南 开 大 学

本科生毕业论文（设计）

中文题目： 决策树和线性回归的实际应用---FIFA 18 数据分析

外文题目： Applications of decision tree and linear regression---FIFA 18 data analysis

学　　号：1410020

姓　　名：杜澧泽

年　　级：大四

专　　业：统计学

系　　别：统计系

学　　院：数学科学学院

指导教师：武颖

完成日期：2018 年 4 月 30 日

关于南开大学本科生毕业论文（设计）

的声明

本人郑重声明：所呈交的学位论文，是本人在指导教师指导下，进行研究工作所取得的成果。除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人创作的、已公开发表或没有公开发表的作品内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。本学位论文原创性声明的法律责任由本人承担。

学位论文作者签名：

2018 年 4 月 30 日

本人声明：该学位论文是本人指导学生完成的研究成果，已经审阅过论文的全部内容，并能够保证题目、关键词、摘要部分中英文内容的一致性和准确性。

学位论文指导教师签名：

2018 年 4 月 30 日

摘　要

本文基于FIFA 18球员数据,对ID3型决策树和普通线性回归在分类能力和预测能力上进行比较，找到两种方法的适用范围和使用中的优劣分析。本文主要使用了java和R两种编程语言，用R语言进行了数据可视化的分析及线性回归部分的模型构建及分析，以及决策树训练集与测试集的筛选与生成;用java实现了ID3型决策树算法，并通过训练集训练模型，对相应的测试集打印出精度和预测的分类。

最终我们可以发现决策树模型对于训练集的依赖程度比较高，当我们的训练集不是很具有代表性时，预测的能力很差;训练集的数据覆盖面越大，数据越多，针对性越强我们得到的决策树模型的可靠性就更高。另一方面,决策树模型经常面临过拟合的问题,我们常用剪枝算法和建立随机森林来解决这个问题。而线性回归模型对于模型的依赖程度很大，与实验的操作和数据的选取方式也有很大的关系，必须保证数据之间是相互独立的。线性回归模型的优点在于模型的建立不需要很多的数据即可以得到一个相对精确的模型，缺点在于过于依赖模型，当模型的假设不能被满足时，预测的效果会非常差(例如线性不能满足时)。

综合两者的特点，我们认为，决策树模型的特点使得其更容易应用在实际的分类问题中，而线性回归模型可以在建模的过程中帮助进行变量选择，辅助模型的构建。

关键词：FIFA 18；ID3 型决策树模型；数据可视化；线性回归；模型分析；模型预测

Abstract

Based on the FIFA 18 player data, this paper compares the ID3 type decision tree and the ordinary linear regression in terms of classification ability and predictive ability, and provides some insight on the application scope and the advantages and disadvantages of the two methods. Two programming languages, java and R, are used in this paper. R language is used for data visualization and analysis of the linear regression model, as well as the selection and generation of the decision tree's training set and testing set. The ID3 algorithm is implemented in java. We also summarize the accuracy and prediction classification of the corresponding testing set.

Finally, we can see that the decision tree model is more dependent on the training set. When the training set is not very representative, the ability to predict is poor. The larger the data coverage of the training set, the more data used, the stronger the pertinence for the reliability of the decision tree model. On the other hand, the decision tree model often faces the problem of overfitting. We can prune the branches or construct random forest to fix it. For the linear regression model, the degree of dependence on the model is very large, and has a strong relationship with the experimental design and the data selection method. It must be ensured that the data are independent of each other. The advantage of the linear regression model is that the establishment of the model does not require much data, that is, a relatively accurate model can be obtained. The disadvantage is that it depends on the model specification. When the model assumptions can not be satisfied, the prediction ability will be very poor (e.g., when the linearity is violated).

Taking into account the advantages and disadvantages of the two methods, we believe that the characteristics of the decision tree model make it easier to be applied to the practical classification problem, and the linear regression model can help to select variables and assist in the construction of the model during the modeling process.

Key words: FIFA 18; ID3 Type decision tree algorithm; data visualization; linear regression; model analysis; model prediction

目　录

决策树和线性回归的实际应用—FIFA 18 数据分析

杜澧泽

一、数据

(一) 数据来源

本数据来自于 kaggle 数据平台的 FIFA 18 More Complete Player Dataset。数据提供者是 Kevin H。

(二) 数据描述及论文思路

1. 数据描述

本数据包含了 FIFA 18 中：

- 每个球员在 185 个方面的能力值；
- 球员信息，例如：年龄，俱乐部，联赛，国籍，薪水以及其他的身体属性；
- 全部的游戏属性，例如：射门和盘带；
- 特殊属性，例如：盘带技巧和国际名望；
- 球员的特质；
- 总体能力，潜力和每一个位置的评价。

本数据一共有 17994 个观测值和 185 个变量，下面是部分数据。

表 1: FIFA 18 数据 (部分))

name	club	age	nationality	overall
Cristiano Ronaldo	Real Madrid CF	32	Portugal	94
L. Messi	FC Barcelona	30	Argentina	93
Neymar	Paris Saint-Germain	25	Brazil	92
L. Suárez	FC Barcelona	30	Uruguay	92
M. Neuer	FC Bayern Munich	31	Germany	92
R. Lewandowski	FC Bayern Munich	28	Poland	91

上面的 FIFA 18 数据（部分）表中我们截取了球员姓名，俱乐部，年龄，国籍，总体能力值数据的前六行，从中我们可以读出球员的一些基础信息，例如：M.Neuer 是国籍德国来自于拜仁慕尼黑俱乐部的总体能力值为 92 的年龄为 31 岁的球员。

2. 论文思路

本篇论文基于 FIFA 18 球员数据，

首先利用 ggplot2 包对数据利用可视化的手段进行直观地分析和考察, 从而发现具有代表性的数据和具有代表性的属性, 并从中选取一家俱乐部（LIVERPOOL 俱乐部）进行针对一支球队的分析，发现一只球队的人员组成和场上位置的信息，并且根据可视化结果对这家俱乐部进行更加深入地分析。之后根据可视化的结果，选取恰当的数据集，为进一步的建立决策树模型和线性回归的模型建立基础；

之后我们利用具有代表性的数据集作为训练集来训练模型，运用 ID3 决策树算法, 构建将球员分类的决策树，并且用与训练集不相关的测试集测试得到的精度。由于在对场上位置与球员属性进行建模时，我们对球员分为了 front，center，back，gk 四部分，而这个划分是具有随机性的，影响了分类的准确度（在划分时，我们根据球员场上位置能力最高的数值去决定，而球员可能在作为前场球员和中场球员在各自的位置能力值同样为最大，这个时候我们随机选取某个位置作为他的分类。例如，球员 A 的 cf 能力值和 cam 能力值同样为最大的 84，这个时候将其划分为前场或者中场球员的概率皆为 0.5）。之后我们将根据得到的精度对模型进行相应的调整和分析；

接下来我们建立用相同的数据建立的线性回归模型，得到四种球员分类和相应球员属性的线性回归模型，并且对模型的显著性和变量的显著性进行分析和选择，再用和之前相同的测试集的数据进行测试；

