

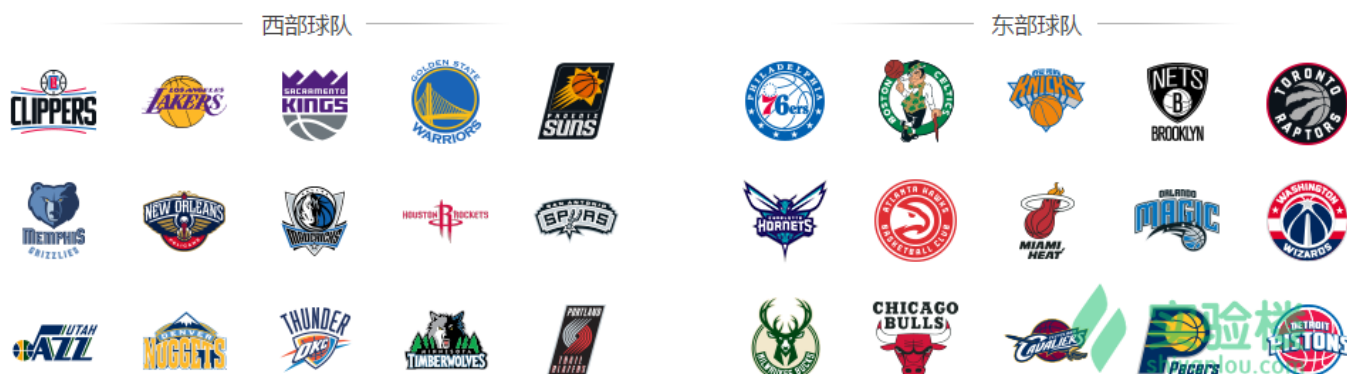
# 利用Python进行NBA比赛数据分析

## 一、实验介绍

### 1.1 内容简介

不知道你是否朋友圈被刷屏过nba的某场比赛进度或者结果？或者你就是一个nba狂热粉，比赛中的每个进球，抢断或是逆转压哨球都能让你热血沸腾。除去观赏精彩的比赛过程，我们也同样好奇比赛的结果会是如何。因此本节课程，将给同学们展示如何使用nba比赛的以往统计数据，判断每个球队的战斗力的，及预测某场比赛中的结果。

我们将基于2015-2016年的**NBA**常规赛及季后赛的比赛统计数据，预测在当下正在进行的2016-2017常规赛每场赛事的结果。



### 1.2 实验知识点

- nba球队的 Elo score 计算
- 特征向量
- 逻辑回归

### 1.3 实验环境

- python2.7
- Xfce终端

### 1.4 实验流程

本次课程我们将按照下面的流程实现**NBA**比赛数据分析的任务：

### ③ NBA常规赛结果预测——利用Python进行比赛数据分析 (/courses/782)

1. 获取比赛统计数据
2. 比赛数据分析，得到代表每场比赛每支队伍状态的特征表达
3. 利用**机器学习**方法学习每场比赛与胜利队伍的关系，并对2016-2017的比赛进行预测

## 1.5 代码获取

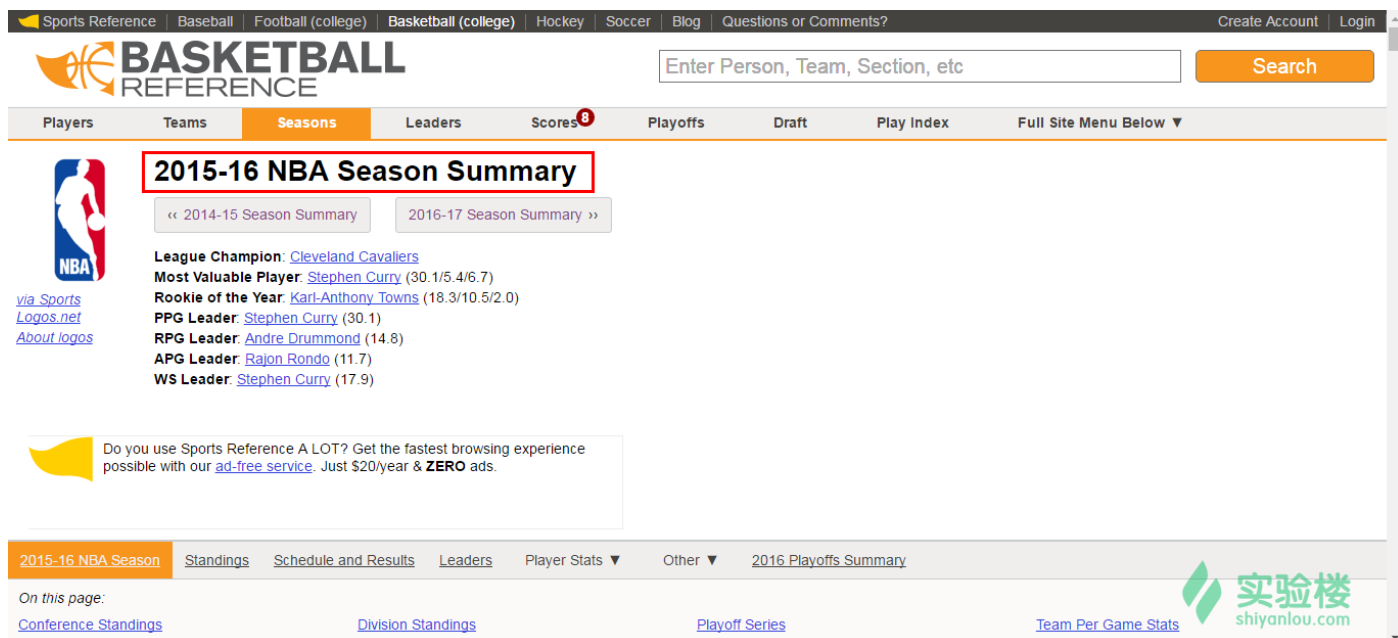
本次实验的源码可通过以下命令获得：

```
$ wget http://labfile.oss.aliyuncs.com/courses/782/prediction.py
```

## 二、获取 NBA比赛统计数据

### 2.1 比赛数据介绍

在本次实验中，我们将采用Basketball Reference.com (<http://www.basketball-reference.com/>) 中的统计数据。在这个网站中，你可以看到不同球员、队伍、赛季和联盟比赛的基本统计数据，如得分，犯规次数等情况，胜负次数等情况。而我们在这里将会使用**2015-16 NBA Season Summary**中数据。



