确的模型的建立与求解。

1.1 数据获取

我们采用了 NBA 数据统计专业网站 Basketball Reference.com 中的 5 组不同统计数据，简要介绍如下（每部分数据的具体含义见附录）：

（1）**Team Per Game Stats**:每支队伍平均每场比赛的表现统计。偏重于单一比赛中该支队伍的表现，例如：这场比赛的得分、篮板、助攻、命中率……

（2）**Opponent Per Game Stats:**所遇到的对手平均每场比赛的统计信息，所包含的统计数据与 **Team Per Game Stats** 中的一致，只是代表的该球队对应的对手的。

（3）**Miscellaneous Stats**：综合统计数据。偏重于反应这只球队的整体而非某一场表现特征，例如：球员名单、平均年龄、赛季命中率……

（4）**15-16result**：2015-2016 赛季每场比赛的胜负情况，以及 2015~2016 年的 nba 常规赛及季后赛的每场比赛的比赛数据。我们将以此作为训练集，通过机器学习来评估各支球队的实力，并将结果应用于新赛季的预测之中。

(5) **16-17Schedule**: 2016-2017 赛季的赛程安排

1.2 数据处理与模型假设

为了在我们力所能及的范围内进行模型建立与求解，我们决定做出如下假设：

（1）假设每只球队的实力水平在赛季进行过程中不发生任何变化

（2）假设每只球队面对不同球队都会发挥出相同的实力

（3）假设赛程安排对球队单一场次中的表现没有影响

简单对上述三条假设做出解释：第一条假设将每只球队的等级分变动止于 16—17 赛季开始之初，即只将 15—16 赛季的比赛结果作为训练集，而**不考虑 16—17 赛季（我们所模拟出来的）比赛结果**。这是因为：我们承认客观世界中 16—17 赛季的任何一场比赛结果对这个庞大的系统造成的“微小扰动”，都有可能引发“蝴蝶效应”，对最终预测产生难以估量的巨大影响。之所以不考虑之，一是我们水平有限，难以在短期内想到妥善的解决方案；二则（更重要的是）由于我们已经假设“每只球队面对不同球队都会发挥出相同的实力”，因此模拟出的某场比赛的胜负仅仅是随机概率，并不能告诉我们比之前更多的信息（例如球员的心理波动、球队的备战情况等等），没有纳入考虑的价值。同时第一条假设还排除了赛季中出现的**球员伤病**对球队实力的影响。

在第一条假设的基础上，第二条与第三条假设就显得相对自然。前者排除了**球员轮休**、**球风相克**、**球队状态波动**对球队实力的影响；后者则排除了背靠背、七连客等**极端赛程**对球队实力的影响。

1.3 数据分析

在获取到数据之后，我们将利用每支队伍过去的比赛情况和 Elo 等级分来判断每支比赛队伍的可胜概率。在评价到每支队伍过去的比赛情况时，我们将使用到 Team Per Game Stats，Opponent Per Game Stats 和 Miscellaneous Stats（之后简称为 T、O 和 M 表）这三个表格的数据，作为代表比赛中某支队伍的比赛特征。我们最终将实现针对每场比赛，预测比赛中哪支队伍最终将会获胜，但并不是给出绝对的胜败情况，而是预判胜利的队伍有多大的获胜概率。之后我们采用随机数模拟的方法获得每个球队的胜场数。

为了做出对比赛结果的较为合理的预测，我们需要建立一个代表比赛的特征向量。由两支队伍的以往比赛情况统计情况（T、O 和M表），和两个队伍各自的 Elo 等级分构成。

Elo 的最初为了提供国际象棋中，更好地对不同的选手进行等级划分。在现在很多的竞技运动或者游戏中都会采取 Elo 等级分制度对选手或玩家进行等级划分，如足球、篮球、棒球比赛或 LOL，DOTA 等游戏。

在这里我们将基于国际象棋比赛，大致地介绍下 Elo 等级划分制度。假设 A 和 B 的当前等级分为R_A和R_B，则 A 对 B 的胜率期望值为：

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}}$$

B 对 A 的胜率期望值为：

$$E_B = \frac{1}{1 + 10^{(R_A - R_B)/400}}$$

如果棋手 A 在比赛中的真实得分S_A（胜 1 分，和 0.5 分，负 0 分）和他的胜率期望值E_A不同，则他的等级分要根据以下公式进行调整：

$$R_A^{new} = R_A^{old} + K(S_A - R_A^{old})$$

在国际象棋中，根据等级分的不同 K 值也会做相应的调整：

- ≥2400，K=16
- 2100~2400 分，K=24
- ≤2100，K=32

因此我们将会用以表示某场比赛数据的特征向量为（对阵双方为 A 队和 B 队）：[A 队 Elo score, A 队的 T,O 和 M 表统计数据，B 队 Elo score, B 队的 T,O 和 M 表统计数据]

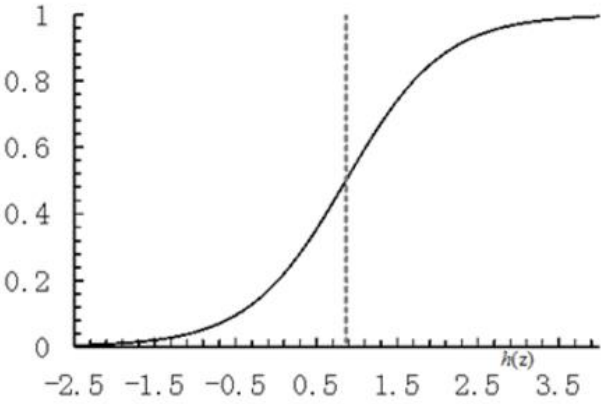
1.4 回归方法简介

综合考虑后我们采用逻辑回归模型进行预测，逻辑回归就是一种减小预测范围，将预测值限定为[0,1]间的一种回归模型.

(1)构造预测函数 h

逻辑回归实际上是一种分类方法，主要用于二分类问题（即输出只有两种，分别代表两个类别），所以利用了 Logistic 函数（或称为 Sigmoid 函数），函数表达式和曲线如下：

$$\mathrm{g(z)} = \frac{1}{1+e^{-z}}$$



逻辑曲线在 z=0 时，十分敏感，在 z>>0 或 z<<0 处，都不敏感，将预测值限定为(0,1)。

构造预测函数为：

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

其中x表示由若干个自变量所构成的向量，θ则为反映这些自变量对结果影响强弱的“系数向量”。 函数$h_\theta(x)$的值表示结果取 1 的概率，因此对于输入 x 分类结果为类别 1 和类别 0 的概率分别为：

$$\mathrm{P(y = 1|\theta)} = \frac{1}{1+e^{-\theta^T x}}$$

$$\mathrm{P(y = 0|\theta)} = \frac{e^{-\theta^T x}}{1 + e^{-\theta^T x}} = 1 - P(y = 1|\theta)$$

（2）构造损失函数 J

由最大似然估计得到如下 Cost 函数和 J 函数：

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & if\ y = 1 \\ -\log(1 - h_\theta(x)) & if\ y = 0 \end{cases}$$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{n} Cost(h_\theta(x_i), y_i) = -\frac{1}{m}\left[\sum_{i=1}^{n} y_i \log h_\theta(x_i) + (1 - y_i)\log(1 - h_\theta(x_i))\right]$$

y 向量为每个训练样本的标签，只有 0 和 1 这两种不同的取值，反映这场比赛的胜负情况。

（3）梯度下降法求的最小值

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x_i) - y_i \right) x_i^j$$

利用该方法可以求出θ的具体值。

（4）向量化

约定训练数据的矩阵形式如下，x 的每一行为一条训练样本，而每一列为不同的特征值。

$$x = \begin{bmatrix} x_1 \\ \dots \\ x_m \end{bmatrix} = \begin{bmatrix} x_{11} & \dots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix}, y = \begin{bmatrix} y_1 \\ \dots \\ y_m \end{bmatrix}, \theta = \begin{bmatrix} \theta_0 \\ \dots \\ \theta_n \end{bmatrix}$$

$$A = x \bullet \theta = \begin{bmatrix} x_{10} & \dots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix} \bullet \begin{bmatrix} \theta_0 \\ \dots \\ \theta_n \end{bmatrix} = \begin{bmatrix} \theta_0 x_{10} + \theta_1 x_{11} + \dots + \theta_0 x_{1n} \\ \dots \\ \theta_0 x_{m0} + \theta_1 x_{m1} + \dots + \theta_0 x_{mn} \end{bmatrix}$$

$$E = h_\theta(x) - \mathrm{y} = \begin{bmatrix} g(A_1) - y_1 \\ \dots \\ g(A_m) - y_m \end{bmatrix} = \begin{bmatrix} e_1 \\ \dots \\ e_m \end{bmatrix} = g(A) - y$$

因此θ的更新过程可表示为：

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x_i) - y_i \right) x_i^j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} e_i x_i^j = \theta_j - \alpha \frac{1}{m} x^T E$$

1.5 模型建立与求解

（1）设置初始 elo 值

以 2015—2016 赛季每场比赛的比赛结果作为训练集，并设置每支球队初始的 elo 为 1600。为了在一定程度上反映主客场对比赛走势的影响，我们将每场比赛中主队的 elo 值人为加上 100.

