目　录

1 研究背景

DotA，这款被无数玩家奉为“信仰”的游戏，如今已经走过了 15 个年头。从 Eul 开始，到 Guinsoo，再到 IceFrog，经过历代作者的努力，DotA 已经发展成最具影响力的电子竞技项目之一；从 Defense of the Ancients 到 DotA2，这款古老而又全新的游戏至今依旧保持着它的活力。DotA 是英文 Defense of the Ancients 的简称，可译作“遗迹守卫战”，通常我们读作“倒塔”，既是 DotA 的中文发音，又很好的说明了游戏的本质——守护己方的防御塔，干掉对手的核心建筑。为了到达摧毁对方的远古遗迹的目的，必须在规则的限制内想尽一切办法。玩家可以通过在战斗中杀死对方小兵、英雄以及中立怪物获得金钱和经验来使自己的英雄升级。更多的经验能使英雄的技能增强或属性增加，而金钱则能在商店中购买更具威力的装备。与常规的对战游戏不同的是，每个玩家仅需要选择一个英雄，了解其技能的优劣，通过控制该英雄来赢得胜利，这也是 DotA 的地图特色之一。

不同的英雄、不同的开局模式、不同的分路选择、不同的核心策略，你会遇到不同性格的玩家，怎么去和不善言谈的玩家沟通，怎样纠正其他玩家的错误思想，思路不一致时如何统一目标使五个人团结一心，面对胡搅蛮缠的玩家要如何处理，这些主观客观的因素都会与比赛的输赢有着或多或少的关系。DotA 中的英雄有上百个，每个英雄都有着自己专门的属性与技能，那么我们就会产生疑问：不同的英雄是否与比赛的输赢有着相关性？如果有，组织怎样的战队赢得可能性比较大呢？还有以及确定了比赛战队的英雄选择，我们是否可以预测比赛的胜利概率……这些疑问也是我们研究主题的由来。

2 数据预处理

我们选择 dota2 的数据，该数据含有训练集 92649 条，测试集 10294 条，变量一共 117 个，其中因变量为队伍的胜负情况，是个二分类变量。自变量中有 113 个变量表示英雄选择情况，是分类指标；另外三个变量分别为游戏分区、游戏模式以及游戏类型，也是分类指标。

由于该数据集的因变量是分类变量，因此我们需要建立分类模型，并且自变量也全是分类变量，所以在数据处理中将其全部转为因子型变量，该类变量在 R 软件中构建逻辑回归，可以自动拆分为虚拟变量，方便我们建模研究。在数据处

理时，游戏分区的代号十分繁杂，因此查询说明文件，有些代号指的是一个国家，因此重新安装国家为界将部分代号汇总，形成新的国家地区变量，删掉原先的分区变量；游戏模式一共有 9 类，通过查询说明文件，也将其设置标签进行判断；游戏类型共 3 类，通过查询说明文件，将其设置标签进行判断；至于 113 个英雄选择情况，变量取值为 1 反映某个英雄在一场比赛中是自己队伍使用，取值为 0 表示双方均不使用，取值为-1 反映某个英雄在比赛中由对手使用。最后对因变量也因子化，满足逻辑回归的格式要求。