对用决策树分类以及线性回归得出的精度进行比较，对两种方法的适用场景和优劣进行讨论。

二、数据可视化及分析

下图是用 R 语言的 ggplot2 包得到的球员年龄密度曲线。

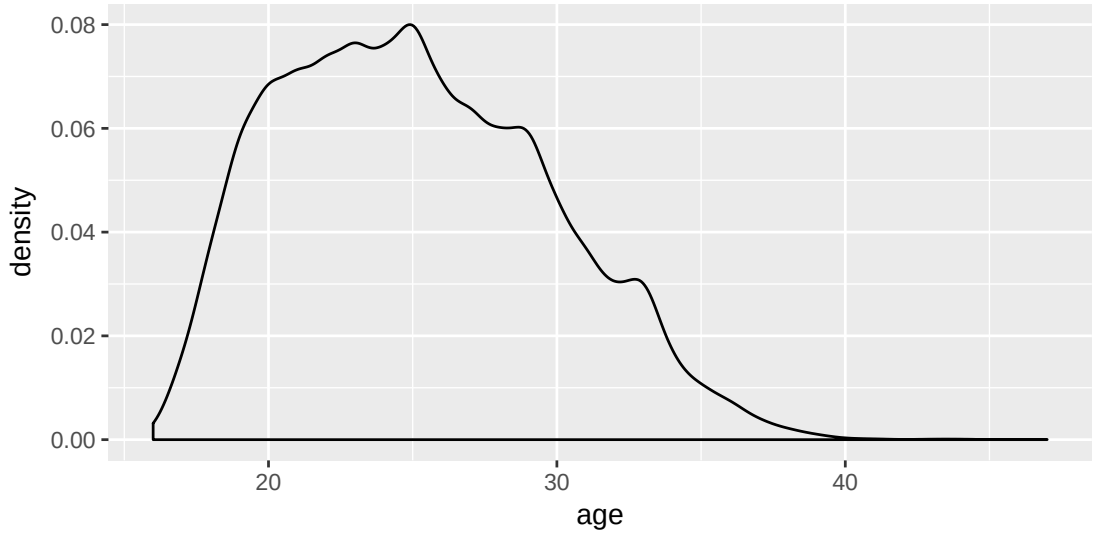

图 1: 年龄密度曲线

如图所示，球员年龄集中在 20-25 岁，年龄最小的球员为 16 岁，年龄最大的超过了 40 岁。

当球员年龄超过 25 岁时，密度曲线急速下降，当球员年龄不足 20 岁时密度曲线急速上升，由此我们可以推断出球员的黄金运动年龄是 20-28 岁。

进一步地，我们想知道年龄和运动能力的相关关系，所以我们画了球员总体能力关于年龄的曲线。

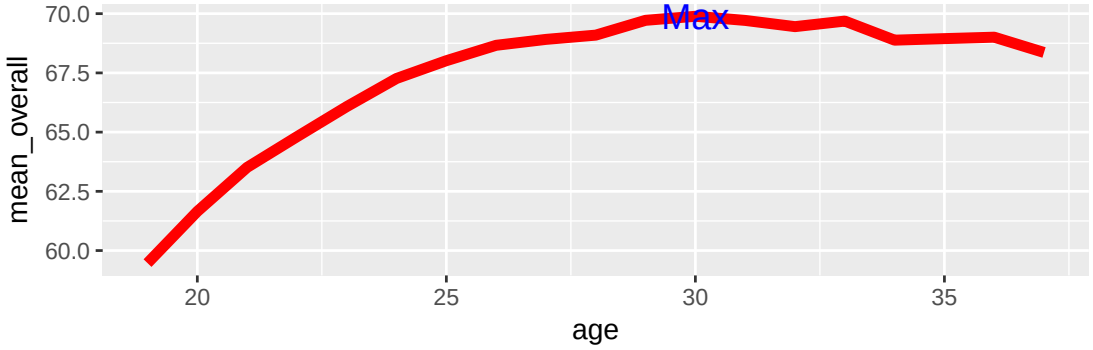

图 2: 总体能力与年龄关系曲线

我们可以看到，能力值在 30 岁之前呈递增趋势，30 岁之后呈递减趋势，但减小的速度较慢，球员总体能力在 30 岁达到巅峰，然而这个结果与图一的峰值有一个延迟，由此我们可以得出结论：在 30 岁仍然活跃在场上的运动员的运动能力比普通球员优秀，他们的运动生命也长于普通球员；此外年轻球员的成长速度比职业生涯末期身体机能衰退的速度要快，因此很多球员在 30 岁之后仍然可以活跃在足球场上。

接下来，我们想知道哪些联赛的明星球员最多（我们将能力值大于 81 的球员划分为明星球员）。所以我们统计出明星球员最多的十大联赛，并用柱状图表示出来。

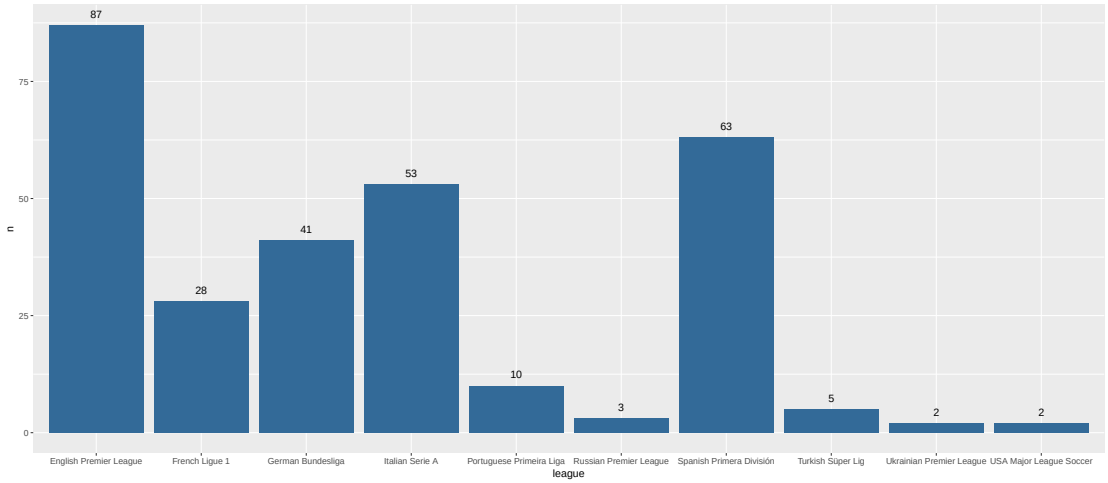

图 3: 十大联赛明星球员数量

我们可以发现 English Premier League 的明星球员最多为 87 人，可以称为世界第一联赛，紧随其后的是 Spanish Primer Division 和 Italian Serie A，分别有 63 和 53 人。

了解了球星在不同联赛的分布后，下面我们想看一下球星的国家分布，看一下世界上哪几个国家盛产足球明星。

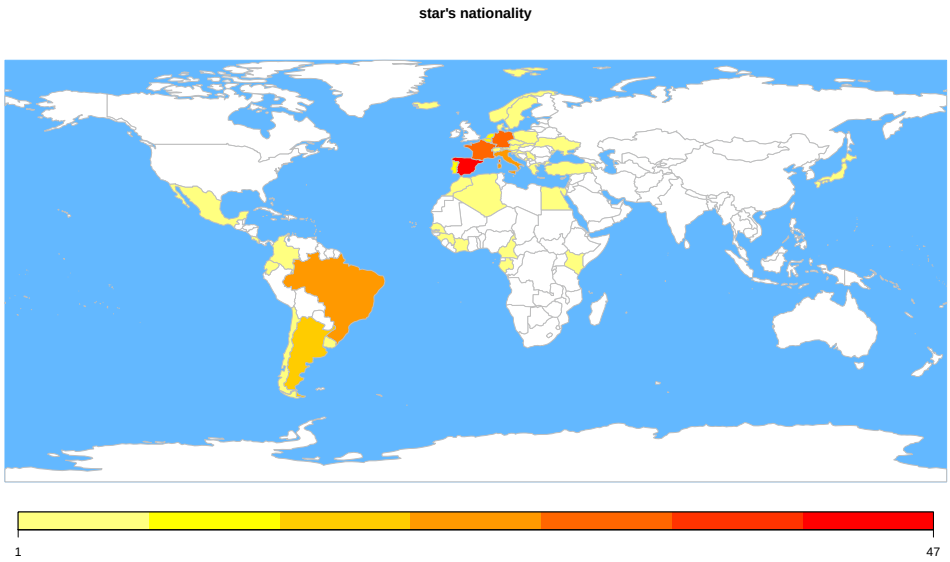

图 4: 球星的国家分布

从图中我们可以看到，这些明星球员大都来自于传统的足球强国，西班牙，德国，法国，巴西，阿根廷是足球人才的产出大国。其中来自于西班牙的球星达到了惊人的 47 人，由此我们也可以理解了西班牙国家队在近些年的国际大赛中的卓越战绩。亚洲方面，日本显得一枝独秀，而中国的球星数量是 0，看来国足想要变强路途还很漫长。