（2）初始化数据

从上文所提到的 **T、O 和 M** 表格中读入数据，去除一些无关数据，之后将这三个表格合并。这里的无关数据既有球馆名称等这类“绝对无关”的信息，也包含那些在之前模型假设中被我们舍弃的信息。我们将 elo 值作为特征向量的第一行，对每场比赛，我们都能给出它的一个特征向量，整合之后共有 66 个特征。建立 X 矩阵（1366x66），矩阵的每一行代表该赛季的某一场比赛，每一列代表该赛季所有比赛的某一特征值。

（3）根据比赛结果对 elo 进行修正

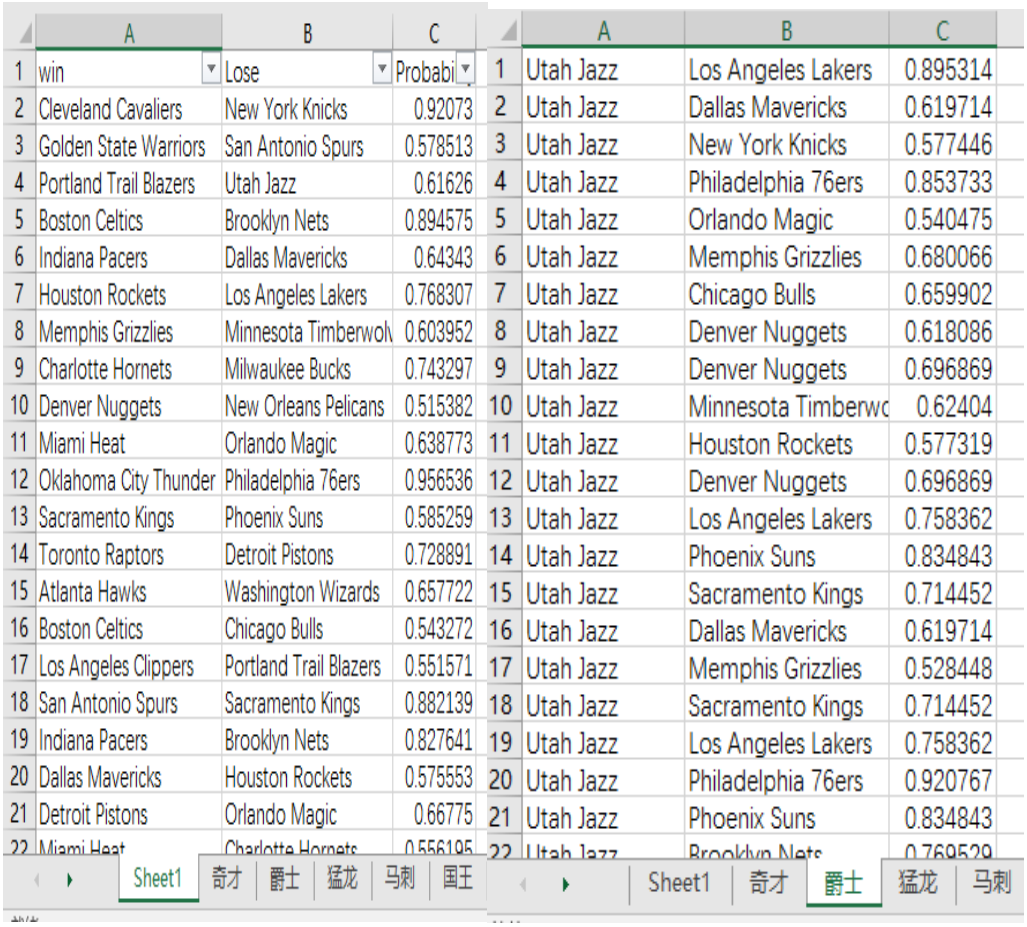
根据上赛季每一场比赛的比赛结果对各支球队的 elo 值不断进行修正，最终获得一个确定的 elo 值，将其作为不变量运用于新赛季的预测之中。

（4）逻辑回归

分别建立特征矩阵 X 和每个特征所属的类 y 以及“参数向量”θ，使用 sklearn 的 Logistic Regression 函数训练模型确定参数向量的具体值。在这里我们采用随机生成 0 或 1 的方法：如果生成的 y 为 0，X 的该行就为胜者特征在前，负者特征在后；如果 y=1，X 的该行就为负者特征在前，胜者特征在后。

（5）预测概率

根据 16-17 赛季的赛程和之前我们通过机器学习得到的“参数向量”、elo 等级分来计算新赛季任意一场比赛两支球队的获胜概率。（为了便于分析，我们整理得到了每支球队全赛季 82 场比赛胜利或失败的概率）



	A	B	C
1	win	Lose	Probabi
2	Cleveland Cavaliers	New York Knicks	0.92073
3	Golden State Warriors	San Antonio Spurs	0.578513
4	Portland Trail Blazers	Utah Jazz	0.61626
5	Boston Celtics	Brooklyn Nets	0.894575
6	Indiana Pacers	Dallas Mavericks	0.64343
7	Houston Rockets	Los Angeles Lakers	0.768307
8	Memphis Grizzlies	Minnesota Timberwol	0.603952
9	Charlotte Hornets	Milwaukee Bucks	0.743297
10	Denver Nuggets	New Orleans Pelicans	0.515382
11	Miami Heat	Orlando Magic	0.638773
12	Oklahoma City Thunder	Philadelphia 76ers	0.956536
13	Sacramento Kings	Phoenix Suns	0.585259
14	Toronto Raptors	Detroit Pistons	0.728891
15	Atlanta Hawks	Washington Wizards	0.657722
16	Boston Celtics	Chicago Bulls	0.543272
17	Los Angeles Clippers	Portland Trail Blazers	0.551571
18	San Antonio Spurs	Sacramento Kings	0.882139
19	Indiana Pacers	Brooklyn Nets	0.827641
20	Dallas Mavericks	Houston Rockets	0.575553
21	Detroit Pistons	Orlando Magic	0.66775
22	Miami Heat	Charlotte Hornets	0.556195

Sheet1 奇才 爵士 猛龙 马刺 国王

	A	B	C
1	Utah Jazz	Los Angeles Lakers	0.895314
2	Utah Jazz	Dallas Mavericks	0.619714
3	Utah Jazz	New York Knicks	0.577446
4	Utah Jazz	Philadelphia 76ers	0.853733
5	Utah Jazz	Orlando Magic	0.540475
6	Utah Jazz	Memphis Grizzlies	0.680066
7	Utah Jazz	Chicago Bulls	0.659902
8	Utah Jazz	Denver Nuggets	0.618086
9	Utah Jazz	Denver Nuggets	0.696869
10	Utah Jazz	Minnesota Timberwo	0.62404
11	Utah Jazz	Houston Rockets	0.577319
12	Utah Jazz	Denver Nuggets	0.696869
13	Utah Jazz	Los Angeles Lakers	0.758362
14	Utah Jazz	Phoenix Suns	0.834843
15	Utah Jazz	Sacramento Kings	0.714452
16	Utah Jazz	Dallas Mavericks	0.619714
17	Utah Jazz	Memphis Grizzlies	0.528448
18	Utah Jazz	Sacramento Kings	0.714452
19	Utah Jazz	Los Angeles Lakers	0.758362
20	Utah Jazz	Philadelphia 76ers	0.920767
21	Utah Jazz	Phoenix Suns	0.834843
22	Utah Jazz	Brooklyn Nets	0.769529

Sheet1 奇才 爵士 猛龙 马刺

（6）随机数模拟

我们利用 matlab 产生（0,1）随机数的方法模拟该比赛的结果。如果产生的随机数小于概率值，我们认为 elo 分较高的球队不出意外取得了胜利；如果产生的随机数小于概率值，我们认为 elo 分较低的球队“爆冷”击败了 elo 分较高的球队。