在这个2015-16总结的所有表格中，我们将使用的是以下三个数据表格：

- **Team Per Game Stats**：每支队伍平均每场比赛的表现统计

数据名	含义
Rk -- Rank	排名

## 数据名

## 含义

5 NBA常规赛结果预测——利用Python进行比赛数据分析 (/courses/782)

G -- Games	参与的比赛场数（都为82场）
MP -- Minutes Played	平均每场比赛进行的时间
FG--Field Goals	投球命中次数
FGA--Field Goal Attempts	投射次数
FG%--Field Goal Percentage	投球命中次数
3P--3-Point Field Goals	三分球命中次数
3PA--3-Point Field Goal Attempts	三分球投射次数
3P%--3-Point Field Goal Percentage	三分球命中率
2P--2-Point Field Goals	二分队命中次数
2PA--2-point Field Goal Attempts	二分队投射次数
2P%--2-Point Field Goal Percentage	二分队命中率
FT--Free Throws	罚球命中次数
FTA--Free Throw Attempts	罚球投射次数
FT%--Free Throw Percentage	罚球命中率
ORB--Offensive Rebounds	进攻篮板球
DRB--Defensive Rebounds	防守篮板球
TRB--Total Rebounds	篮板球总数
AST--Assists	助攻
STL--Steals	抢断
BLK -- Blocks	封盖
TOV -- Turnovers	失误
PF -- Personal Fouls	个犯
PTS -- Points	得分

- **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
- **Miscellaneous Stats**：综合统计数据

数据项	数据含义
9 NBA常规赛结果预测——利用Python进行比赛数据分析 (/courses/782)	
Rk (Rank)	排名
Age	队员的平均年龄
W (Wins)	胜利次数
L (Losses)	失败次数
PW (Pythagorean wins)	基于毕达哥拉斯理论计算的赢的概率
PL (Pythagorean losses)	基于毕达哥拉斯理论计算的输的概率
MOV (Margin of Victory)	赢球次数的平均间隔
SOS (Strength of Schedule)	用以评判对手选择与其球队或是其他球队的难易程度对比，0为平均线，可以为正负数
SRS (Simple Rating System)	3
ORtg (Offensive Rating)	每100个比赛回合中的进攻比例
DRtg (Defensive Rating)	每100个比赛回合中的防守比例
Pace (Pace Factor)	每48分钟内大概会进行多少个回合
FTTr (Free Throw Attempt Rate)	罚球次数所占投射次数的比例
3PAr (3-Point Attempt Rate)	三分球投射占投射次数的比例
TS% (True Shooting Percentage)	二分之一球、三分球和罚球的总共命中率
eFG% (Effective Field Goal Percentage)	有效的投射百分比（含二分之一球、三分球）
TOV% (Turnover Percentage)	每100场比赛中失误的比例
ORB% (Offensive Rebound Percentage)	球队中平均每个人的进攻篮板的比例
FT/FGA	罚球所占投射的比例
eFG% (Opponent Effective Field Goal Percentage)	对手投射命中比例
TOV% (Opponent Turnover Percentage)	对手的失误比例

数据项	数据含义
DRB% (Defensive Rebound Percentage)	球队平均每个球员的防守篮板比例
FT/FGA (Opponent Free Throws Per Field Goal Attempt)	对手的罚球次数占投射次数的比例

毕达哥拉斯定律：

$$win\% = \frac{runs\ scored^2}{runs\ scored^2 + runs\ allowed^2}$$

我们将用这三个表格来评估球队过去的战斗力，另外还需2015-16 NBA Schedule and Results 中的2015~2016年的nba常规赛及季后赛的每场比赛的比赛数据，用以评估 Elo score（在之后的实验小节中解释）。在 Basketball Reference.com 中按照从常规赛至季后赛的时间。列出了2015年10月份至2016年6月份的每场比赛的比赛情况。

2015-16 NBA Season

Standings

Schedule and Results

Leaders

Player Stats ▼

Other ▼

October

November

December

January

February

March

April

May

June

October Schedule

Share & more ▼

Date	Start (ET)	Visitor/Neutral	PTS	Home/Neutral	PTS		Notes
Tue, Oct 27, 2015	8:00 pm	Detroit Pistons	106	Atlanta Hawks	94	Box Score	
Tue, Oct 27, 2015	8:00 pm	Cleveland Cavaliers	95	Chicago Bulls	97	Box Score	
Tue, Oct 27, 2015	10:30 pm	New Orleans Pelicans	95	Golden State Warriors	111	Box Score	
Wed, Oct 28, 2015	7:30 pm	Philadelphia 76ers	95	Boston Celtics	112	Box Score	
Wed, Oct 28, 2015	7:30 pm	Chicago Bulls	115	Brooklyn Nets	100	Box Score	
Wed, Oct 28, 2015	7:30 pm	Utah Jazz	87	Detroit Pistons	92	Box Score	
Wed, Oct 28, 2015	8:00 pm	Denver Nuggets	105	Houston Rockets	85	Box Score	
Wed, Oct 28, 2015	10:30 pm	Minnesota Timberwolves	112	Los Angeles Lakers	111	Box Score	
Wed, Oct 28, 2015	8:00 pm	Cleveland Cavaliers	106	Memphis Grizzlies	76	Box Score	
Wed, Oct 28, 2015	7:30 pm	Charlotte Hornets	94	Miami Heat	104	Box Score	
Wed, Oct 28, 2015	8:00 pm	New York Knicks	122	Milwaukee Bucks	97	Box Score	