接下来我们感兴趣的是当今世界上十大最顶尖俱乐部的数据，我们按照队内平均能力值的高低决定出了十大顶尖俱乐部。

表 2: 十大俱乐部数据

league	club	count	mean_overall	mean_age
Spanish Primera División	FC Barcelona	24	83	26
Italian Serie A	Juventus	26	82	28
Spanish Primera División	Real Madrid CF	28	80	24
German Bundesliga	FC Bayern Munich	26	79	25
English Premier League	Manchester United	33	78	25
French Ligue 1	Paris Saint-Germain	28	78	24
Italian Serie A	Napoli	27	78	26
Italian Serie A	Inter	24	77	26
Italian Serie A	Roma	27	77	26
Spanish Primera División	Sevilla FC	26	77	26

下图是 FIFA 18 中的十大顶尖俱乐部的平均能力值和球员数量的散点图。其中点的大小代表了球员的数量的多少，点的颜色越浅代表能力值越高，从图中我们可以看到 FC Barcelona 的

平均能力值最高为 83，Manchester United 的球员数最多为 32 人。

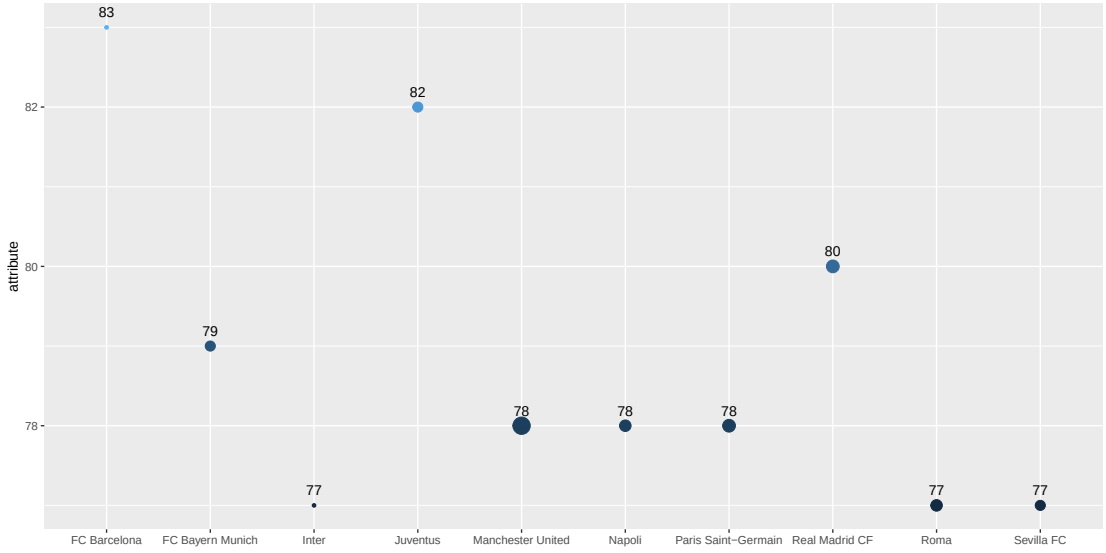

图 5: 十大顶尖俱乐部

接下来我们研究一个足球俱乐部的球员组成的情况，这里我们以 Liverpool 足球俱乐部为例：

下表是 Liverpool 足球俱乐部的数据，包含了球员姓名，年龄，国籍，身价，能力值，场上位置。由于我们对于相同能力值的场上位置（例如 cam 83，cdm 83，cm 83），我们采取了随机选取一个位置的方法，故此分类有比较大的随机性

表 3: liverpool 球员数据

name	age	nationality	eur_value	overall	loc
Coutinho	25	Brazil	56000000	86	lam
S. Mané	25	Senegal	39000000	84	rw
M. Salah	25	Egypt	32500000	83	lw
Roberto Firmino	25	Brazil	33000000	83	cam
J. Matip	25	Cameroon	26500000	83	lcb
A. Lallana	29	England	25000000	83	cam
N. Clyne	26	England	22000000	82	lwb
J. Henderson	27	England	21500000	82	cm
G. Wijnaldum	26	Netherlands	27000000	82	cf
D. Sturridge	27	England	23500000	82	st
D. Lovren	27	Croatia	17500000	81	lcb
S. Mignolet	29	Belgium	12000000	81	gk
E. Can	23	Germany	19500000	80	ldm
L. Karius	24	Germany	14000000	80	gk
A. Oxlade-Chamberlain	23	England	20000000	80	rw
J. Milner	31	England	9500000	80	rdm

name	age	nationality	eur_value	overall	loc
D. Ings	24	England	10000000	76	cf
L. Marković	23	Serbia	10000000	76	rw
Alberto Moreno	24	Spain	8000000	76	rwb
A. Robertson	23	Scotland	8000000	75	lwb
M. Grujić	21	Serbia	5000000	71	cdm
D. Solanke	19	England	3800000	70	lf
A. Bogdán	29	Hungary	1200000	70	gk
J. Flanagan	24	England	2000000	70	rcb
J. Gomez	20	England	3100000	70	cb
D. Ward	24	Wales	2000000	70	gk
C. Brannagan	21	England	1700000	69	lcm
T. Alexander-Arnold	18	England	1900000	69	lwb
O. Ejaria	19	England	1400000	66	lam
B. Woodburn	17	Wales	1100000	65	cf
L. Jones	21	England	400000	62	rcb
H. Wilson	20	Wales	550000	61	rw

由于场上的具体位置过于繁琐,所以我们把场上的位置分为 “gk”, “front”, “center”, “back” 四种，方便我们接下来的数据处理。

- 将 gk 划分为 gk
- 将 cf,lf,ls,lw,rf,rw,rs,st 划分为 front（即以 f,s,w,t 结尾）
- 将 cam,ldm,ram,rcm,rdm,rm,cdm,cm,lm,lam,lcm 划分为 center（即以 m 结尾）
- 将 cb,lcb,lwb,rcb,rwb,rb,lb 划分为 back。（即以 b 结尾）