（7）常规赛战绩预测

统计每场比赛的胜负情况，依此排出我们所预估的常规赛战绩排名。为了适当规避一次取随机数的过大的偶然性，我们重复该过程 15 次，生成 15 次战绩排名，然后再取这 15 次的平均战绩作为我们最终的预测结果。

1	骑士	62	20
2	猛龙	58	24
3	老鹰	50	32
4	凯尔特人	50	32
5	黄蜂	49	33
6	热火	47	35
7	步行者	47	35
8	奇才	44	38
9	活塞	42	40
10	公牛	38	44
11	魔术	35	57
12	雄鹿	29	63
13	尼克斯	28	64
14	篮网	18	64
15	76人	10	72

东部球队战绩与排名（预测）

1	勇士	70	12
2	马刺	69	13
3	雷霆	61	21
4	快船	53	29
5	开拓者	50	32
6	小牛	43	39
7	火箭	43	39
8	爵士	42	40
9	灰熊	34	48
10	国王	31	51
11	掘金	31	51
12	森林狼	31	51
13	鹈鹕	27	55
14	太阳	22	60
15	湖人	15	67

西部球队战绩与排名（预测）

排名	球队	胜	负	胜率
1	凯尔特人[11]	53	29	64.6%
2	骑士[1]	51	31	62.2%
3	猛龙	51	31	62.2%
4	奇才[1]	49	33	59.8%
5	老鹰	43	39	52.4%
6	雄鹿	42	40	51.2%
7	步行者	42	40	51.2%
8	公牛	41	41	50%
9	热火	41	41	50%
10	活塞	37	45	45.1%
11	黄蜂	36	46	43.9%
12	尼克斯	31	51	37.8%
13	魔术	29	53	35.4%
14	76人	28	54	34.1%
15	篮网	20	62	24.4%

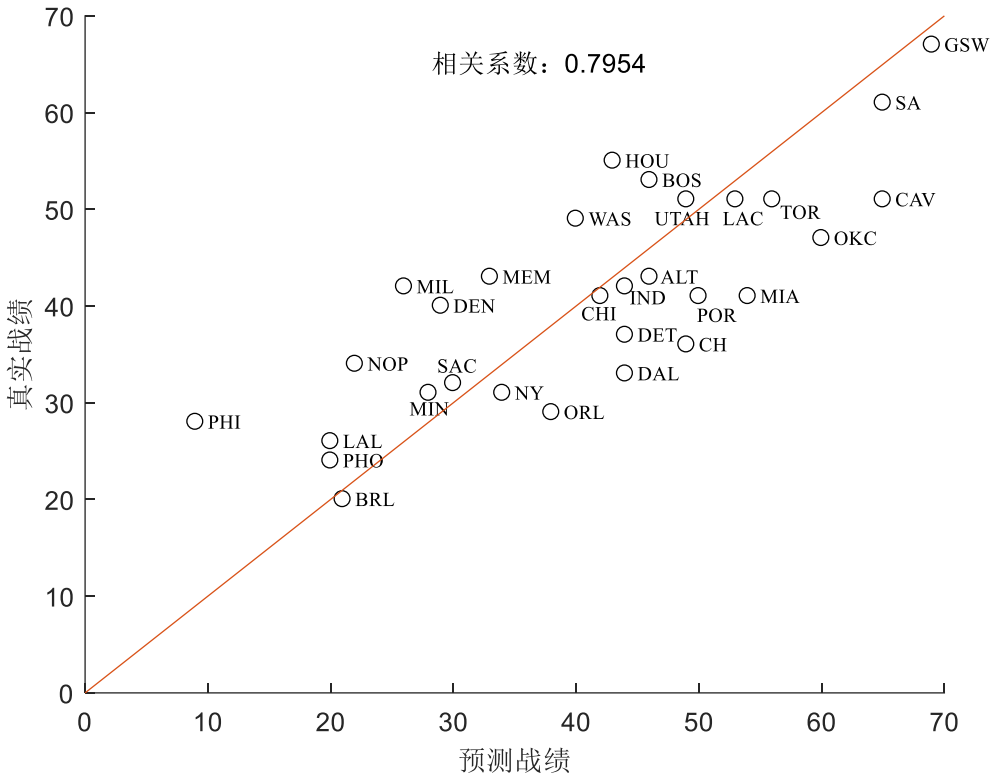
东部球队战绩与排名（真实）

排名	球队	胜	负	胜率
1	勇士[11]	67	15	81.7%
2	马刺[1]	61	21	74.4%
3	火箭	55	27	67.1%
4	快船	51	31	62.2%
5	爵士[1]	51	31	62.2%
6	雷霆	47	35	57.3%
7	灰熊	43	39	52.4%
8	开拓者	41	41	50%
9	掘金	40	42	48.8%
10	鹈鹕	34	48	41.5%
11	小牛	33	49	40.2%
12	国王	32	50	39%
13	森林狼	31	51	37.8%
14	湖人	26	56	31.7%
15	太阳	24	58	29.3%

西部球队战绩与排名（真实）

我们把模型生成的结果与本赛季常规赛的真实排名进行对比，放在同一个坐标系中做出散点图。如果预测结果与真实情况完全符合，则图中的点都将落在直线 y=x 上。我们用这两组数据的相关系数来评价预测的精确度。

预测战绩与真实战绩的散点图



通过比较我们发现，该模型在一定程度上可以反应本赛季的真实情况，但也不可否认其结果存在较大误差。这暴露出了我们所用传统模型的局限性，我们将会在下文中对这种偏差进行试探性的分析。

（8）总决赛战绩预测

运用与常规赛战绩预测完全相同的方法，我们按照 NBA 季后赛的赛制对十六支进入季后赛的球队再一次进行了模拟对阵。囿于篇幅限制，我们只展示总决

赛的预测结果。

根据我们的预测，最终进入总决赛的两支球队将会是勇士队与骑士队，且勇士队在 7 局 4 胜制的比赛中具有一个主场优势。经过 elo 等级分的比较，每场比赛对阵双方的胜负概率如下：

1	GoldenState Warriors	Cleveland Cavaliers	0.698143774	0.301856226
2	Golden State Warriors	Cleveland Cavaliers	0.698143774	0.301856226
3	Golden State Warriors	Cleveland Cavaliers	0.590908716	0.409091284
4	Golden State Warriors	Cleveland Cavaliers	0.590908716	0.409091284
5	Golden State Warriors	Cleveland Cavaliers	0.698143774	0.301856226
6	Golden State Warriors	Cleveland Cavaliers	0.590908716	0.409091284
7	Golden State Warriors	Cleveland Cavaliers	0.698143774	0.301856226

即在勇士队的主场，双方获胜概率大致为 7：3，而在骑士队的主场，双方获胜概率大致为 6：4。由此经过简单的概率计算，即可得出我们对总决赛的预测：

	Golden State Warriors	Cleveland Cavaliers
4	0.170188626	0.015248978
5	0.205489985	0.034589489
6	0.235258439	0.071892
7	0.144776411	0.122556073
sum	0.75571346	0.24428654

该表内的数值表示某一事件发生的概率，左侧的数字 n 代表着总决赛将在第 n 场随着某只球队的胜利而结束。我们发现，最终勇士队赢得总冠军的概率大约为 75%，而骑士队赢得总冠军的概率则大约为 25%。

1.6 局限性分析

经过初步分析，我们认为该模型的局限性主要来自以下几个方面：

Ⅰ 数据类型较为单一

正如之前提到过的，该模型虽然属于机器学习模型，但其所使用的数据主要以有限的静态数据为主。从 Basketball Reference.com 人工获得的数据主要存在两个问题，一是数据量不够丰富。该网站的数据信息对于一个业余 NBA 爱好者已经足够眼花缭乱，但对于需要海量数据的机器学习模型的建立而言显然仍略显单薄。二是数据同质化问题突出。尽管这些数据可以在很大程度上反映这场比赛的走势，但其同样有很多力不能逮的地方：比如“垃圾时间”的得分与“关键时刻”的得分价值远不能同日而语，但体现在这些数据上却是完全一样的。

Ⅱ 建模过程所舍弃的因素较多

正如本部分一开始所提到的，为了在我们的能力范围内做出预测，我们在模型的建立过程中做出了若干假设，忽略掉了许多因素。例如：球员的伤病是每支球队每个赛季都必须面对的问题；且不同球队之间确实存在球风相克的现象……这些细枝末节看似影响微弱，但它们引起的连锁反应的确可以极大改变本赛季的走势。此外，我们所计算的 elo 等级分截止于 15—16 赛季结束，因此休赛期各