3 描述性分析

基于 R 软件分析，我们做出以下描述性分析：

3.1 总体的比赛输赢情况：

表 1 比赛总体结果表

Result	-1	1
Frequency	43868	48782

我们在直觉上觉得两种结果的差异并不大。

3.2 不同国家地区的比赛输赢频数：

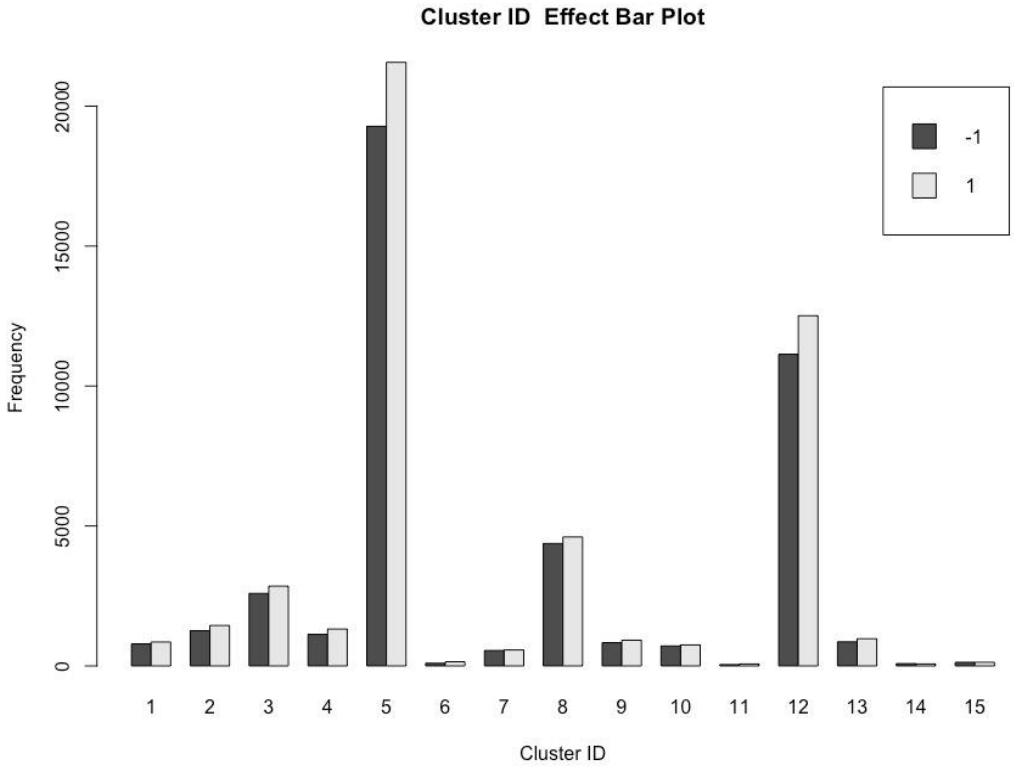

图 1 游戏分区频数图

在不同地区或国家的比赛结果较为均衡，但在大多数地区，比赛队伍获得胜利的概率稍稍高于失败的概率。

3.3 不同游戏模式下的比赛输赢频数：

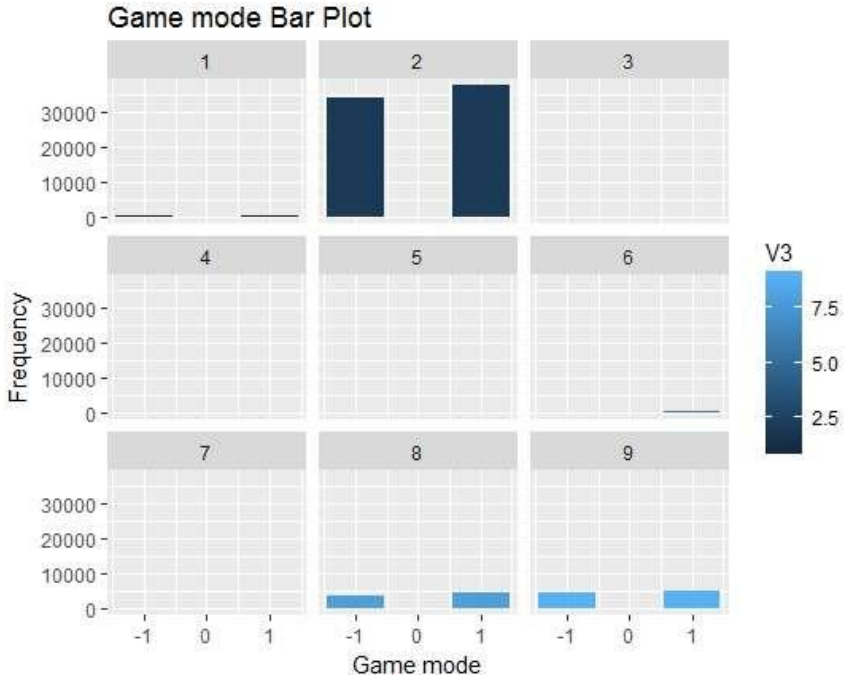

图 2 游戏模式频数图

在图中我们看出游戏玩家尤其偏爱第 2 种游戏模式，第 8、9 两种游戏模式也会有少量玩家会选择，但是选择其他几种游戏模式的玩家屈指可数。而且每一种游戏模式下的输赢状况差异不明显，胜负概率均在 50%左右，比较均衡。

3.4 不同游戏类型下的比赛输赢频数：

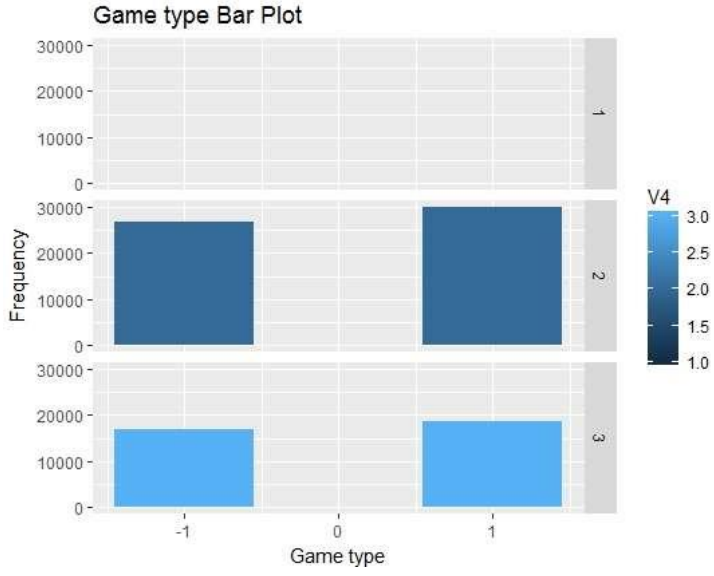

图 3 游戏类型频数图

在游戏类型的选择中，游戏玩家喜爱第 2、3 两种游戏类型，而第一中游戏类型的存在更类似于摆设。类似于游戏模式的结论，每一种游戏类型下的输赢状况差异也不明显，但是每种游戏类型下队伍取得胜利的概率稍稍高于失败的概率。