之后我们统计出 Liverpool 足球俱乐部场上每一个位置的球员数量，并且用柱状图表示出来。

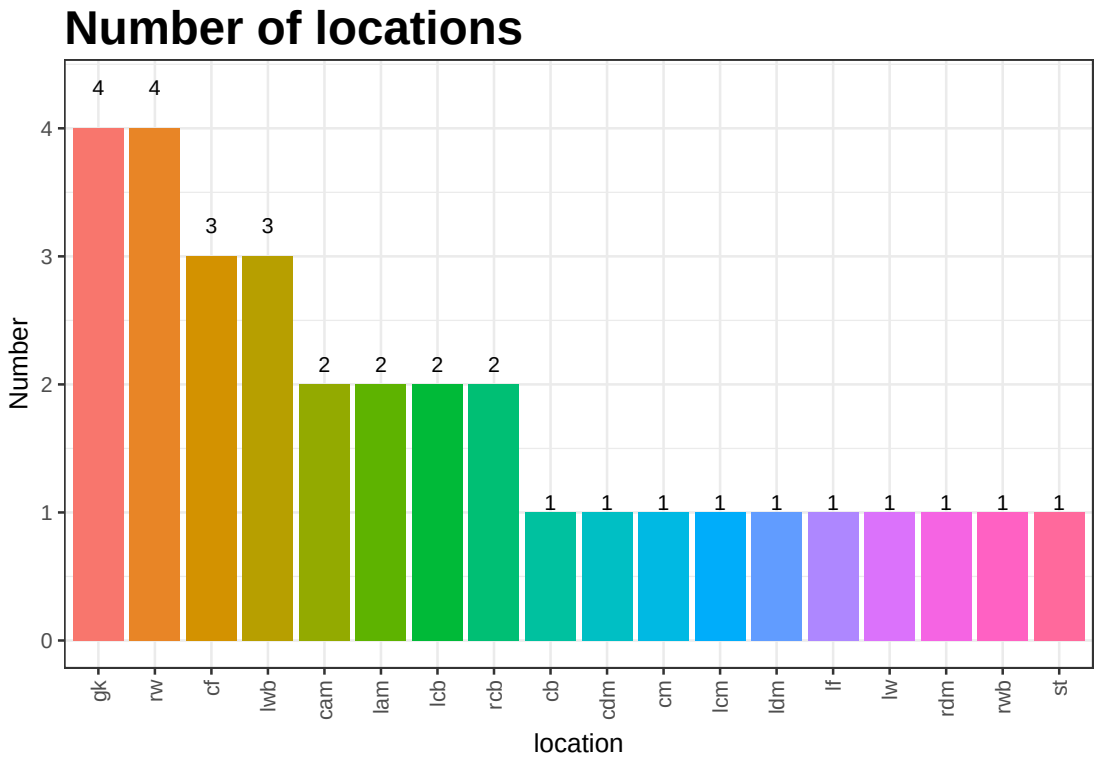

图 6: LFC 球员位置

从图中我们可以看到 Liverpool 足球俱乐部的球员位置构成，我们可以看到前，中，后场球员及门将的比例为 10：9：9：4，我们认为这是一个比较均衡的足球俱乐部球员的配置。以 “w” 结尾的代表的是边锋，我们可以看到 Liverpool 足球俱乐部可以打边锋的球员有 7 个之多，而 “cf” 代表的中锋只有一个。据此我们可以推断出，这家俱乐部的打法会是以边路的速度冲击为主，而没有传统意义上的中锋作为支点。另外我们发现以 “l” 开头，“b” 结尾代表的左后卫人数人数很少，因此这个球队的左路会是他防守时的薄弱环节，一旦有这个位置的球员受伤，能够补充进首发的球员便捉襟见肘。之后我们将根据基于游戏属性构建的决策树模型和线性回归模型重新划分 Liverpool 足球俱乐部的场上球员的位置。

三、决策树模型

(一) 决策树简介及类型

决策树是一种分类器，它代表的是属性与分类之间的一种映射关系。树中的每个非叶子结点代表了一个属性（attribute），根据这个属性将位于此结点的对象划分到他的子节点，通过迭代，直到结点中的所有对象划为一类或满足终止条件结束迭代。通俗地讲，每一个结点相当于一个 “问题”，根据每一个对象回答的 “答案”（即树杈），将其划分到下面的结点进行新一轮的 “问答”。我们建立决策树的最终目的是对未来的数据进行很好的分类，即我们关心的是决策树的预测能力，当用测试集测试时，精度高的决策树更好。常见的决策树有 ID3,C4.5，CART 等，这篇论文将采用 ID3 算法。

(二)ID3 算法

决策树 ID3 算法的核心算法是以信息增益作为变量选择的准则，选择信息增益最大的属性进行决策树的分叉。

1. 下面我们介绍熵（entropy）和信息增益（information gain）的概念

熵（entropy）

熵表示状态的混乱程度, 也可以理解为信息量。我们知道一个 1-100 的数可以用七个 yes/no 的问题来决定，即 $\log(100) = 6.64 \approx 7$，所以对于一个大小为 $|S|$ 的集合 S，我们至少需要做 $\log(|S|)$ 次的迭代，才能做出决定。若 $S = P \bigcup N$, 其中 P 和 N 是两个不相交的集合，并且 $|P| = p, |N| = n$, 我们要决定 S 中的任何一个元素需要做 $(p/(n+p)) * \log(p) + (n/(n+p)) * \log(n)$ 次迭代，这就是我们所说的熵（entropy），用 H(Y) 来表示。

$$H(Y) = \sum_{i=1}^{k} -P(Y = y_i) * \log(P(Y = y_i))$$

条件熵和信息增益（conditional entropy and information gain）

条件熵用 H（Y|X）表示，

$$H(Y|X = v) = \sum_{i=1}^{k} -P(Y = y_i|X = v) * \log(P(Y = y_i|X = v))$$

$$H(Y|X) = \sum_{vvaluesofX} P(X = v) * H(Y|X = v)$$

信息增益用户 I(Y;X) 表示，

$$I(Y; X) = H(Y) - H(Y|X)$$

在每一个结点我们选择可以最大化 I(Y;X) 的属性。

2. 决策树算法

buildtree(examples,attributes,default-label)
if empty(examples) then return default-label
if(examples have same label y) then return y
if empty(attributes) then return majority vote in examples
q=best_attribute(examples,attributes)
tree= create-node with attribute q
foreach value v of attribute q
    v-ex = subset of examples with q==v
    subtree = buildtree(v-ex,attributes-{q},majority-class(examples))
add arc form tree to subtree
return tree

(三) 数据集的选取及决策树算法实现

考虑到本数据的数据量过于庞大，所以我们选择有代表性的数据进行数据分析，因为往往人们的关注点都在优秀的顶级球员身上，故我们将总体能力值大于 80 的球员筛选出来，并组合

他们的 name, club, age, league, height_cm, weight_kg, nationality, eur_value, eur_wage, overall, potential, 及他们的游戏属性例如：射门，盘带，头球精度等作为我们训练决策树的训练集。在这里面我们去除掉来自于 Liverpool 足球俱乐部的球员，因为我们将要用这组数据作为测试集。

为了更好地分类，我们对于每一个属性，根据能力值的大小分为六个层次：

- 能力值小于等于 50：“1”
- 能力值小于等于 60 大于 50 “2”
- 能力值小于等于 70 大于 60 “3”
- 能力值小于等于 80 大于 70 “4”
- 能力值小于等于 90 大于 80 “5”
- 能力值大于 90 “6”

在这里我们将位置从具体的场上位置（cf, cam 等），划分为 front, center, back, gk 四种。

1. 第一种分类方法及测试集

下表是训练集的部分数据，可以看到我们把原先的数据转化成了对于每个属性 1-6 的评级，把分类转化为了 front, center, back, gk 四种。从而可以方便地应用到决策树模型中去。

表 4: 决策树 training set（部分）

gk_diving	gk_handling	gk_kicking	gk_positioning	gk_reflexes	class
1	1	1	1	1	front
1	1	1	1	1	front
1	1	1	1	1	front
1	1	1	1	1	front
6	5	6	6	5	gk
1	1	1	1	1	front
5	5	5	5	5	gk
1	1	1	1	1	front
1	1	1	1	1	center
1	1	1	1	1	front

之后，我们将数据导出为 txt 文件，training set 中共有 391 个观测值，

接下来，我们利用 Liverpool 足球俱乐部的球员数据作为测试集去检验决策树的精度，同样的我们把原始数据转化为可以方便应用决策树模型的形式，下表是部分测试集数据。

表 5: 决策树测试集（部分）

gk_diving	gk_handling	gk_kicking	gk_positioning	gk_reflexes	class
1	1	1	1	1	center

gk_diving	gk_handling	gk_kicking	gk_positioning	gk_reflexes	class
1	1	1	1	1	front
1	1	1	1	1	front
1	1	1	1	1	front
1	1	1	1	1	back
1	1	1	1	1	center
1	1	1	1	1	back
1	1	1	1	1	center
1	1	1	1	1	center
1	1	1	1	1	front