支球队的人员流动也没有体现在我们的模型之中，这同样在一定程度上对我们模型的精确性产生了负面影响。

综上所述，我们在接下来的一段时间内将有针对性地对上文所提到的这些局限性进行改进与修正，并尝试建立更加精确的非传统模型，对当季 NBA 的一系列排名及荣誉归属进行试探性的预测。

二、对 NBA 选秀的结果预测

1.问题提出

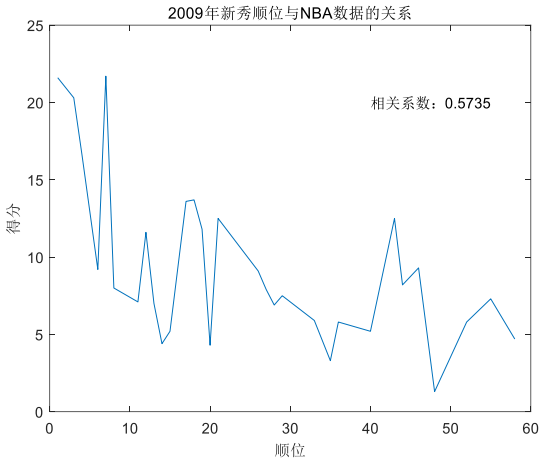
NBA 每年都会在 NBA 选秀大会上挑选新球员。球员被选中的顺序称为顺位。第一位被选中的称为状元，他也被认为是这一届球员中最优秀的。其后被选中的称为榜眼、探花。一般情况下，一个球员越被靠前选中，说明他的实力越强。

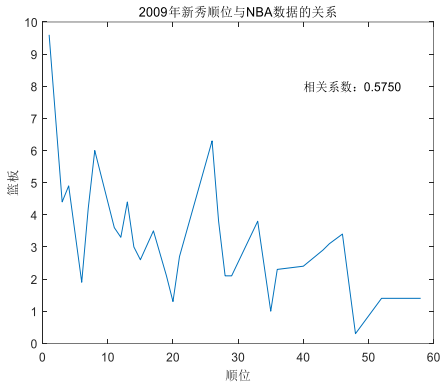
但是，我们发现了一种普遍的现象，联盟中一些超级巨星往往也是低顺位出身，但是一些高顺位的状元、榜眼、探花等人却在 NBA 销声匿迹。

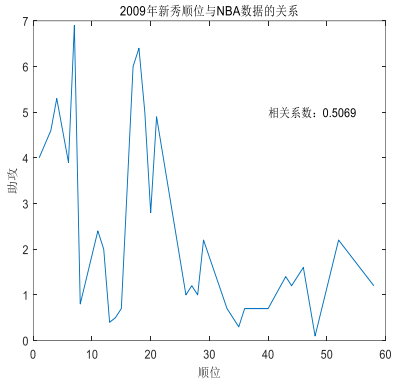
MVP排行榜	球员	顺位
1	威斯布鲁克	4
2	哈登	3
3	詹姆斯	1
4	伦纳德	15
5	小托马斯	60
6	库里	7
7	沃尔	1
8	德罗赞	9
9	利拉德	6
10	海沃德	9

根据上表我们可以看出，2016-2017 赛季 NBA 常规赛 MVP 排行榜前十名出现了好几个“低顺位”，如小托马斯、伦纳德，相对较低的还有德罗赞和海沃德。而 2013 年状元秀安东尼·本内特、2007 年状元秀格雷格·奥登、2001 年状元秀夸梅布朗等人在 NBA 表现甚至远远低于平均球员水准，被称为“水货状元”。

下面三幅图显示了 2009 年新秀的顺位与其在 NBA 得分、篮板、助攻的折线图。







由此我们提出疑问：能否根据球员的大学数据来预测他在 NBA 的表现？这两者之间的相关性到底怎么样？这也是我们接下来的模型需要解决的问题。

2.模型建立

（1）问题分析

首先，评价一名 NBA 球员，其得分是非常重要的一项指标。但是，一个球员的价值又不能全靠得分来衡量。对于中锋和大前锋，他们在球场上的防守贡献值往往大于进攻贡献值，这可以通过篮板、抢断、盖帽等数据体现。对于控球后卫，他们的作用主要体现在组织球队的进攻上，助攻也体现了他们对球队的贡献。因此我们希望建立一个指标来客观的评估 NBA 球员的综合实力。如果我们获得了一定的数据集，就可以通过机器学习的回归算法来预测球员的 NBA 得分和综

合评分。

另外，如果我们获得了球员的综合实力评分，我们就可以以此为依据对这一届新秀根据其在 NBA 的表现做一个重排。那么就可以分析球员的大学数据和球员在该届新秀中的实力排名之间的关系。对这个问题，我们可以采用机器学习中的分类算法来预测。

（2）NBA 球员综合实力的评估

①主成分分析法（PCA）

主成分分析法利用降维的思想，把多个指标转化为少数几个综合指标（主成分），其中每个主成分都能够反映原始变量的大部分信息，且所含信息互不重复。

主成分分析法本质上是一种数学变换的方法，它把给定的一组相关变量通过线性变换转成另一组不相关的变量，这些新的变量按照方差依次递减的顺序排列。在数学变换中保持变量的总方差不变，使第一变量具有最大的方差，称为第一主成分，第二变量的方差次大，并且和第一变量不相关，称为第二主成分。依次类推，I 个变量就有 I 个主成分。

主成分分析的基本原理就是将一个矩阵中的样本数据投影到一个新的空间中去。对于一个矩阵来说，将其对角化即产生特征根及特征向量的过程，也是将其在标准正交基上投影的过程，而特征值对应的即为该特征向量方向上的投影长度，因此该方向上携带的原有数据的信息越多。

主成分分析法的分析步骤如下：

a.将原始数据按行排列组成矩阵 X

b.对 X 进行数据标准化，使其均值变为零

c.求 X 的协方差矩阵 C

d.将特征向量按特征值由大到小排列，取前 k 个按行组成矩阵 P

f.通过计算 Y = PX，得到降维后数据 Y

g.用下式计算每个特征根的贡献率 Vi;

Vi=xi/(x1+x2+.....)

②因子分析法

因子分析常采用以主成分分析为基础的反覆法。当根据主成分分析，决定保留 l 个主成分之后，接着求 l 个特征向量的行平方和来代替相关数矩阵对角线之值，形成约相关矩阵。根据约相关系数矩阵，可进一步通过反复求特征值和特征向量方法确定因子数目和因子的系数。

为了确定因子的实际内容,还须进一步旋转因子，使每一个变量尽量只负荷于一个因子之上。这就是简单的结构准则。常用的旋转有直角旋转法和斜角旋转法。作直角旋转时，各因素仍保持相对独立。在作斜角旋转时，允许因素间存在一定关系。

③评估球员综合实力

根据特征值大于 1 的原则, 我们选择 3 个因子, 累积方差贡献率为 80.31%。把主成分表示成各变量的线性组合：

	内线因子	组织因子	得分因子
效率值	0.25587	0.40954	0.84534
投篮	0.69197	-0.36194	0.1093
前场	0.88466	0.08559	-0.11975
后场	0.93666	0.10568	0.21955
助攻	-0.48157	0.80016	0.09087
抢断	-0.19566	0.66198	0.32758
盖帽	0.83387	-0.18301	-0.06956
失误	-0.07498	-0.81782	-0.3771
犯规	-0.65413	-0.59876	0.22692
得分	-0.13808	0.12193	0.91251

第一个因子 F1 主要由投篮命中率，进攻篮板、防守篮板、盖帽、犯规五个指标决定，我们命名其为内线因子。

第二个主因子 F2 主要由助攻、抢断、事失误三个指标决定，我们命名其为组织因子。

第三个 F3 主要由效率值、得分两个指标决定，我们命名其为得分因子。

	Vars	Vars.Prop	Vars.Cum
内线因子	3.624	36.24	36.24
组织因子	2.471	24.71	60.95
得分因子	1.936	19.36	80.31