实验楼

shiyanlou.com

可在上图中，看到2015年10月份的部分比赛数据。在每个Schedule表格中所包含的数据为：

数据项	数据含义
Date	比赛日期
Start (ET)	比赛开始时间
Visitor/Neutral	客场作战队伍
PTS	客场队伍最后得分

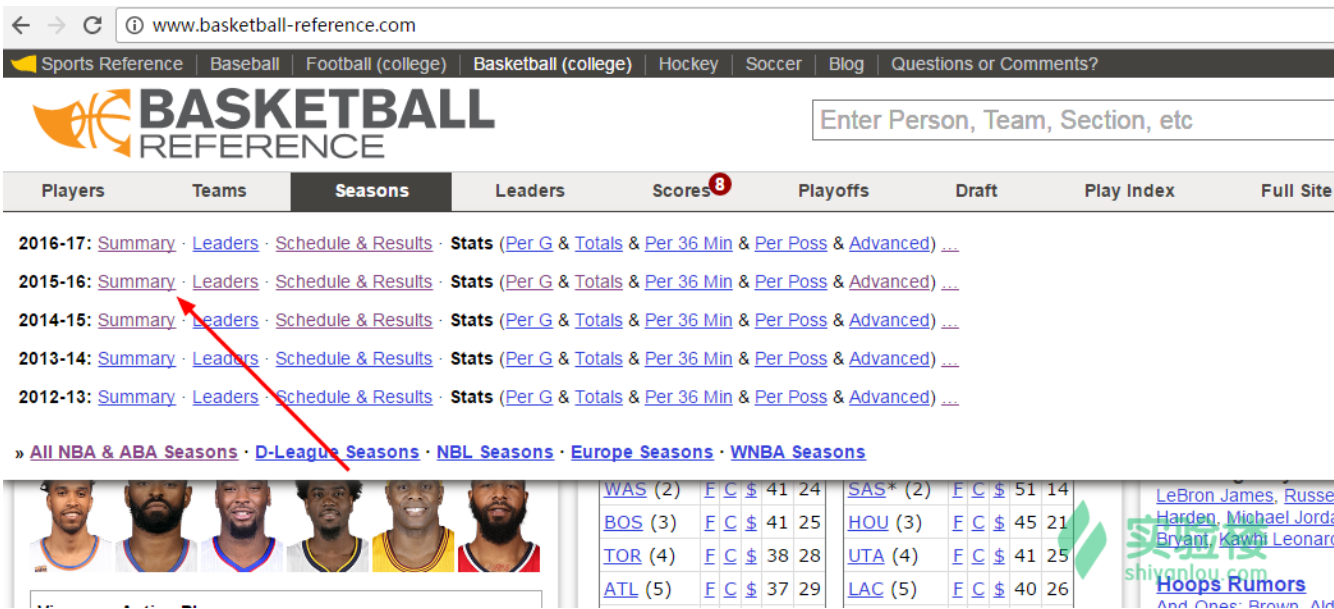
数据项	数据含义
⑤ NBA常规赛结果预测——利用Python进行比赛数据分析 (/courses/782)	
Home/Neutral	主场队伍
PTS	主场队伍最后得分
Notes	备注，表明是否为加时赛等

在预测时，我们同样也需要在2016-17 NBA Schedule and Results中2016~2017年的NBA的常规赛比赛安排数据。

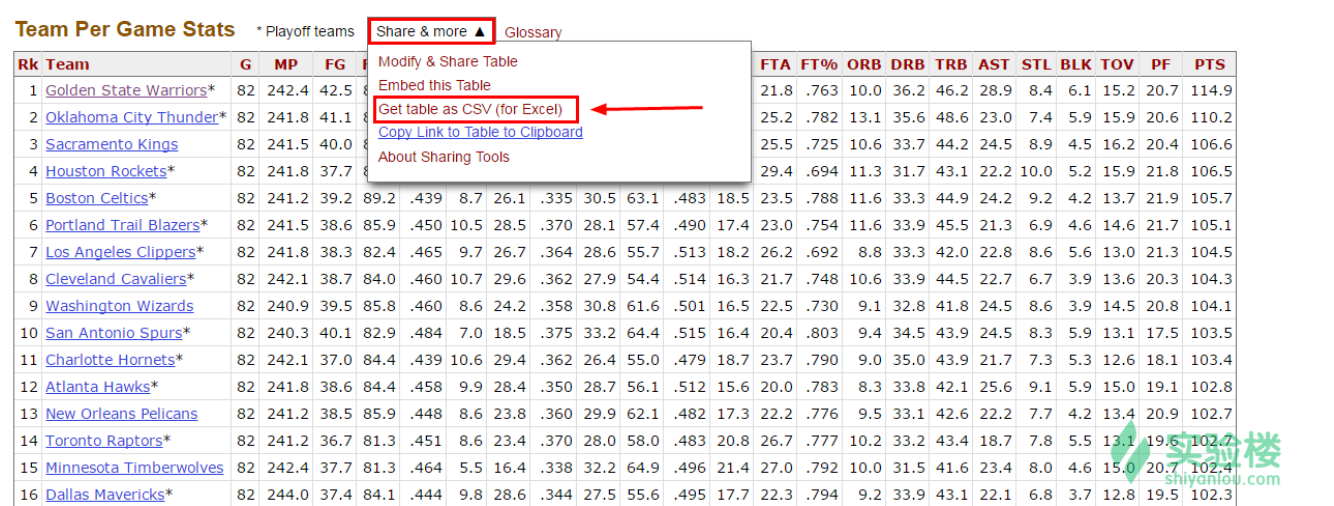
## 2.2 获取比赛数据

我们将以获取Team Per Game Stats表格数据为例，展示如何获取这三项统计数据。

1. 进入到用Basketball Reference.com (http://www.basketball-reference.com/)中，在导航栏中选择 Season 并选择 2015~2016 赛季中的 Summary：



2. 进入到2015~2016年的 Summary 界面后，滑动窗口找到 Team Per Game Stats 表格，并选择左上方的Share & more，在其下拉菜单中选择Get table as CSV (for Excel)：





3. 复制在界面中生成的csv格式数据，并复制粘贴至一个文本编辑器保存为csv文件即可：  
🕒 NBA常规赛结果预测——利用Python进行比赛数据分析 (/courses/782)

Team Per Game Stats \* Playoff teams Share & more Glossary

Reload page to return to the table-formatted data.