之后，我们将数据导出为 txt 文件，testing set 中共有 34 个观测值，

决策树的实现是在 java 环境下完成的，最终预测的精度为 0.6875，我们把预测的结果与原始分类在下表中列出。

表 6: testing set 原分类与预测分类对比

	predict	original
Coutinho	center	center
S. Mané	front	front
M. Salah	front	front
Roberto Firmino	front	front
J. Matip	back	back
A. Lallana	center	center
N. Clyne	back	back
J. Henderson	center	center
G. Wijnaldum	center	center
D. Sturridge	front	front
D. Lovren	back	back
S. Mignolet	gk	gk
E. Can	center	center
L. Karius	gk	gk
A. Oxlade-Chamberlain	front	front
J. Milner	center	center
D. Ings	center	front
L. Marković	center	front
Alberto Moreno	back	back
A. Robertson	back	back
M. Grujić	center	center
D. Solanke	center	front
A. Bogdán	gk	gk
J. Flanagan	center	back

	predict	original
J. Gomez	center	back
D. Ward	gk	gk
C. Brannagan	center	center
T. Alexander-Arnold	center	back
O. Ejaria	back	center
B. Woodburn	center	front
L. Jones	gk	back
H. Wilson	center	front

表格中从上至下可以看到，上面的数据分类准确率很高，下面的数据分类准确率较低，根据现实的情况，上面的球员是主力球员，而下面的球员是年龄比较小的青训球员或者是替补球员，意味着对于能力值比较高的球员的分类效果及预测能力比较好。而由于我们的训练集是选取的能力值大于 80 的顶尖球员，也就是我们得到的决策树对于顶尖球员的预测能力比较好。根据这些发现，我们可以大胆地猜测，个人球风比较鲜明的球员（比较容易被分类）比较容易在一个球队打主力，而球员的成长过程也是一个不断找到最适合自己的球风的过程，也是对于年轻球员的一个启发，找到自己的特色并不断进步才是成长为球星的正确道路。

2. 第二种分类方法

由此，我们尝试去扩大训练集，将范围扩大到能力值大于 60 的球员，我们相信这时的决策树的预测能力将大大提高，对于能力值不高的球员也可以较为准确的预测。

下表是我们得到的新的训练集的部分数据。

表 7: 第二颗决策树训练集（部分）

gk_diving	gk_handling	gk_kicking	gk_positioning	gk_reflexes	class
1	1	1	1	1	front
1	1	1	1	1	center
1	1	1	1	1	front
1	1	1	1	1	front
6	5	6	6	5	gk
1	1	1	1	1	front
5	5	5	5	5	gk
1	1	1	1	1	front
1	1	1	1	1	center
1	1	1	1	1	front

之后，我们将数据导出为 txt 文件，这个时候训练集中包含 14370 个观测值，

这个时候，我们的预测精度大大提升为 0.90625。

下面我们通过一个表格把两次的预测结果与原始分类放在一起来比较一下两种分类的不同。

表 8: 第二颗树 testing set 原分类与预测分类对比

	new_predict	old_predict	original
Coutinho	center	center	center
S. Mané	front	front	front
M. Salah	front	front	front
Roberto Firmino	front	front	front
J. Matip	back	back	back
A. Lallana	center	center	center
N. Clyne	back	back	back
J. Henderson	center	center	center
G. Wijnaldum	center	center	center
D. Sturridge	front	front	front
D. Lovren	back	back	back
S. Mignolet	gk	gk	gk
E. Can	center	center	center
L. Karius	gk	gk	gk
A. Oxlade-Chamberlain	front	front	front
J. Milner	center	center	center
D. Ings	front	center	front
L. Marković	center	center	front
Alberto Moreno	back	back	back
A. Robertson	back	back	back
M. Grujić	center	center	center
D. Solanke	front	center	front
A. Bogdán	gk	gk	gk
J. Flanagan	back	center	back
J. Gomez	back	center	back
D. Ward	gk	gk	gk
C. Brannagan	center	center	center
T. Alexander-Arnold	center	center	back
O. Ejaria	center	back	center
B. Woodburn	center	center	front
L. Jones	back	gk	back
H. Wilson	front	center	front

我们可以看到，利用新范围的训练集得到的决策树更好地预测了能力值较低的球员的分类，这个时候依然分类与原始分类有偏差的球员，我们认为这是“万金油”类型的球员，即一人可以胜任场上的多个角色，这也是符合我们对于足球世界的认识的，通过查阅相关比赛资料，我们可以看到，原始分类与我们决策树得到的分类不同的球员，例如：T. Alexander-Arnold 在现实中多打右边后卫（back），但也有打后腰（center）的比赛, A. Oxlade-Chamberlain 现在打中前卫（center）但是在转会 Liverpool 足球俱乐部之前，多打边锋（front）。

3. 第三种分类方法

最后我们把训练集选取为除去 Liverpol 俱乐部球员的全部球员，考虑到 Liverpol 俱乐部没有能力值不足 60 的球员，所以我们认为这次预测的精度不会有很大变化。

表 9: 第三颗决策树训练集（部分）

gk_diving	gk_handling	gk_kicking	gk_positioning	gk_reflexes	class
1	1	1	1	1	front
1	1	1	1	1	center
1	1	1	1	1	front
1	1	1	1	1	front
6	5	6	6	5	gk
1	1	1	1	1	front
5	5	5	5	5	gk
1	1	1	1	1	center
1	1	1	1	1	center
1	1	1	1	1	front

之后，我们将数据导出为 txt 文件，这个时候训练集中包含 17962 个观测值，

这个时候，我们的预测精度降低为 0.81250，我们把三种分类方法得到的预测结果与原始分类进行比较。

表 10: 三颗树训练集原分类与预测分类对比

	third_predict	second_predict	first_predict	original
Coutinho	center	center	center	center
S. Mané	front	front	front	front
M. Salah	center	front	front	front
Roberto Firmino	center	front	front	front
J. Matip	back	back	back	back
A. Lallana	center	center	center	center
N. Clyne	back	back	back	back
J. Henderson	center	center	center	center
G. Wijnaldum	center	center	center	center
D. Sturridge	front	front	front	front
D. Lovren	back	back	back	back
S. Mignolet	gk	gk	gk	gk
E. Can	center	center	center	center
L. Karius	gk	gk	gk	gk
A. Oxlade-Chamberlain	front	front	front	front
J. Milner	center	center	center	center
D. Ings	front	front	center	front
L. Marković	center	center	center	front

	third_predict	second_predict	first_predict	original
Alberto Moreno	back	back	back	back
A. Robertson	back	back	back	back
M. Grujić	front	center	center	center
D. Solanke	front	front	center	front
A. Bogdán	gk	gk	gk	gk
J. Flanagan	back	back	center	back
J. Gomez	back	back	center	back
D. Ward	gk	gk	gk	gk
C. Brannagan	center	center	center	center
T. Alexander-Arnold	center	center	center	back
O. Ejaria	center	center	back	center
B. Woodburn	center	center	center	front
L. Jones	back	back	gk	back
H. Wilson	front	front	center	front

很明显，在预测精度上，第二棵树大于第三棵树大于第一棵树。从而我们认为，决策树模型对于训练集的要求比较高，当训练集数据对于新数据的代表性比较好时，预测的精度会很高，但如果有过多的训练数据会干扰模型训练的精度，所以训练数据并不是越多越好，提醒我们在构建决策树模型时对于训练数据的选取非常重要。

四. 从决策树算法出发的扩展

1. 用贪心算法进行剪枝

我们知道运用决策树算法时经常需要面对一个变量数很大的原始训练集数据, 由于我们对于变量之间的关系在训练之前并不了解, 而由于为了追求训练效果, 往往使用大量数据训练, 这个时候就难免会出现过拟合的问题, 即模型过于依赖现有的训练集, 对于未来数据的预测的偏差增大, 决策树的效率会降低很多。