（Var：方差贡献率；Var.Prop：方差贡献率在这三个因子中的占比； Var.Cum：累计方差贡献率）

以各因子的方差贡献率占三个因子总方差贡献率的比重作为权重进行加权，得出每个球员的综合得分：

$$F=(36.24*F1+24.71*F2+19.36*F3)/80.31$$

（3）数据处理

我们获得了 2000-2016 年参加 NBA 选秀的球员的大学时期的数据和他在 NBA 时期的数据。

首先根据评估球员综合实力的算法计算每个球员的综合得分，并对球员的综合实力进行了重排。

pick_overall	real_rank	team_id	player	college_name	pts_per_g	trb_per_g	ast_per_g
1	2	LAC	Blake Griffin	University of Oklahoma	21.6	9.6	4
3	3	OKC	James Harden	Arizona State University	20.3	4.4	4.6
4	4	SAC	Tyreke Evans	University of Memphis	16.7	4.9	5.3
6	32	MIN	Jonny Flynn	Syracuse University	9.2	1.9	3.9
7	1	GSW	Stephen Curry	Davidson College	21.7	4.2	6.9
8	20	NYK	Jordan Hill	University of Arizona	8	6	0.8
11	30	NJN	Terrence Williams	University of Louisville	7.1	3.6	2.4
12	21	CHA	Gerald Henderson	Duke University	11.6	3.3	2
13	29	IND	Tyler Hansbrough	University of North Carolina	7	4.4	0.4
14	45	PHO	Earl Clark	University of Louisville	4.4	3	0.5
15	39	DET	Austin Daye	Gonzaga University	5.2	2.6	0.7
17	7	PHI	Jrue Holiday	University of California, Los Angeles	13.6	3.5	6
18	6	MIN	Ty Lawson	University of North Carolina	13.7	2.8	6.4
19	13	ATL	Jeff Teague	Wake Forest University	11.8	2.1	5
20	48	UTA	Eric Maynor	Virginia Commonwealth University	4.3	1.3	2.8

（2009 年部分新秀的 NBA 数据与重排）

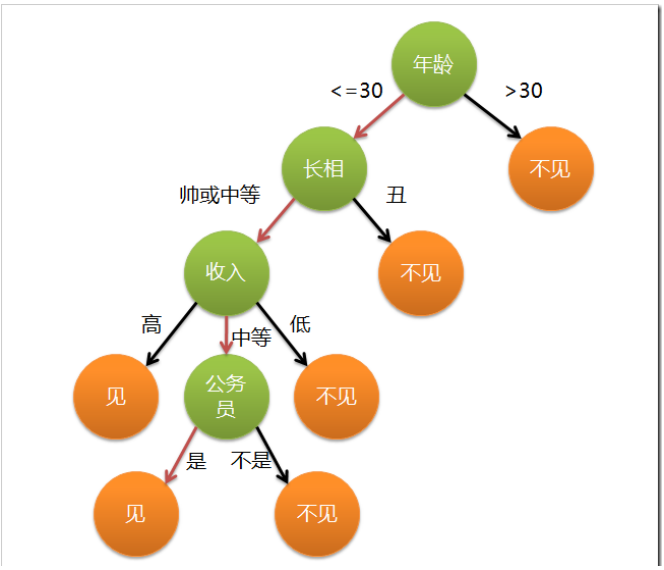
接着我们用 MATLAB 分析了球员的综合实力评分与大学中各项数据和体测部分数据的相关系数，删除了以下相关系数非常小的变量。

另外一些球员因为在海外打球或者没有参加部分体侧，存在部分数据缺失现象。为了解决这个问题，我们采用这一行数据的平均值来填充这些空缺位置，近似代替这些球员的水平。

（4）决策树

决策树本质上是一个树结构（可以是二叉树或非二叉树），其每个非叶子节点表示一个特征属性上的测试，每个分支代表这个特征属性在某个值域上的输出，而每个叶节点存放一个类别。

使用决策树进行决策的过程就是从根节点开始，测试待分类项中相应的特征属性，并按照其值选择输出分支，直到到达叶子节点，将叶子节点存放的类别作为决策结果。



上图完整表达了这个女孩决定是否见一个约会对象的策略，其中绿色节点表示判断条件，橙色节点表示决策结果，箭头表示在一个判断条件在不同情况下的决策路径，图中红色箭头表示了上面例子中女孩的决策过程。

（5）随机森林

随机森林顾名思义，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。

随机森林的随机主要体现在对训练数据集的行采样和列采样随机。数据集的每一行表示一个样本，对于行采样，采用有放回的方式，也就是在采样得到的样本集合中，可能有重复的样本。假设输入样本为 N 个，那么采样的样本也为 N 个。这样使得在训练的时候，每一棵树的输入样本都不是全部的样本，使得相对不容易出现过拟合。然后进行列采样，从 M 个特征中，选择 m 个(m << M)。

之后随机森林需要对采样到的数据使用完全分裂的方式建立出决策树，这样决策树的某一个叶子节点要么是无法继续分裂的，要么里面的所有样本的都是指向的同一个分类。

3.模型求解

（1）用 Excel 完场数据初步处理，填充缺失值，根据公式计算球员综合得分并且对球员进行重排；

（2）利用 MATLAB 中的 corrcoef 函数计算每个特征与综合得分的相关系数，对特征做初步筛选；

（3）利用 Python 的 sklearn 库中的 RandomForest 进行模型训练与预测。（源代码见附录）

（4）将预测结果导入 excel 中，因为数据过少，通过平均绝对误差来评价模型的准确性。

4.结果分析与评价

（1）对实力排名的预测

				NBA数据			球员信息	
选秀顺位	实力排名	预测实力排名	综合得分	场均得分	场均篮板	场均助攻	球员	大学
21	10	32	6.486033	12.5	2.7	4.9	Darren Collison	UCLA
26	12	20	6.088253	9.1	6.3	1	Taj Gibson	USC
27	15	21	5.270396	7.9	3.8	1.2	DeMarre Carroll	Missouri
28	41	7	1.690497	6.9	2.1	1	Wayne Ellington	North Carolina
29	26	21	3.403987	7.5	2.1	2.2	Toney Douglas	Florida State

上表示对球员在 NBA 实力排名的预测。从上表可看到误差相当大，平均绝对误差为 15。

（2）对场均得分的预测

NBA数据			球员数据			预测数据
场均得分	场均篮板	场均助攻	球员	大学	大学得分	预测得分
9.1	2.6	4.9	Greivis Vasquez	Maryland	21.9	8.98115
2.8	2.5	0.5	Daniel Orton	Kentucky	9.4	7.3756
2.9	1.3	0.5	Lazar Hayward	Marquette	23.5	7.87475
2.3	1.8	0.2	Dexter Pittman	Texas	19.8	10.15265
3.3	1	1.2	Armon Johnson	Nevada	17	8.45415
1.6	0.2	0.6	Andy Rautins	Syracuse	14.5	9.0538
6.8	4.3	1.6	Landry Fields	Stanford	24	7.82175
8.4	4.2	3.1	Lance Stephenson	Cincinnati	17.4	8.87065
3.6	1.9	0.4	Devin Ebanks	West Virginia	14.5	7.4681
0	0	0	Gani Lawal	Georgia Tech	20.2	7.80945
3.6	2.8	0.5	Luke Harangody	Notre Dame	27.3	9.53815
1.9	0.6	1.4	Willie Warren	Oklahoma	20.2	8.67915
2	1	0.2	Derrick Caracter	UTEP	20.1	8.73845

上表是对球员在 NBA 场均得分的预测。平均绝对误差为 4.328376，从表格中直观看出，预测结果与真实结果相近的仅仅是个别球员（第 1、7、8 个球员），对大部分球员还是有较大偏差。

（3）对综合实力的预测

					NBA数据			球员信息		
选秀顺位	实力排名	预测综合实力	综合实力	预测得分	场均得分	场均篮板	场均助攻	球员	大学	大学得分
38	51	4.66889813	-2.65428	9.0538	1.6	0.2	0.6	Andy Rautins	Syracuse	14.5
39	17	3.79900615	4.339142	7.82175	6.8	4.3	1.6	Landry Fields	Stanford	24
40	11	4.04524	5.173017	8.87065	8.4	4.2	3.1	Lance Stephenso	Cincinnati	17.4
43	37	4.08486	0.721798	7.4681	3.6	1.9	0.4	Devin Ebanks	West Virgini	14.5
46	49	5.109732	-1.00037	7.80945	0	0	0	Gani Lawal	Georgia Tech	20.2
52	31	5.705669	1.443442	9.53815	3.6	2.8	0.5	Luke Harangody	Notre Dame	27.3
54	41	4.999	0.211784	8.67915	1.9	0.6	1.4	Willie Warren	Oklahoma	20.2
58	44	5.54455964	-0.04342	8.73845	2	1	0.2	Derrick Caracte	UTEP	20.1

上表是对球员综合实力的预测。平均绝对误差为 4.1377。可以看出，预测偏差仍然较大。

（4）结果分析

通过对以上三组预测结果的整理，我们发现该结果的预测结果存在相当大的误差。其原因主要与以下两点：

①.通过直观观察数据球员的大学数据和 NBA 数据，我们发现球员的大学数据与其在 NBA 的表现关系不大

②.球员在 NBA 的表现受到多方面制约，比如身体天赋，这些是数据上体现不出来的。

③数据量过小。每年参加选秀的球员共有 60 个，这也就限制了我们的数据量。不同年的评价标准，如果把多年的球员数据放在一起训练我们发现其误差更大。很大程度上是因为不同年选秀的评价标准不同。