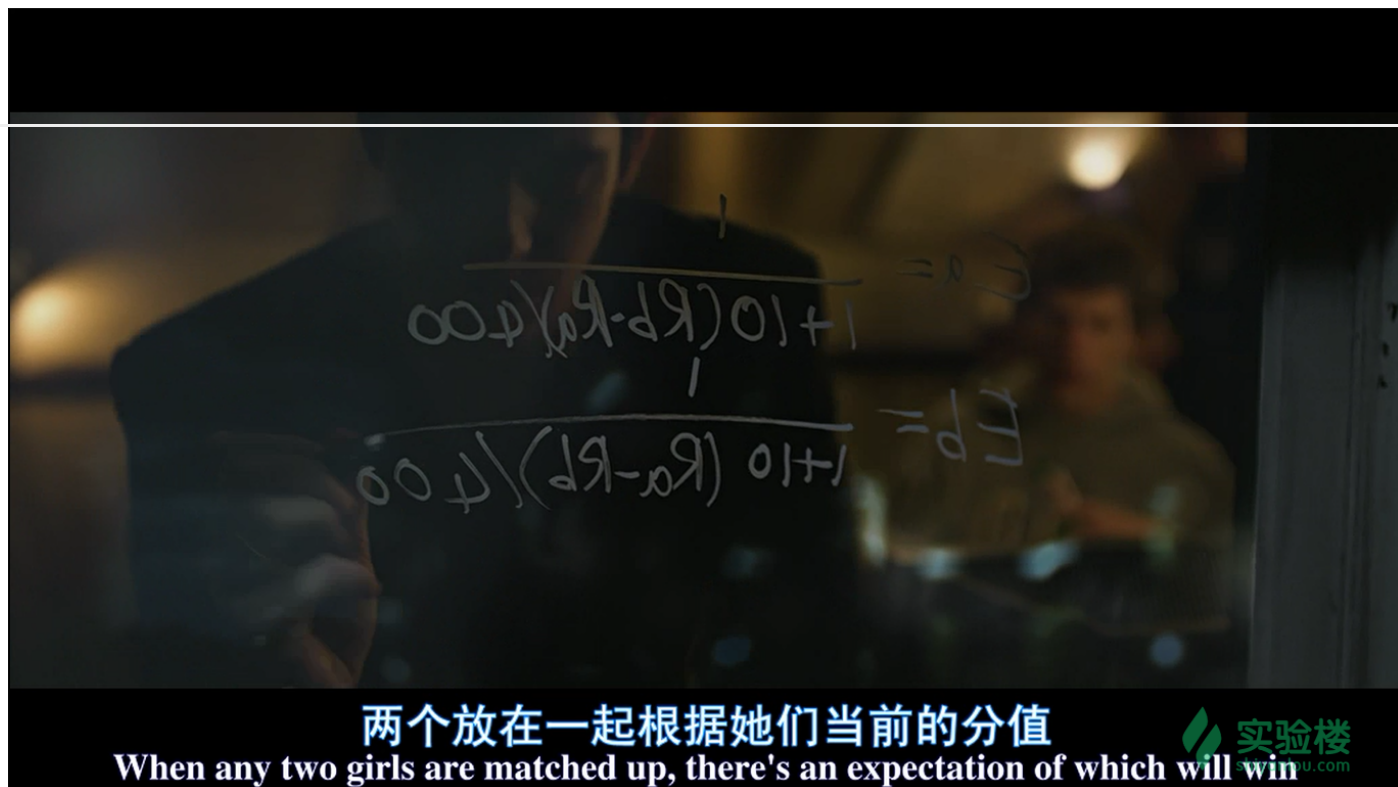
```
Rk, Team, G, MP, FG, FGA, FG%, 3P, 3PA, 3P%, 2P, 2PA, 2P%, FT, FTA, FT%, ORB, DRB, TRB, AST, STL, BLK, TOV, PF, PTS
1, Golden State Warriors*, 82, 242.4, 42.5, 87.3, .487, 13.1, 31.6, .416, 29.4, 55.7, .528, 16.7, 21.8, .763, 10.0, 36.2, 46.2, 28.9, 8.4, 6.1, 15.2, 20.7, 114.9
2, Oklahoma City Thunder*, 82, 241.8, 41.1, 86.4, .476, 8.3, 23.7, .349, 32.9, 62.6, .524, 19.7, 25.2, .782, 13.1, 35.6, 48.6, 23.0, 7.4, 5.9, 15.9, 20.6, 110.2
3, Sacramento Kings, 82, 241.5, 40.0, 86.4, .464, 8.0, 22.4, .359, 32.0, 64.0, .500, 18.5, 25.5, .725, 10.6, 33.7, 44.2, 24.5, 8.9, 4.5, 16.2, 20.4, 106.6
4, Houston Rockets*, 82, 241.8, 37.7, 83.5, .452, 10.7, 30.9, .347, 27.0, 52.6, .514, 20.4, 29.4, .694, 11.3, 31.7, 43.1, 22.2, 10.0, 5.2, 15.9, 21.8, 106.5
5, Boston Celtics*, 82, 241.2, 39.2, 89.2, .439, 8.7, 26.1, .335, 30.5, 63.1, .483, 18.5, 23.5, .788, 11.6, 33.3, 44.9, 24.2, 9.2, 4.2, 13.7, 21.9, 105.7
6, Portland Trail Blazers*, 82, 241.5, 38.6, 85.9, .450, 10.5, 28.5, .370, 28.1, 57.4, .490, 17.4, 23.0, .754, 11.6, 33.9, 45.5, 21.3, 6.9, 4.6, 14.6, 21.7, 105.1
7, Los Angeles Clippers*, 82, 241.8, 38.3, 82.4, .465, 9.7, 26.7, .364, 28.6, 55.7, .513, 18.2, 26.2, .692, 8.8, 33.3, 42.0, 22.8, 8.6, 5.6, 13.0, 21.3, 104.5
8, Cleveland Cavaliers*, 82, 242.1, 38.7, 84.0, .460, 10.7, 29.6, .362, 27.9, 54.4, .514, 16.3, 21.7, .748, 10.6, 33.9, 44.5, 22.7, 6.7, 3.9, 13.6, 20.3, 104.3
9, Washington Wizards, 82, 240.9, 39.5, 85.8, .460, 8.6, 24.2, .358, 30.8, 61.6, .501, 16.5, 22.5, .730, 9.1, 32.8, 41.8, 24.5, 8.6, 3.9, 14.5, 20.8, 104.1
10, San Antonio Spurs*, 82, 240.3, 40.1, 82.9, .484, 7.0, 18.5, .375, 33.2, 64.4, .515, 16.4, 20.4, .803, 9.4, 34.5, 43.9, 24.5, 8.3, 5.9, 13.1, 17.5, 103.5
11, Charlotte Hornets*, 82, 242.1, 37.0, 84.4, .439, 10.6, 29.4, .362, 26.4, 55.0, .479, 18.7, 23.7, .790, 9.0, 35.0, 43.9, 21.7, 7.3, 5.3, 12.6, 18.1, 103.4
12, Atlanta Hawks*, 82, 241.8, 38.6, 84.4, .458, 9.9, 28.4, .350, 28.7, 56.1, .512, 15.6, 20.0, .783, 8.3, 33.8, 42.1, 25.6, 9.1, 5.9, 15.0, 19.1, 102.8
13, New Orleans Pelicans, 82, 241.2, 38.5, 85.9, .448, 8.6, 23.8, .360, 29.9, 62.1, .482, 17.3, 22.2, .776, 9.5, 33.1, 42.6, 22.2, 7.7, 4.2, 13.4, 20.9, 102.7