针对这种情况，我们采用贪心算法进行剪枝，即对于每一个除了根节点外的所有内部节点，都进行“剪枝”的尝试。对于同一个调整集，选择进行“剪枝”操作后预测精度提升最多的“新决策树”作为我们下一轮迭代的初始模型，迭代当“剪枝”后预测精度不再增大时停止，得到我们想要的决策树模型。通过此算法，可以很好的解决过拟合的问题，但此算法的时间复杂度非常大，当数据量很大，变量数目很多时，计算时间会很长。

用贪心算法进行剪枝的算法实现如下：

Prune(tree T,TUNE set)
1.Compute T's accuracy on TUNE; call it Acc(T)
2.For every internal node N in T, do:

a)New tree T_n=copy of T, but prune the subtree under N

b)N becomes a leaf node in T_n. The class is the majority vote of TRAIN examples reaching N

c)Acc(T_n)=$T_n's$ accuracy on TUNE

3.Let $T*$ be the tree with the largest Acc(), set T=$T*$
4.Repeat from step 1 until no more improvement
Return T

2. 随机森林算法

随机森林 (random forest) 是另外一种解决过拟合问题的算法, 即将多颗决策树结合在一起, 预测值取多颗决策树结果中的大多数。随机森林的两个主要想法是：

1.bagging: 随机地在数据集中选取一部分作为训练集

2.randomized node optimization: 每次当一个结点被分裂时，随机地从可选的属性中选取一部分做为当前结点可选的属性的集合。

决策树算法实现如下：For each tree

1. 训练集的选取：从 N 个可选的 examples 中有放回地选取 n 个作为训练集。

2. 在构建决策树的过程中，对于每个结点，随机地从 M 个可选的属性中随机选取一个大小为 m 的子集（m<<M）

3. 选取信息增益最大的属性作为划分当前结点的依据。

四、线性回归模型

(一) 线性回归模型思路及假设

线性回归是一种统计过程，用于通过线性模型描述变量之间的关系时，从自变量预测因变量的值。线性回归方程可以写为：

$$\vec{y} = m\vec{x} + \vec{b}$$

其中 m 是回归线的斜率，b 是回归线的 Y 截距。在统计学中，线性回归是一种估算一个变量 y 在给定其他变量 x 的值的条件下的期望值的方法，依赖的自变量可能是标量或向量。如果自变量是一个向量，则说明是多重线性回归。

线性回归通常可以表示为，$y = \alpha + \beta x + \epsilon$, ϵ 被称为误差项，他是模型随机性的来源。通常我们假设误差项满足均值为零的正态分布，并且对于每一个观测值得误差项独立同分布。并且认为 ϵ 与 x 是无关的。

我们将建立四个线性回归模型, 分别是,

- front 能力值作为响应变量, 球员属性作为自变量的线性回归模型
- center 能力值作为响应变量, 球员属性作为自变量的线性回归模型
- back 能力值作为响应变量, 球员属性作为自变量的线性回归模型
- gk 能力值作为响应变量, 球员属性作为自变量的线性回归模型

我们认为能力值在 0-100 内连续变化, 每一个球员属性有 6 个层次 (与之前相同), 假设误差项满足独立同分布于均值为零, 方差为 σ^2 的正态分布。我们的预测函数为比较每一个球员四项能力值的大小，将最大值所处的位置作为球员分类划分的依据。例如球员 A 的（front，center,back,gk）能力值分别为（80，70，60，30），则我们将球员 A 划分为 front。

(二) 构建线性回归模型与结果分析

我们将选取与决策树模型相同的三组训练集和相同的测试集，这个时候我们不再对球员属性进行划分而是使用其原始的数值数据。

我们将首先对数据进行处理，分别将，球员所有属于 front 的场上位置能力值的最大值作为 front 能力值，球员所有属于 center 的场上位置能力值的最大值作为 center 能力值，球员所有属于 back 的场上位置能力值的最大值作为 back 能力值，球员所有属于 gk 的场上位置能力值的最大值作为 gk 能力值。

1. 第一种线性回归模型

我们首先用第三组训练集来训练模型，即使用全部数据来作为训练集。

表 11: 线性回归模型训练集 3（部分）

crossing	ball_control	long_shots	gk_diving	front	center	back	gk
85	93	92	7	92	89	66	0
77	95	88	6	92	92	62	0
75	95	77	9	89	88	64	0
77	91	86	27	88	87	68	0
15	48	16	91	0	0	0	92
62	89	83	15	88	84	61	0
17	42	12	90	0	0	0	90
80	92	82	11	88	88	64	0
85	89	90	10	81	87	78	0
68	85	82	5	87	81	55	0

接下来我们进行线性模型的搭建，分别对 front，center，back，gk 关于球员属性建模。

对于 front 与球员属性模型，我们可以得到对模型显著性的 F 检验的 p-value: $< 2.2e\text{-}16$，因此我们可以认为我们得到的模型是显著的，而对于单个系数 T 检验得到的 p-value，得到 sliding_tackle（0.156618），marking（0.766923），volleys（0.939927）是不显著的，即这些因素对于模型的影响是很小的，我们可以更新模型使得模型不包含这几个自变量，但考虑到现实足球世界里，每一个球员属性都对一个球员确定其场上位置有影响，去掉变量的代价太大，所以我们选择保留这些不显著的自变量。

同样地，对于 center 与球员属性模型的显著性检验得到的 p-value: $< 2.2e\text{-}16$，并且 Adjusted R-squared 等于 0.9868，可以说模型对于数据的线性拟合度非常好，我们可以看到，在上一个模型的 t 检验中不显著的 sliding_tackle（$< 2e\text{-}16$）在这个模型里是显著的，可以说明铲球对于一个中场球员更为重要，而前场球员的这项能力便显得不是那么有用了。同样的，对于本模型中不显著的自变量我们保留他在模型中，因为删掉一个自变量可能对模型的准确度造成极大的影响，而保留其的影响是微乎其微的。

对于第三个模型，我们 summary 的结果与前两个模型的结果非常相似，故我们认为第三个模型符合我们的要求。

第四个模型与上面三个模型一样，值得我们信赖。

我们将模型得到的结果与预测值呈现在下面的表格中，

表 12: 线性回归模型预测结果

original class	front	center	back	gk	predict class
center	83.4304229	81.0209652	61.6920567	-0.97212611	front
front	82.4717441	78.2045262	60.7620935	0.07951481	front
front	81.5632392	78.4242696	64.5798623	1.71237272	front
front	84.6249890	82.5539246	66.3264632	-2.90898872	front
back	64.4360135	76.6793399	80.4821742	-0.11923898	back
center	80.8724825	82.1279082	74.5117504	-1.60703707	center
back	73.7526629	76.7595169	80.4587893	-0.73844897	back
center	77.2406530	81.6100803	80.5121154	-1.86439404	center
center	80.9428473	81.5702194	74.8944375	-1.31568218	center
front	81.4150950	77.0529674	53.4752085	-0.99961616	front
back	55.5434394	71.7122049	78.5101451	0.98689360	back
gk	0.3721218	1.3524625	-0.2035219	79.34640996	gk
center	74.9051293	79.2246522	79.2129320	-0.81849481	center
gk	-2.7289015	-2.6470988	-4.6307492	81.56731703	gk
front	77.4020405	76.2909037	66.8891493	-0.66903545	front
center	82.0342409	88.2344053	85.9135854	-11.68151681	center
front	76.0148924	72.5247660	55.8494726	0.11305417	front
front	76.4904575	74.6427666	60.9653165	-2.29154766	front
back	73.7440902	73.9760045	73.9966273	-1.00131360	back
back	71.6734069	73.9855920	74.9020634	-1.70295401	back
center	70.0143531	70.6906667	70.1425818	-0.31916523	center
front	69.9267330	67.8971643	48.6350372	0.33289552	front
gk	-2.1285373	-4.3846144	-1.2930135	70.91701305	gk
back	59.5811725	66.5244201	70.2154719	0.70643278	back
back	58.1182996	66.6035201	70.6915974	-1.55117595	back
gk	0.7083964	0.2946611	0.2110375	69.71703697	gk
center	66.1965297	69.3596664	67.3442441	-1.39760433	center
back	66.1284474	68.4138433	68.0797583	-0.57274446	center
center	64.1683335	67.7549871	58.2250015	1.53192898	center
front	65.0591885	62.9943639	45.6193540	-0.10927153	front
back	39.9455210	51.6103506	57.3311562	2.12095627	back
front	61.2033210	58.8702888	48.3083043	-0.67168629	front