5.模型优缺点评价

（1）优点：

①相对于传统的选秀模型直接把选秀顺位作为标签进行预测的方法，我们希望能找到更客观的标签，来正确评估球员在 NBA 的表现。进而采用根据其实力评分重排的方法。

②相对于其他的机器学习算法，随机森林简单、容易实现、计算开销小，令人惊

奇的是，它在很多现实任务中展现出强大的性能，被誉为“代表集成学习技术水平的方法”。

③利用主成分因子分析法对球员实力进行综合的评估能客观地衡量球员的实力。

（2）缺点：

①很多球员的大学表现和 NBA 表现相差很大，规律性不强；

②数据量过少，每年参加选秀的球员只有 60 个，无法获得较强规律；

③ 考虑因素不足，我们除了球员的大学数据，还需要考虑球员的各项体测数据，球员的上升空间、技术特点等。

三、更多的尝试

在做完前两部分的内容后，我们又把目光聚焦于如何提升我们的模型表现，让我们的分析预测更加准确，此部分内容并未完成，一是收集与处理数据花了很多时间，二是学习深度学习算法本身花了很多时间。我们认为这是利用大数据来分析比赛的未来的方向，我们希望如果以后有机会的话，可以继续完成此部分的内容。

在这部分中我们更详细地考虑了具体的比赛中的因素，分析具体的比赛。我们也通过收集大数据来分析比赛，实现了比赛的动画模拟，以及使用 two stream CNN 分析战术、给出评估。为了更好的利用神经网络分析动画，我们还需要针对 NBA 比赛提取更多的特征。

3.1 为什么我们需要更多的尝试

数据不能体现场上的表现，更多的细节会提供更精确的分析结果，以及大数据和深度学习技术为我们的想法提供了实现的可行性。

Team-by-team forecast

Forecast from Today

ELO	CARM-ELO	1-WEEK CHANGE	TEAM	CONF.	CHANCE OF MAKING ... CONF. SEMIS	CONF. FINALS	FINALS	WIN TITLE
1823	1843	+1	Warriors 8-0	West	✓	✓	91%	84%
1682	1681	+37	Spurs 8-5	West	✓	✓	9%	7%
1621	1618	+0	Cavaliers 8-0	East	✓	✓	48%	5%
1612	1613	+16	Celtics 8-5	East	✓	✓	52%	4%

从这张表格可以看到，对骑士和凯尔特人谁能进入总决赛，ESPN 和我们的 ELO 都给出了和事实结果非常不同的预测，数据可能体现不出来战术的比拼，常规赛的留力、轮休对 ELO 的影响，球员间的对位优势，强势球员对球队的作用等。

3.2 利用爬虫进行更大规模数据的收集

为了能够超越传统预测模型的局限，我们需要更多更具体的数据来体现球队水平、球员竞技状态等情况。除了在传统模型中收集的 BBR 网站上的基础数据，我们需要更多的数据支持我们分析。

我们锁定了 NBA 数据统计专业网站 stats.nba.com。该网站数据更为齐全，适合我们进行进一步的数据挖掘与补充。

基于 python 的爬虫是挖掘数据的强大的工具，可是由于 NBA 的数据纷繁复杂，种类极其多，为每个网页编写程序进行爬取较为费时费力，因此我们需要寻找更为合适的方法。

首先我们寻找到了一个提供 NBA 数据分析 API 的网站 probasketballapi.com。该网站提供七天免费试用的 API，可以爬取多种数据。包括：Basic Resources: Teams, Players, Games, Shot Charts; Advanced Stats: Advanced Team Stats, Advanced Player Stats; Box Score: Team Boxscores, Player Boxscore; Four Factors: Team Four Factors, Player Four Factors; Misc Stats: Team Misc Stats, Player Misc;

Stats; SportsVU: Team SportsVU Data, Player SportsVU Data; Player Usage: Player Usage Data. 注意此 API 提供的 SportVU 数据也和其他数据一样，属于纯数据类型。该网站提供了基于 PHP 的接口，为了统一处理，我们将代码改写为 python。

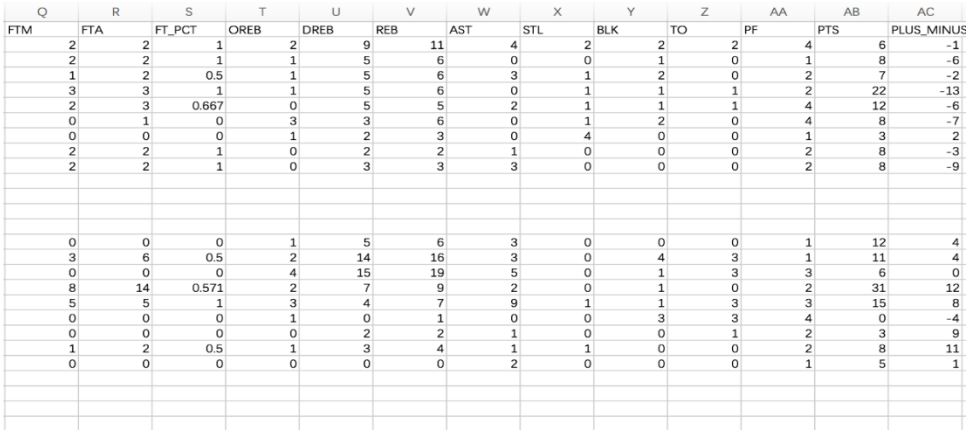
这样，只用改写 url 以及不同的 query_string 即可调取不同的所需要的数据，并将其写入 csv 中方便之后的调用。接下来我们又找到了 github 中提供的 stats.nba.com 的'官方'接口 nba_py，发现该接口提供的数据类型更为丰富具体，且调取更加方便。该接口共提供了数十种不同的数据，不仅包括赛季的平均数据，还包括每场比赛的具体数据，给接下来的数据挖掘与建模提出了巨大的挑战。

下面举一个简单的例子展示如何使用 nba_py 进行数据调取

```
from nba_py import game
r=game.Boxscore('0041400122').player_stats()
r.to_csv('/Users/james/Desktop/2.csv')
```

首先我们载入 game 模块，然后我们选择调取某场比赛的 Boxscore 数据中关于运动员的数据。0041400122 为该场比赛的 GameID。如果我们查看返回的数据类型会发现是 list，下面只需要简单的命令即可写入 csv 中，我们打开 csv 文件发现生成了这样一个表格：

B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
GAME_ID	TEAM_ID	TEAM_ABBREV	TEAM_CITY	PLAYER_ID	PLAYER_NAME	START_POSI	COMMENT	MIN	FGM	FGA	FG_PCT	FG3M	FG3A	FG3_PCT	FTM
41400122	1610612749	MIL	Milwaukee	203507	Giannis Antetoki	F		37:52:00	2	11	0.182	0	0	0	2
41400122	1610612749	MIL	Milwaukee	101141	Ersan Ilyasova	F		23:44	3	10	0.3	0	3	0	2
41400122	1610612749	MIL	Milwaukee	2585	Zaza Pachulia	C		22:32	3	8	0.375	0	0	0	1
41400122	1610612749	MIL	Milwaukee	203114	Khris Middleton	G		38:19:00	8	20	0.4	3	7	0.429	3
41400122	1610612749	MIL	Milwaukee	203487	Michael Carter-\	G		32:51:00	5	12	0.417	0	1	0	2
41400122	1610612749	MIL	Milwaukee	203089	John Henson			25:28:00	4	9	0.444	0	1	0	0
41400122	1610612749	MIL	Milwaukee	201162	Jared Dudley			17:02	1	4	0.25	1	2	0.5	0
41400122	1610612749	MIL	Milwaukee	201573	Jerryd Bayless			15:09	3	6	0.5	0	0	0	2
41400122	1610612749	MIL	Milwaukee	201564	O.J. Mayo			27:03:00	3	10	0.3	0	3	0	2
41400122	1610612749	MIL	Milwaukee	203898	Tyler Ennis		DNP - Coach's Decision								
41400122	1610612749	MIL	Milwaukee	203268	Jorge Gutierrez		DNP - Coach's Decision								
41400122	1610612749	MIL	Milwaukee	203948	Johnny O'Bryant III		DNP - Coach's Decision								
41400122	1610612749	MIL	Milwaukee	203101	Miles Plumlee		DNP - Coach's Decision								
41400122	1610612741	CHI	Chicago	2399	Mike Dunleavy	F		33:24:00	4	12	0.333	4	9	0.444	0
41400122	1610612741	CHI	Chicago	2200	Pau Gasol	F		33:35:00	4	12	0.333	0	1	0	3
41400122	1610612741	CHI	Chicago	201149	Joakim Noah	C		32:19:00	3	9	0.333	0	0	0	0
41400122	1610612741	CHI	Chicago	202710	Jimmy Butler	G		46:00:00	10	19	0.526	3	9	0.333	8
41400122	1610612741	CHI	Chicago	201565	Derrick Rose	G		38:05:00	4	14	0.286	2	6	0.333	5
41400122	1610612741	CHI	Chicago	201959	Taj Gibson			11:00	0	0	0	0	0	0	0
41400122	1610612741	CHI	Chicago	203503	Tony Snell			13:21	1	3	0.333	1	3	0.333	0
41400122	1610612741	CHI	Chicago	202703	Nikola Mirotic			22:21	3	9	0.333	1	3	0.333	1
41400122	1610612741	CHI	Chicago	201166	Aaron Brooks			9:55	2	3	0.667	1	2	0.5	0
41400122	1610612741	CHI	Chicago	203946	Cameron Bairstow		DNP - Coach's Decision								
41400122	1610612741	CHI	Chicago	203926	Doug McDermott		DNP - Coach's Decision								
41400122	1610612741	CHI	Chicago	1737	Nazr Mohammed		DNP - Coach's Decision								
41400122	1610612741	CHI	Chicago	202734	ETwaun Moore		DNP - Coach's Decision								



Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
FTM	FTA	FT_PCT	OREB	DREB	REB	AST	STL	BLK	TO	PF	PTS	PLUS_MINUS
2	2	1	2	9	11	4	2	2	2	4	6	-1
2	2	1	1	5	6	0	0	1	0	1	8	-6
1	2	0.5	1	5	6	3	1	2	0	2	7	-2
3	3	1	1	5	6	0	1	1	1	2	22	-13
2	3	0.667	0	5	5	2	1	1	1	4	12	-6
0	1	0	3	3	6	0	1	2	0	4	8	-7
0	0	0	1	2	3	0	4	0	0	1	3	2
2	2	1	0	2	2	1	0	0	0	2	8	-3
2	2	1	0	3	3	3	0	0	0	2	8	-9
0	0	0	1	5	6	3	0	0	0	1	12	4
3	6	0.5	2	14	16	3	0	4	3	1	11	4
0	0	0	4	15	19	5	0	1	3	3	6	0
8	14	0.571	2	7	9	2	0	1	0	2	31	12
5	5	1	3	4	7	9	1	1	3	3	15	8
0	0	0	1	0	1	0	0	3	3	4	0	-4
0	0	0	0	2	2	1	0	0	1	2	3	9
1	2	0.5	1	3	4	1	1	0	0	2	8	11
0	0	0	0	0	0	2	0	0	0	1	5	1

包括了对战球队（芝加哥公牛队与密尔沃基雄鹿队），球员名称与 ID，首发球员位置，DNP 原因，出场时间，进球数，出手数，命中率，三分进球数，三分球命中率，罚球命中数，罚球出手数，罚球命中率，进攻篮板，防守篮板，总篮板，助攻，抢断，封盖，犯规数，得分，以及正负值。这些数据非常有用，对于分析具体的球员表现，分析重点比赛场次很有意义。

基于这两个 API，我们可以在几十种不同的模块中收集很多的数据，NBA 一直以来都提供非常具体充实的数据供球迷查询分析，供专家预测，供球队制订战术与建队策略。

3.3 基于位置的大数据

我们并没有将数据收集满足于上述步骤。经过我们对国内外主流篮球论坛的分析，借鉴了众多对比赛分析预测的经验与失误之后，认识到传统数据分析的最大问题（不管是最基本的数据分析还是包含了上一部分的‘大数据’分析），就是在于数据并不能反映球场的真实情况。一场比赛 48 分钟（不计加时），球场局势瞬息万变，往往因为一个球员的防守或者进攻问题（防守不好，对位被针对，投篮失准被防空，被夹击）等等问题以及战术调整等等产生剧烈的变化。哪怕数据统计‘精确到’每场，依然远不足以反映球场形势。仅举几个简单的例子，去年总决

赛 G7 最后将近五分钟的时间内，勇士与骑士一分未得，然而这五分钟两队的攻防令人窒息，相当精彩，在数据上却完全无法体现，更不用说伊戈达拉的快攻如何被詹姆斯封盖，欧文的单打三分如何一剑封喉等等瞬间，均无法在纯数字数据中得到体现。今年的常规赛中骑士队因为球员年龄较大，詹姆斯习惯性常规赛留力，不过分追求战绩等问题，很多常规比赛数据相当不好看，但是季后赛却继续横扫对手，统治力十足。可见基于基本的数据做出的预测往往会令人质疑，但是这些主观的球场的表现，又无法被量化，因此产生了本段阐述的，数据无法反应真实表现的情况。

我们发现了 stats.nba 公司近些年引进的技术 SportVu。该技术通过在球场架设多部超高速超清专用摄像机，每秒捕捉球场动态 25 次，来获得过去无法获得的无比准确丰富的球场信息。然而进一步的寻找令人失望，该数据并未向公众提供，无法通过 API 获取。

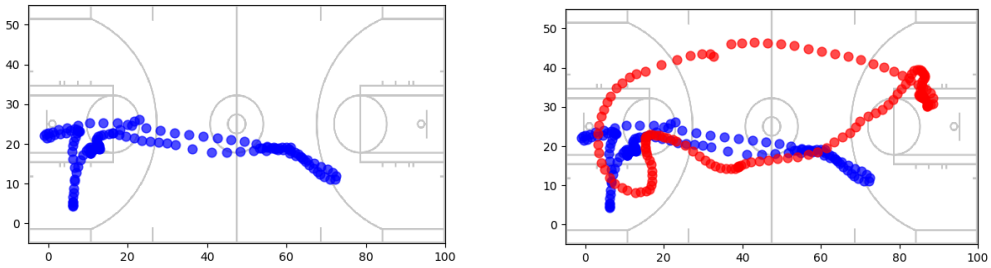
为此我们首先联系了上述文章的作者，经过询问后被告知，在 SportVU 技术的早期推广阶段可以获取该数据，最近几年的数据已经不再公开。但是在 https://github.com/savvastj/BasketballData/tree/master/2016.NBA.Raw.SportVU.Game.Logs 页面我们发现了 2016 年上半年的部分场次的 SportVU 打包数据。我们下载了其中一场比赛（骑士与勇士）的数据做初步的分析。我们将数据做了初步的转换与处理，然后存储到 csv 文件中（代码见文尾）。打开 csv 文件可以看到一场比赛会产生约 60 万行的海量数据，该数据为每 0.04 秒一次采样，包括在场十位球员的姓名，球衣号，位置坐标，篮球的坐标与高度，基于这样的表格，我们可以利用已经实现的程序绘制正常比赛的动画图像，分析具体的球场细节信息。

为了更准确详细地分析两支球队比赛的情况，我们希望利用 SportVu 技术的

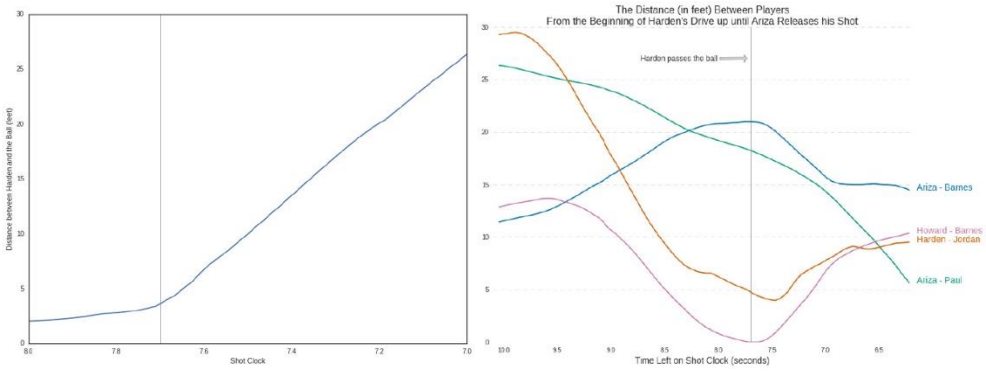
原始数据重现比赛，达到分析球队表现的目的。

3.3.1 简单尝试

利用 SportVu 技术可以做很多的事情，既可以很简单地计算出某球员的跑动距离与速度，也可以绘制一些可视化的结果，比如我们的一些尝试：



绘制单个球员或者对位球员的轨迹图



表示球与球员的距离以及球员间距离

3.3.2 进一步利用数据表示场上情况

为了充分利用数据，我们编写了一个将原始数据转化为动画的脚本（见文尾）。可以把全场比赛用抽象的球员的点表示出来。通过观看动画，我们可以分析球队的战术，简单地了解到球场上的情况。