14, Toronto Raptors*, 82, 241.2, 36.7, 81.3, .451, 8.6, 23.4, .370, 28.0, 58.0, .483, 20.8, 26.7, .777, 10.2, 33.2, 43.4, 18.7, 7.8, 5.5, 13.1, 19.6, 102.7
15, Minnesota Timberwolves, 82, 242.4, 37.7, 81.3, .464, 5.5, 16.4, .338, 32.2, 64.9, .496, 21.4, 27.0, .792, 10.0, 31.5, 41.6, 23.4, 8.0, 4.6, 15.0, 20.7, 102.4
16, Dallas Mavericks*, 82, 244.0, 37.4, 84.1, .444, 9.8, 28.6, .344, 27.5, 55.6, .495, 17.7, 22.3, .794, 9.2, 33.9, 43.1, 22.1, 6.8, 3.7, 12.8, 19.5, 102.3
17, Indiana Pacers*, 82, 242.4, 38.3, 85.2, .450, 8.1, 23.0, .351, 30.2, 62.1, .486, 17.4, 22.8, .764, 10.3, 33.9, 44.2, 21.2, 9.0, 4.8, 14.9, 20.0, 102.2
18, Orlando Magic, 82, 242.7, 39.5, 86.8, .455, 7.8, 22.2, .350, 31.8, 64.7, .492, 15.2, 20.1, .757, 10.3, 33.0, 43.3, 23.6, 8.2, 5.1, 14.1, 20.7, 102.1
19, Detroit Pistons*, 82, 242.4, 37.9, 86.4, .439, 9.0, 26.2, .345, 28.9, 60.2, .480, 17.1, 25.5, .668, 12.5, 33.9, 46.3, 19.4, 7.0, 3.7, 13.5, 19.0, 102.0
20, Denver Nuggets, 82, 241.8, 37.7, 85.4, .442, 8.0, 23.7, .338, 29.7, 61.7, .482, 18.5, 24.1, .766, 11.5, 33.1, 44.6, 22.7, 7.4, 4.8, 14.7, 21.0, 101.9
21, Chicago Bulls, 82, 242.7, 38.6, 87.4, .441, 7.9, 21.4, .371, 30.7, 66.1, .464, 16.5, 21.0, .787, 11.1, 35.2, 46.3, 22.8, 6.0, 5.7, 13.9, 18.8, 101.6
22, Phoenix Suns, 82, 240.3, 37.2, 85.6, .435, 9.0, 25.8, .348, 28.2, 59.8, .472, 17.5, 23.2, .751, 11.5, 33.3, 44.8, 20.7, 7.7, 3.8, 17.2, 22.7, 100.9
23, Miami Heat*, 82, 241.8, 38.4, 81.7, .470, 6.1, 18.0, .336, 32.3, 63.6, .508, 17.1, 23.0, .744, 9.8, 34.3, 44.1, 20.8, 6.7, 6.5, 14.1, 18.3, 100.0
24, Memphis Grizzlies*, 82, 241.8, 36.8, 83.6, .440, 6.1, 18.5, .331, 30.7, 65.1, .471, 19.3, 24.7, .783, 11.2, 30.5, 41.6, 20.7, 8.8, 4.3, 13.3, 21.7, 99.1
25, Milwaukee Bucks, 82, 241.8, 38.4, 82.2, .467, 5.4, 15.6, .345, 33.0, 66.6, .495, 17.0, 22.7, .747, 10.5, 31.2, 41.7, 23.1, 8.2, 5.8, 15.2, 20.7, 99.0
26, Brooklyn Nets, 82, 240.9, 38.2, 84.4, .453, 6.5, 18.4, .352, 31.8, 66.0, .481, 15.7, 20.7, .757, 10.5, 31.9, 42.4, 22.3, 7.6, 4.0, 14.8, 18.0, 98.6
27, New York Knicks, 82, 241.5, 36.9, 84.0, .439, 7.4, 21.5, .346, 29.4, 62.5, .471, 17.2, 21.4, .805, 10.4, 34.0, 44.4, 20.5, 5.7, 5.7, 13.4, 19.7, 98.4
28, Utah Jazz, 82, 243.4, 36.1, 80.4, .449, 8.5, 23.9, .355, 27.6, 56.5, .488, 17.1, 23.0, .744, 10.7, 32.5, 43.2, 19.0, 7.7, 5.2, 14.9, 20.2, 97.7
```