我们用第三种训练集得到的测试精度为 0.9375，这个精度是目前我们得到的最高的精度。可以看到预测的结果与球员能力没有偏向，两个预测失败的球员一个能力值很高，一个能力值很低。这个时候对模型的拟合和预测都非常符合我们的要求，可以看到对于简单模型，线性回归可以发挥非常好的作用。同时我们也可以发现对连续的数值型的变量（即回归问题），线性回归更充分的利用了数据，而决策树模型只能离散地来考虑问题，当把连续型的数据转化为离散的数据时，我们对于数据掌握的信息自然损失了很多，模型的预测能力也因而降低。

2. 第二种线性回归模型

下面我们用第二种训练集来训练模型，即能力值大于 60 的全部球员。

表 13: 线性回归模型训练集 2（部分）

crossing	ball_control	long_shots	gk_diving	front	center	back	gk
85	93	92	7	92	89	66	0
77	95	88	6	92	92	62	0
75	95	77	9	89	88	64	0
77	91	86	27	88	87	68	0
15	48	16	91	0	0	0	92
62	89	83	15	88	84	61	0
17	42	12	90	0	0	0	90
80	92	82	11	88	88	64	0
85	89	90	10	81	87	78	0
68	85	82	5	87	81	55	0

接下来我们用得到的训练集进行线性模型的搭建，

表 14: 线性回归模型预测结果

original class	front	center	back	gk	predict class
center	83.2127391	80.857220	61.44031939	-0.84853830	front
front	82.2439607	77.783677	60.43552030	0.17161056	front
front	81.3625821	77.829098	64.15197342	1.79132224	front
front	84.3119618	81.935202	65.78743701	-2.84174366	front
back	64.1620951	76.402107	80.17834167	-0.06176265	back
center	80.5308817	81.386037	73.94028875	-1.62564490	center
back	73.4770365	76.015877	79.88540492	-0.61080580	back
center	76.9230417	80.900366	79.88662683	-1.82219461	center
center	80.7772192	80.785751	74.37450541	-1.16904423	center
front	81.1979780	76.685970	53.16766973	-0.99408896	front
back	55.2175149	71.460150	78.22006643	1.05730463	back
gk	0.6416545	1.335836	-0.08693479	79.12232589	gk
center	74.6378557	78.567586	78.64043161	-0.72539684	back
gk	-2.3238557	-2.188569	-4.12611606	81.13906166	gk
front	77.3715115	76.077320	66.76690716	-0.54877792	front
center	81.8826571	87.812964	85.57692507	-11.50067004	center
front	75.8668737	72.126520	55.62126268	0.24033294	front
front	76.4255591	74.419341	60.92105055	-2.14240129	front
back	73.5418149	73.519592	73.59785392	-0.96640377	back
back	71.5163824	73.374004	74.57917879	-1.66115594	back
center	69.9553265	70.464765	69.93703857	-0.30088142	center

original class	front	center	back	gk	predict class
front	69.8493416	67.976086	48.69653202	0.33523912	front
gk	-1.5799581	-3.979394	-0.73659134	70.71304749	gk
back	59.4332329	66.419449	70.08415895	0.66425906	back
back	58.0381868	66.669008	70.63742221	-1.50924173	back
gk	1.3253735	1.203635	1.04444516	69.34954698	gk
center	66.2654952	69.812133	67.60247683	-1.43218110	center
back	66.1920375	68.541537	68.18867790	-0.64254718	center
center	64.1799354	68.162025	58.44064548	1.57201506	center
front	65.3342657	63.792318	46.24957068	-0.05325835	front
back	40.3092882	52.562359	58.08865365	2.20467948	back
front	61.5329126	59.522700	48.90312945	-0.58947473	front

这个时候得到的预测精度为 0.90625。三个预测失败的例子其中两个和第三种训练集的结果一样，由此我们可以推断出线性回归模型对于数据的依赖程度比较低，与决策树模型对于顶尖球员预测能力较好的情况，可以比较好的预测更为一般的球员而不被球员类型限制。线性回归模型的预测精度与有效数据量的大小成正比，因为我们的原始数据均来自于真实数据，所以不存在outliers。而决策树模型对于这个训练集的分类能力最强，因此在实际应用中，需要我们“有效率”的去选择训练集，这个时候只需要不是非常大的训练集就可以得到用决策树算法得到的令人满意的分类器。

3. 第三种线性回归模型

下面我们用第一种训练集来训练模型，即用全部能力值大于 80 的球员数据。

表 15: 线性回归模型训练集 1（部分）

crossing	ball_control	long_shots	gk_diving	front	center	back	gk
85	93	92	7	92	89	66	0
77	95	88	6	92	92	62	0
75	95	77	9	89	88	64	0
77	91	86	27	88	87	68	0
15	48	16	91	0	0	0	92
62	89	83	15	88	84	61	0
17	42	12	90	0	0	0	90
80	92	82	11	88	88	64	0
85	89	90	10	81	87	78	0
68	85	82	5	87	81	55	0

接下来我们用得到的进行线性模型的搭建，

表 16: 线性回归模型预测结果

original class	front	center	back	gk	predict class
center	83.773927	83.823847	63.299039	-0.21598016	center
front	82.412123	80.303866	62.233162	0.44945870	front
front	81.875135	79.355957	65.820866	1.65097867	front
front	84.996268	84.066996	66.848160	-2.68181356	front
back	63.415757	75.706407	79.509661	1.95874595	back
center	81.076884	82.662284	75.179155	-1.64807048	center
back	74.096824	77.877404	81.018853	-0.32612223	back
center	76.732121	81.563497	79.986489	-0.27819109	center
center	80.851746	81.462147	74.840403	-0.56055216	center
front	81.521986	78.244372	54.272793	0.72488201	front
back	54.022485	73.040760	78.661552	2.44617123	back
gk	2.741086	2.781067	2.133477	78.68552198	gk
center	74.417355	79.637502	78.695449	0.43946251	center
gk	1.878928	3.086990	1.530861	78.39370010	gk
front	77.826654	78.662624	68.363986	0.49878672	center
center	81.020671	88.260256	84.971799	-8.11972598	center
front	75.353102	73.076214	56.143304	3.06809160	front
front	76.525948	75.958663	62.533270	0.24874750	front
back	74.156480	76.058473	75.265942	-0.57672226	center
back	71.894554	75.473405	76.116253	-0.82945181	back
center	70.968743	72.495612	71.116520	0.55252724	center
front	69.942022	70.544340	50.561853	3.17383044	center
gk	2.177197	1.143500	3.464976	71.98995708	gk
back	60.589971	70.463519	71.977480	-0.02215806	back
back	59.587019	70.647350	72.545541	-0.77306046	back
gk	6.014731	6.878179	5.475841	68.40654472	gk
center	66.971453	72.390686	68.963846	1.11939149	center
back	67.537733	70.957362	69.978856	0.88546331	center
center	65.563092	72.968429	61.677673	1.30335046	center
front	65.738304	67.742597	48.753471	4.05887123	center
back	41.827194	55.860778	58.698366	7.90721270	back
front	63.980135	64.498457	52.335504	2.42738632	center