3.4 利用深度学习技术分析比赛

我们也希望尝试深度学习方法，让机器为我们分析球场情况，做出对球队表

现的判断，来补充我们对球队等级分和获胜概率的判断。

3.4.1 目标与难点

我们对深度学习能够做的事情的认识产生了不断的改变，最初我们的目标很简单：用得分与否的数据作为 label，输入视频进行训练，让深度学习模型学会预测能否得分，后来我们发现一旦可以输入视频，就意味着比赛已经打完了，我们想预测的'未来'其实已经发生了，但是深度学习模型判断得分和判断球队的表现好坏，评价球队能否成功是类似的，所以初步来看我们可以用旧的比赛视频更好的分析这两支球队的表现，状态，就像专家看比赛录像做出自己的判断类似，最终我们会根据最新进行的比赛结果来修正我们的 ELO 指数，为接下来的比赛提供更好的预测。

利用深度学习技术有不少难点：1 视频分析的难点：比图像分析要困难。2 特征工程：如何找到更多的特征，提高训练效率与质量。3 训练时间长，开销大。

3.4.2 特征工程

利用深度学习分析视频本来就比分析图片信息要复杂得多，因此更好的特征输入会有助于网络更好的学习。

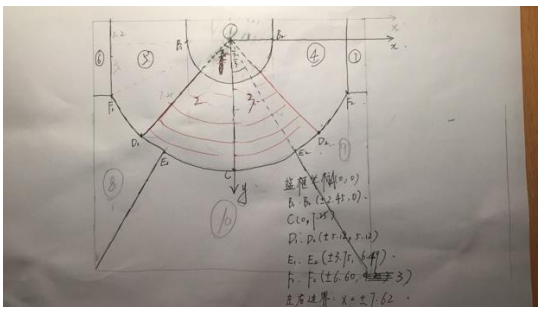
我们假设 RNN+CNN 的模型可以像人一样，分析球场上展示出的各种跑位与战术。我们的动画的优势在于较为简单，比较好分析，但是为了更好的分析结果，还需要更多的特征。

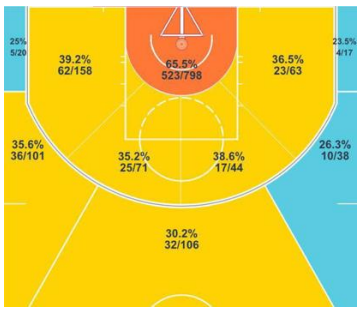
战术分析是很复杂的工程，最基础的战术都有几十种，比如 A 基础配合：传切、突分、掩护、策应、快攻 B 进攻：单中锋、双中锋进攻、8 字掩护进攻、移动进攻，反掩护策应进攻，后场配合进攻、中场配合进攻、插中策应进攻。C 防守防守站位：1-3-1，1-2-2，2-2-1，2-1-2，1-2-1-1.等等。还有各种战术的变

种。我们不懂战术，只是普通爱好者，但是可以通过提取特征，把场上的情况交给机器分析。比如战术、跑位等特征都可以在动画中展现出来，这些特征交给机器提取，学习。

我们认为影响命中率的关键因素在于球员个人的命中率、是否空位、包夹等因素。球员的每帧的微小的运动（micro action），都在于达成一些宏观的目标（macro action）：跑动至命中率高的区域，甩开防守球员，获得空位。因此我们需要给机器输入一些仅仅从动画中无法学习到的特征。

比如不同球员的命中率数据，就需要我们输入，让机器更好地学习、评价球员的出手选择等高阶特征。以 15-16 年的一场常规赛为例，将球场划分为十个部分，统计球员在每个区域的赛季命中率，作为特征输入。根据位置坐标，索引本时刻球员所处的命中率区域。我们计算得到了球场的划分方法，因此可以根据球员的位置信息判断出手球员正处于球场的哪个区域，判断方法为从球员出发引一条射线，只有一个交点的区域为所处区域。





我们对球场的划分以及球员（Lebron James 当赛季的不同区域投篮命中率）

我们还考虑了更多的特征，1 得分情况：得分与否，两分球/三分球/罚球，在哪个区域得分，哪个队员得分。2 球员情况：场上十名球员的情况：因为很多特征长期不变，可以构建一个大数组，由位置坐标检索，与场上球员对应。球员在场上位置（中锋、前锋、后卫）；身体素质数据：身高，体重，臂展。助攻，前场

篮板，后场篮板，体力值（速度的四次方函数）。其中数据特征做归一化处理即可，离散特征可以用 one-hot 向量表示。特征更新频率：每五帧（0.2 秒）。输出结果由进攻回合划分，当判断出某回合结束后，输出得分的情况。

我们现在只是做到了特征的选取和数据收集工作，归一化和 label 的输入还没有进行。

3.4.3 模型选择

我们选择了比较常用的视频处理模型，two stream CNN。将单帧的图像信息和帧与帧之间的变化信息进行融合。单帧的图像可以形成对空间的描述，而通过光流法等方法形成对时间信息的描述（差分），从而达到时间和空间互补的目的，分析视频信息。

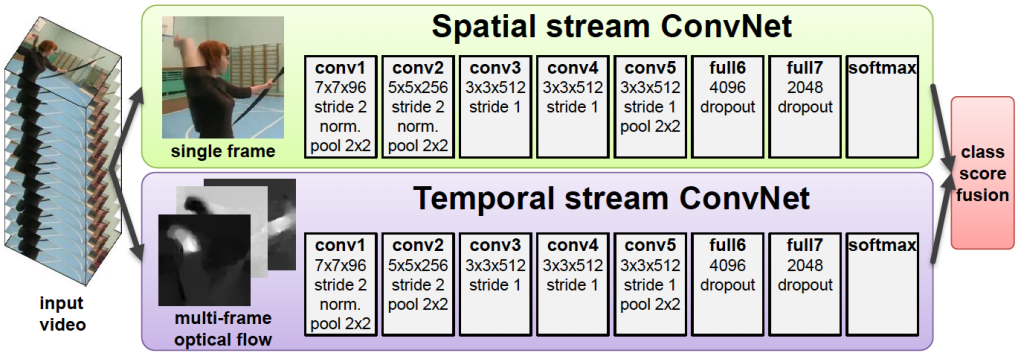


Figure 1: **Two-stream architecture for video classification.**

未来我们可以在处理好特征问题后，使用该模型分析动画，需要处理的问题在于计算开销和时间上。利用深度学习处理视频本身也不够成熟，还有待于更多的网络的开发。

3.5 未来展望

如果未来还有机会，或者更专业的篮球分析专家，可以考虑更好地利用新技术来分析大数据。赛场上的实时数据被 SportVU 及时记录（各支球队均已购买

SportVU 专业设备），利用模型来分析自己和对手球队的战术，分析出下一步最佳的战术策略，帮助球队教练组更好的决策。

其中的难点在于：篮球比赛可以算作不完全信息博弈的一种，场上的情况非常复杂，根据已有的战术动画分析成功率等信息稍微容易，预测接下来的策略就比较复杂。用简单的深度学习模型无法处理，但是目前利用深度学习解决不完全信息博弈问题的尝试也越来越多，因此未来的发展也许可期。

四.参考文献

【1】逻辑回归，http://blog.csdn.net/pakko/article/details/37878837

【2】NBA 常规赛结果预测——利用 Python 进行比赛数据分析

https://www.shiyanlou.com/courses/782/labs/2647/document

【3】stats.nba.com

【4】probasketballapi.com

【5】Python client for NBA statistics located at stats.nba.com

https://github.com/seemethere/nba_py

【6】Basketball Reference.com

【7】 周志华，《机器学习》，北京：清华大学出版社，2016，57—60

【8】https://wenku.baidu.com/view/2538ef4fd4d8d15abf234e0c.html

【9】https://zhuanlan.zhihu.com/p/26883727

【10】

http://wiki.mbalib.com/wiki/%E4%B8%BB%E6%88%90%E5%88%86%E5%88%86%E6%9E%90%E6%B3%9