为了方便同学们进行实验，我们已经将数据全部都保存成csv文件上传至实验楼的云环境中。在后续的代码实现小节里，我们将给出获取这些文件的地址。

## 三、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。因此我们将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。

关于Elo score等级分，不知道同学们是否看过《社交网络》这部电影，在这部电影中，Mark（主人公原型就是扎克伯格，Facebook创始人）在电影起初开发的一个美女排名系统就是利用其好友Eduardo在窗户上写下的排名公式，对不同的女生进行等级制度对比，最后PK出胜利的一方。



这条对比公式就是**Elo Score等级分制度**。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。在上图中**Eduardo**在窗户上写下的公式就是根据 Logistic Distribution 计算PK双方（A和B）对各自的胜率期望值计算公式。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：



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

shiyaniou.com

在国际象棋中，根据等级分的不同K值也会做相应的调整：

- $\geq 2400$  ,  $K=16$
- 2100~2400分 ,  $K=24$
- $\leq 2100$  ,  $K=32$

因此我们将会用以表示某场比赛数据的**特征向量**为（加入A与B队比赛）：[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]

## 四、基于数据进行模型训练和预测

### 4.1 实验前期准备

在本次实验环境中，我们将会使用到python的 pandas , numpy , scipy 和 sklearn 库，不过实验楼中已经安装了 numpy ，所以在实验前，我们需要先利用 pip 命令安装另外三个Python库。

```
$ sudo pip install pandas
$ sudo pip install scipy
$ sudo pip install sklearn
```

在安装完所需的实验库之后，进入到实验环境的 Code 目录下，创建 cs\_782 文件夹，并且通过以下地址获取我们为大家处理好的csv文件压缩包 data.zip：

```
$ cd Code
$ mkdir cs_782 && cd cs_782

# 获取数据文件
$ wget http://labfile.oss.aliyuncs.com/courses/782/data.zip

# 解压data压缩包并且删除该压缩包
$ unzip data.zip
$ rm -r data.zip
```

在 data 文件夹中，包含了2015~2016年的NBA数据**T,O和M表**，及经处理后的常规赛和挑战赛的  
比赛数据 2015~16result.csv ，这个数据文件是我们通过在 basketball-reference.com 的2015-16  
Schedule and result的几个月份比赛数据中提取得到的，其中包括三个字段：

- WTeam: 比赛胜利队伍
- LTeam: 失败队伍

- WLoc: 胜利队伍一方所在的为主场或是客场 另外一个文件就是 16-17Schedule.csv , 也是经过
- ③ NBA常规赛结果预测——利用Python进行比赛数据分析 (/courses/782)  
我们加工处理得到的NBA在2016~2017年的常规赛的安排, 其中包括两个字段:
- Vteam: 访问/客场作战队伍
- Hteam: 主场作战队伍

## 4.2 代码实现

在 Code\cs\_782 目录下, 创建 prediction.py 开始实验。

首先插入实验相关模块:

```
# -*- coding:utf-8 -*-
import pandas as pd
import math
import csv
import random
import numpy as np
from sklearn import linear_model
from sklearn.model_selection import cross_val_score
```

设置回归训练时所需用到的参数变量:

```
# 当每支队伍没有elo等级分时, 赋予其基础elo等级分
base_elo = 1600
team_elos = {}
team_stats = {}
X = []
y = []
folder = 'data' #存放数据的目录
```

在最开始需要初始化数据, 从**T**、**O**和**M**表格中读入数据, 去除一些无关数据并将这三个表格通过 **Team** 属性列进行连接:

```
# 根据每支队伍的Miscellaneous Opponent, Team统计数据csv文件进行初始化
def initialize_data(Mstat, Ostat, Tstat):
    new_Mstat = Mstat.drop(['Rk', 'Arena'], axis=1)
    new_Ostat = Ostat.drop(['Rk', 'G', 'MP'], axis=1)
    new_Tstat = Tstat.drop(['Rk', 'G', 'MP'], axis=1)

    team_stats1 = pd.merge(new_Mstat, new_Ostat, how='left', on='Team')
    team_stats1 = pd.merge(team_stats1, new_Tstat, how='left', on='Team')
    return team_stats1.set_index('Team', inplace=False, drop=True)
```

获取每支队伍的 Elo Score 等级分函数, 当在开始没有等级分时, 将其赋予初始 base\_elo 值:

## 📍 NBA常规赛结果预测——利用Python进行比赛数据分析 (/courses/782)

```
def get_elo(team):
    try:
        return team_elos[team]
    except:
        # 当最初没有elo时，给每个队伍最初赋base_elo
        team_elos[team] = base_elo
        return team_elos[team]
```

定义计算每支球队的 Elo等级分 函数：

```
# 计算每个球队的elo值
def calc_elo(win_team, lose_team):
    winner_rank = get_elo(win_team)
    loser_rank = get_elo(lose_team)

    rank_diff = winner_rank - loser_rank
    exp = (rank_diff * -1) / 400
    odds = 1 / (1 + math.pow(10, exp))
    # 根据rank级别修改K值
    if winner_rank < 2100:
        k = 32
    elif winner_rank >= 2100 and winner_rank < 2400:
        k = 24
    else:
        k = 16
    new_winner_rank = round(winner_rank + (k * (1 - odds)))
    new_rank_diff = new_winner_rank - winner_rank
    new_loser_rank = loser_rank - new_rank_diff

    return new_winner_rank, new_loser_rank
```

基于我们初始好的统计数据，及每支队伍的**Elo score**计算结果，建立对应2015~2016年常规赛和季后赛中每场比赛的数据集（在主客场比赛时，我们认为主场作战的队伍更加有优势一点，因此会给主场作战队伍相应加上100等级分）：

## 🕒 NBA常规赛结果预测 - 利用Python进行比赛数据分析 (/courses/782)

```
def build_data_set(all_data):
    print("Building data set..")
    x = []
    skip = 0
    for index, row in all_data.iterrows():

        Wteam = row['WTeam']
        Lteam = row['LTeam']

        # 获取最初的elo或是每个队伍最初的elo值
        team1_elo = get_elo(Wteam)
        team2_elo = get_elo(Lteam)

        # 给主场比赛的队伍加上100的elo值
        if row['WLoc'] == 'H':
            team1_elo += 100
        else:
            team2_elo += 100

        # 把elo当为评价每个队伍的第一个特征值
        team1_features = [team1_elo]
        team2_features = [team2_elo]

        # 添加我们从basketball reference.com获得的每个队伍的统计信息
        for key, value in team_stats.loc[Wteam].iteritems():
            team1_features.append(value)
        for key, value in team_stats.loc[Lteam].iteritems():
            team2_features.append(value)

        # 将两支队伍的特征值随机的分配在每场比赛数据的左右两侧
        # 并将对应的0/1赋给y值
        if random.random() > 0.5:
            X.append(team1_features + team2_features)
            y.append(0)
        else:
            X.append(team2_features + team1_features)
            y.append(1)

        if skip == 0:
            print X
            skip = 1

        # 根据这场比赛的数据更新队伍的elo值
        new_winner_rank, new_loser_rank = calc_elo(Wteam, Lteam)
        team_elos[Wteam] = new_winner_rank
        team_elos[Lteam] = new_loser_rank

    return np.nan_to_num(X), y
```