这个时候预测的精度为 0.8125。对于第一个训练集只为能力值大于 80 的顶尖球员，可以看到这个时候线性回归模型对于顶尖球员的预测能力比较好，之前两个模型预测错误的顶尖球员在这个模型中预测正确，因此，线性回归模型也需要选取有代表性的数据作为训练集，但并没有决策树模型表现出来的对于数据的依赖程度大。同时我们可以发现，在数据量不大的情况下，线性回归得到的模型预测能力比决策树得到的要强很多，因为当数据大于等于两个时，我们就可以用最小二乘法拟合出一条直线方程去最小化残差平方和；而决策树的搭建需要大量的数据去

训练才能往下延伸成为一颗可以用来分类的树。

4. 线性回归模型总结及优缺点

可以看到对于本论文的数据拟合的模型，线性回归的表现非常好，这是因为 FIFA 18 这个数据集并不大，并且因变量与自变量之间有很强的线性相关关系，我们知道线性回归模型是一种简单的模型，因此在处理数据量不大，模型比较简单的情况，我们更倾向于运用线性回归解决问题，这也是著名的奥卡姆剃刀理论，当两种模型的效力相近时，我们更愿意使用，或者是更佳的选择是简单的模型。但是，模型简单也是线性回归的一大缺陷，因为在实际生活中的数据往往不存在直观的线性关系，又或者数据里有很多 outliers，都会极大地影响模型的准确性，而对于变量的类型是 factor 的分类问题，线性回归也不能很好地建立模型。而当我们错误地使用线性回归模型到不适合它的数据上（不满足线性，存在 outlier）我们会发现，模型的残差平方和会非常大，另一方面也对原始数据发生了过拟合，预测的效果会大大降低。针对这个情况，我认为线性回归模型在进行数据预处理，或者进行数据清洗时是非常有用的，它可以发现模型中异常点的出现以及来检验模型的假设是否满足。

五、两种模型比较

表 17: 两种模型预测精度比较

	训练集 1	训练集 2	训练集 3
决策树	0.6875	0.90625	0.8125
线性回归	0.8125	0.90625	0.9375

我们可以看到决策树的预测结果对训练集的选取有很大的关系，当训练集为能力值大于 60 的全部球员时的预测精度最高，但当训练集过大或过小时，预测精度都会相应的下降，当选取的训练集为能力值大于 80 的全部球员时明显对于顶尖球员的预测精度更高。而线性回归模型对于整体的预测精确度对于训练集没有明显的偏向，当训练集的数据量越大时预测的精度越高。因为线性回归的预测情况很大程度基于模型的选择，在这里我们选择了线性模型很好的拟合了原始球员属性数据的情况，但如果我们的数据不符合线性模型，这个时候的预测结果会出现较大的偏差；而决策树的模型是基于数据，当数据选取的比较合理的时候，得到的预测结果的精度会大大提升，但决策树往往伴随着过拟合的问题，即有些球员属性在帮助我们分类的时候没有用途，反而“拖后腿，混淆视听”了，我们称这种属性为噪音，为了消除这种情况，我们可以使用“修建树杈”的算法，得到精度最优的数，也可以通过构造随机森林消除过拟合带来的影响。

附　录

决策树实现（java）

```
DecisionTreeImpl(DataSet train) {
  this.labels = train.labels;
  this.attributes = train.attributes;
  this.attributeValues = train.attributeValues;
  root = buildtree(train.instances,train.attributes,null,null);
}
DecTreeNode buildtree(List<Instance> instances,List<String> list,List<Instance> parentinstances,String parentAttributeValue)
    DecTreeNode tree = new DecTreeNode(null, null, null, false);
    if(instances.isEmpty()) {
        tree = new DecTreeNode(majorityLabel(parentinstances),null,parentAttributeValue,true);
    }
    else if(sameLabel(instances)) {
      tree = new DecTreeNode(instances.get(0).label,null,parentAttributeValue,true);
  }
  else if(list.isEmpty()) {
      tree = new DecTreeNode(majorityLabel(instances),null,parentAttributeValue,true);
  }
  else {
  double info[]=new double[list.size()];
  int index = 0;
  for(int i =0 ; i<list.size() ; i++) {
      info[i]=InfoGain(instances,list.get(i));
  }
  double max = info[0];
  for(int i=0 ; i<info.length; i++) {
        if(max<info[i]) {
           max=info[i];
           index= i;
        }


    }
    String bestattribute = list.get(index);
    int n =getAttributeIndex(bestattribute);
    tree = new DecTreeNode(majorityLabel(instances),bestattribute,parentAttributeValue,false);
    List<String> values = attributeValues.get(bestattribute);
        for(int i =0;i<values.size();i++) {
        List<Instance> temp = new ArrayList<Instance>();
        for(int j =0;j<instances.size();j++) {
            if(instances.get(j).attributes.get(n).equals(values.get(i)))
            temp.add(instances.get(j));
        }
            List<String> l = new ArrayList<String>(list);
            l.remove(bestattribute);
             tree.addChild(buildtree(temp,l,instances,values.get(i)));
    }
 }
     return tree;
}

  boolean sameLabel(List<Instance> instances){
       for(int i =0 ; i<instances.size()-1 ; i++) {
          if(!instances.get(i).label.equals(instances.get(i+1).label))
                  return false;
      }
      return true;
  }
  String majorityLabel(List<Instance> instances){
      int i=0;
      int j=0;
      int p=0;
      int q=0;
      for(int k = 0; k <instances.size();k++) {
        if(instances.get(k).label.equals("front"))
```

```
            i++;
        else if(instances.get(k).label.equals("center")){
            j++;
        }
        else if(instances.get(k).label.equals("back")) {
            p++;
        }
        else
            q++;
    }
      if(i>=j&i>=p&i>=q)
         return "front";
      else if(j>i&j>=p&j>=q)
          return "center";
      else if(p>i&p>j&p>=q)
          return "back";
      else
          return "gk";
}
  double entropy(List<Instance> instances){
      double i=0;
      double j=0;
      double p=0;
      double q=0;
      double h=0;
      for(int k = 0; k <instances.size();k++) {
            if(instances.get(k).label.equals("front"))
                i++;
            else if(instances.get(k).label.equals("center")){
                j++;
            }
            else if(instances.get(k).label.equals("back")) {
                p++;
            }

           else
               q++;
        }
      if(i==0|j==0|p==0|q==0)
          h= 0;
      else
          h= (-i/(i+j+p+q))*Math.log(i/(i+j+p+q))/Math.log(2)+(-j/(i+j+p+q))*Math.log(j/(i+j+p+q))/Math.log(2)+
             (-p/(i+j+p+q))*Math.log(p/(i+j+p+q))/Math.log(2)+(-q/(i+j+p+q))*Math.log(q/(i+j+p+q))/Math.log(2);
      return h;
  }
  double conditionalEntropy(List<Instance> instances, String attr){
      List<String> values = attributeValues.get(attr);
      int k=getAttributeIndex(attr);
      int times[] = new int[values.size()];
      double h[] = new double[values.size()];
      double condentropy = 0;
      List<Instance> temp = new ArrayList <Instance>();
      for(int j=0;j<values.size();j++) {
          times[j]=0;
      }
      for(int j = 0;j<values.size();j++) {
          for(int i=0;i<instances.size();i++) {
              if(instances.get(i).attributes.get(k).equals(values.get(j))) {
                  times[j]++;
                  temp.add(instances.get(i));

              }
          }
          h[j]=entropy(temp);
          temp.clear();
      }
```

```
    for(int j =0;j<values.size();j++) {
        condentropy=condentropy+h[j]*times[j]/instances.size();
    }
    return condentropy;
}
double InfoGain(List<Instance> instances, String attr){
    return entropy(instances) - conditionalEntropy(instances,attr);
}
```

致　谢

本论文是在我的导师武颖老师的指导下完成的。导师渊博的专业知识，严肃的科学态度，严谨的治学精神，诲人不倦的高尚师德都对我产生了深远的影响。从课题的选择到项目的最终完成，武颖老师都始终给予了我悉心的指导。在此，谨向武颖老师致以我最诚挚的谢意和最衷心的感谢！