最终在main函数中调用这些数据处理函数，使用sklearn的 Logistic Regression 方法建立回归模型：

## 🔗 NBA常规赛结果预测——利用Python进行比赛数据分析 (/courses/782)

```
Mstat = pd.read_csv(folder + '/15-16Miscellaneous_Stat.csv')
Ostat = pd.read_csv(folder + '/15-16Opponent_Per_Game_Stat.csv')
Tstat = pd.read_csv(folder + '/15-16Team_Per_Game_Stat.csv')

team_stats = initialize_data(Mstat, Ostat, Tstat)

result_data = pd.read_csv(folder + '/2015-2016_result.csv')
X, y = build_dataSet(result_data)

# 训练网络模型
print("Fitting on %d game samples.." % len(X))

model = linear_model.LogisticRegression()
model.fit(X, y)

#利用10折交叉验证计算训练正确率
print("Doing cross-validation..")
print(cross_val_score(model, X, y, cv = 10, scoring='accuracy', n_jobs=-1).mean())
```

最终利用训练好的模型在16~17年的常规赛数据中进行预测。

利用模型对一场新的比赛进行胜负判断，并返回其胜利的概率：

```
def predict_winner(team_1, team_2, model):
    features = []

    # team 1, 客场队伍
    features.append(get_elo(team_1))
    for key, value in team_stats.loc[team_1].iteritems():
        features.append(value)

    # team 2, 主场队伍
    features.append(get_elo(team_2) + 100)
    for key, value in team_stats.loc[team_2].iteritems():
        features.append(value)

    features = np.nan_to_num(features)
    return model.predict_proba([features])
```

在main函数中调用该函数，并将预测结果输出到 16-17Result.csv 文件中：

### ③ NBA常规赛结果预测——利用Python进行比赛数据分析 (/courses/782)

```
#利用训练好的model在16-17年的比赛中进行预测
print('Predicting on new schedule..')
schedule1617 = pd.read_csv(folder + '/16-17Schedule.csv')
result = []
for index, row in schedule1617.iterrows():
    team1 = row['Vteam']
    team2 = row['Hteam']
    pred = predict_winner(team1, team2, model)
    prob = pred[0][0]
    if prob > 0.5:
        winner = team1
        loser = team2
        result.append([winner, loser, prob])
    else:
        winner = team2
        loser = team1
        result.append([winner, loser, 1 - prob])

with open('16-17Result.csv', 'wb') as f:
    writer = csv.writer(f)
    writer.writerow(['win', 'lose', 'probability'])
    writer.writerows(result)
```

运行 prediction.py :

```
Building data set..
Fitting on 1316 game samples..
Doing cross-validation..
0.687149170637
Predicting on new schedule..
```



生成预测结果文件 16-17Result.csv 文件：

```
shiyancelou:cs_782/ $ ls
16-17Result.csv  data  prediction.py
```



```
Terminal 终端 - 16-17Result.csv (~/.Code/cs_782) - VIM
NBA常规赛结果预测——利用Python进行比赛数据分析 (/courses/782)
win,lose,probability
Cleveland Cavaliers,New York Knicks,0.91067212472724179
Golden State Warriors,San Antonio Spurs,0.67751995373786889
Portland Trail Blazers,Utah Jazz,0.78034321506747417
Boston Celtics,Brooklyn Nets,0.89100444738622164
Indiana Pacers,Dallas Mavericks,0.6345336389906695
Houston Rockets,Los Angeles Lakers,0.67805771646072954
Memphis Grizzlies,Minnesota Timberwolves,0.65491923242387262
Charlotte Hornets,Milwaukee Bucks,0.64169870433964937
New Orleans Pelicans,Denver Nuggets,0.60026772806521767
Miami Heat,Orlando Magic,0.59514948630088449
Oklahoma City Thunder,Philadelphia 76ers,0.91201833029761836
Sacramento Kings,Phoenix Suns,0.5099461445973249
Toronto Raptors,Detroit Pistons,0.7779102997181252
Atlanta Hawks,Washington Wizards,0.67738794457924079
Chicago Bulls,Boston Celtics,0.5434784503516823
Portland Trail Blazers,Los Angeles Clippers,0.68741320853332366
San Antonio Spurs,Sacramento Kings,0.77430265094727624
Indiana Pacers,Brooklyn Nets,0.7396154831577868
Dallas Mavericks,Houston Rockets,0.61942387243753616
Detroit Pistons,Orlando Magic,0.69077820118114175
Miami Heat,Charlotte Hornets,0.64380480580458488
Golden State Warriors,New Orleans Pelicans,0.85215880884083961
"16-17Result.csv" [dos] 1230L, 67635C
```



## 五、总结

在本节课程中，我们利用 Basketball-reference.com 的部分统计数据，计算每支nba比赛队伍的 Elo score，和利用这些基本统计数据评价每支队伍过去的比赛情况，并且根据国际等级划分方法 Elo Score 对队伍现在的战斗等级进行评分，最终结合这些不同队伍的特征判断在一场比赛中，哪支队伍能够占到优势。但在我们的预测结果中，与以往不同，我们没有给出绝对的正负之分，而是给出胜算较大一方的队伍能够赢另外一方的概率。当然在这里，我们所采用评价一支队伍性能的数据量还太少（只采用了15~16年一年的数据），如果想要更加准确、系统的判断，有兴趣的你当然可以从各种统计数据网站中获取到更多年份，更加全面的数据。结合不同的**回归**、**决策**机器学习模型，搭建一个更加全面，预测准确率更高的模型。在kaggle (<https://www.kaggle.com/>)中有相关的篮球预测比赛项目，有兴趣的同学可尝试一下。

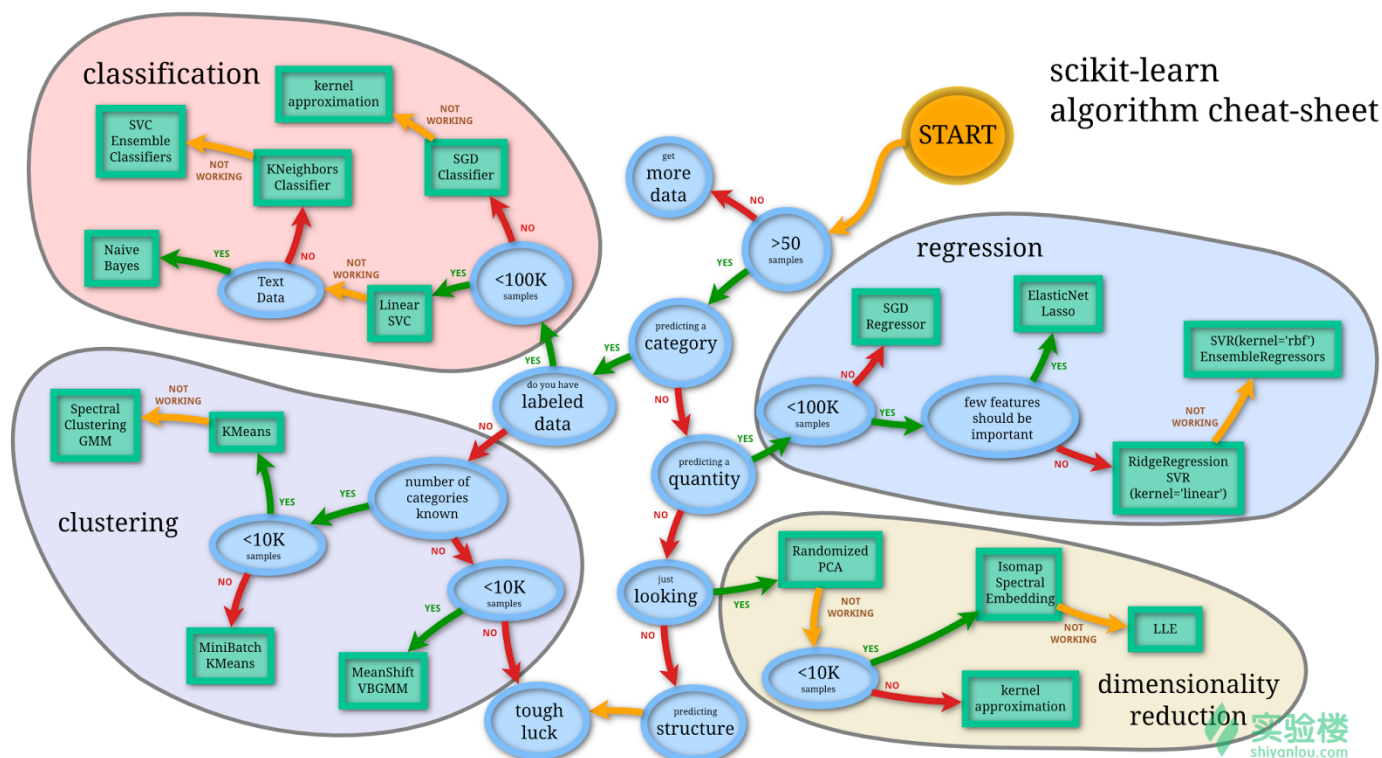
## 六、参考阅读

- 知乎：在哪能看到最全面细致的NBA数据统计  
(<https://www.zhihu.com/question/19952290>)
- How I won my NCAA tournament bracket pool using machine learning  
(<http://ftw.usatoday.com/2016/04/villanova-bracket-winner>)

## 七、课后习题

NBA常规赛结果预测——利用Python进行比赛数据分析 (/courses/782)

本次课程中，我们只是利用了 scikit-learn 提供的 Logistic Regression 方法进行回归模型的训练，你可否尝试 scikit-learn 中的其他机器学习方法，或者其他类似于 TensorFlow 的开源框架，结合我们所提供的数据集进行训练。若采用 Scikit-learn 中的方法，可参看实验楼的课程：ebay 在线拍卖数据分析 (<https://www.shiyanlou.com/courses/714>)。或是结合下图进行模型的尝试：



## 八、常见疑问解答补充

在这里我们将对之前在文档中解释得比较模糊的部分做一点补充，之后有疑问的欢迎同学们在课程讨论区提出讨论。

Q1: 在生成训练集时，“将特征值随机分配在每场比赛数据的左右侧”是什么意思？为什么要做如下的随机分配：

```
# 将两支队伍的特征值随机的分配在每场比赛数据的左右两侧
# 并将对应的0/1赋给y值
if random.random() > 0.5:
    X.append(team1_features + team2_features)
    y.append(0)
else:
    X.append(team2_features + team1_features)
    y.append(1)
```

Ans: 将胜利队伍和失败队伍的特征值随机分配到每场比赛数据的左右侧意思是, 为了随机产生  
④ NBA常规赛结果预测——利用Python进行比赛数据分析 (/courses/782)  
[winTeam, loseTeam] (胜利队伍特征值在左侧, 对应的y值标签为0), [loseTeam, winTeam]  
(失败队伍在左侧, 对应的y值标签为1) 这样的训练样本。你也可以固定利用数据集前一半为  
[winTeam, loseTeam], 后一半为[loseTeam, winTeam]这样来生成数据。只要保证两类数据的分  
布比较均衡, 且在训练时随机得到两类训练样本即可。

Q2: 为什么按照X: [winTeam, loseTeam]对应标签Y: 0, X: [loseTeam, winTeam]对应标签Y:  
1, 这样的取法在后边进行预测[team1, team2]的比赛结果时, 是否应该按照 $prob < 0.5$ 时  
team1胜出, 胜出概率为 $1 - prob$ ,  $prob > 0.5$ 时team2胜出, 胜出概率为 $prob$ ?

Ans: 可查看sklearn logistic regression api ([http://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LogisticRegression.html#sklearn.linear\\_model.LogisticRegression](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression))中有关 predict\_prob 的函数定义。

predict\_prob 返回的是对于测试样本中跟不同类别的匹配程度 ( 概率, Probability estimates ),  
对于N个类别, predict\_prob 返回的则是一个N维数组, 按照类别顺序对应这个测试样本分别属于  
这些类别的可能性。在实验中我取的是 predict\_prob[0], 所以是测试样本属于类别1[winTeam,  
loseTeam]的可能性。因此当返回的值大于 0.5 时, 我们则认为是team1胜出, 其概率为 predict\_  
prob[0], 否则是team2胜出, 胜出概率为  $1 - \text{predict\_prob}[0]$

\*本课程内容, 由作者授权实验楼发布, 未经允许, 禁止转载、下载及非法传播。