3.5 是否选取第一个英雄时的比赛输赢频率：

表 2 是否选取第一个英雄时的比赛频率表

Result/Pick	-1	0	1
-1	0.4440624	0.4728942	0.5096478
1	0.5559376	0.5271058	0.4903522

我们以第一个英雄为例，分析了是否选取该英雄与比赛结果的概率。由表中可以看出该英雄是否被选择，输赢概率差异均没有特别明显。

3.6 不同比赛结果的情况下双方使用最频繁的英雄频数图：

表 3 不同比赛结果的情况下双方使用最频繁的英雄频数表

Result/Team	Us	Enemy
1	Hero9(9529)	Hero9(7547)
-1	Hero44(7007)	Hero9(8200)

在队伍比赛胜利的情况下，我方队伍使用的最频繁的英雄是 mirana（9 号英雄），而对方队伍使用该英雄的频率也比较高，可见这个英雄比较受到游戏玩家的喜爱。而在队伍比赛失败的情况下，我方队伍使用的最频繁的英雄是 phantom_assassin（44 号英雄），那么这是否提醒了我们应该尽量避免使用该英雄呢？对方队伍使用最频繁的仍然是 9 号英雄，那么它是否与比赛输赢有着关联，我们还要进行更深一步的探讨。

3.7 每位英雄的胜率和使用频率的关系（胜率最高的前 10 位英雄）

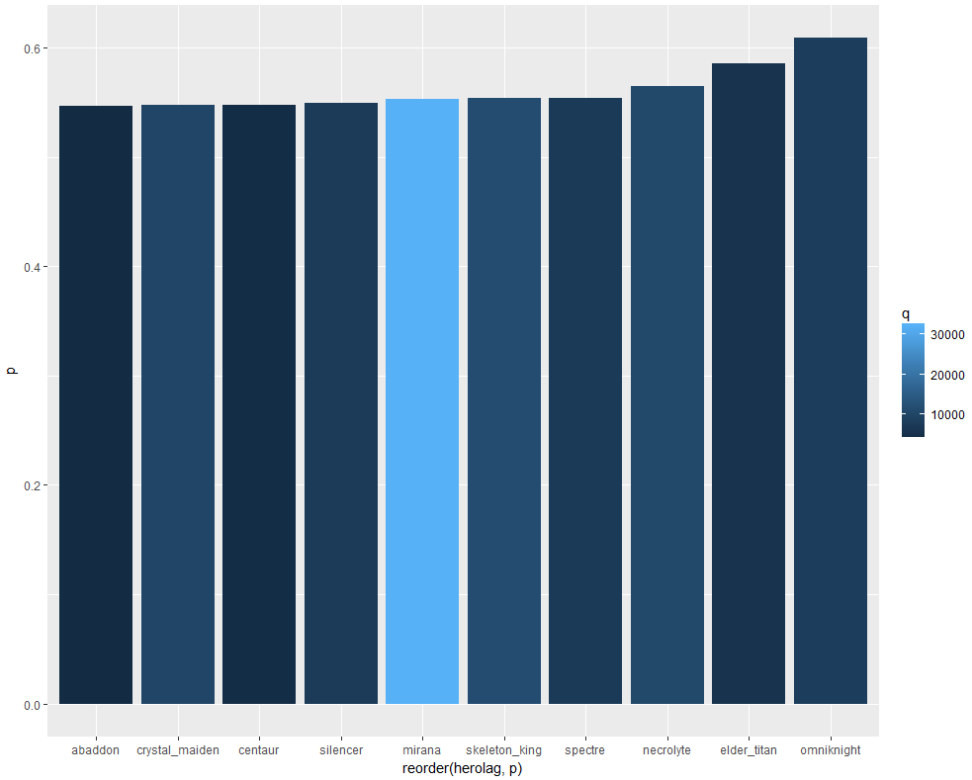

图 4 英雄的胜率和使用频率的关系图

如图，横坐标表示英雄的名字，纵坐标是英雄的胜率，条形颜色的深浅表示英雄的使用频率（颜色越浅表示使用的越频繁），在图中，我们可以看出英雄的胜率和使用频率并不存在明显的正相关或者负相关的关系（由于条形图没有按颜色深浅的渐变排列），我们还可以看出 omniknight（57 号英雄）是总体获胜率（包括我方和对方）最高的英雄。

3.8 游戏类型 2 与游戏类型 3 的英雄胜率与使用频率比较：

游戏类型 2 下的英雄胜率（只取胜率最高的前 10 位英雄）

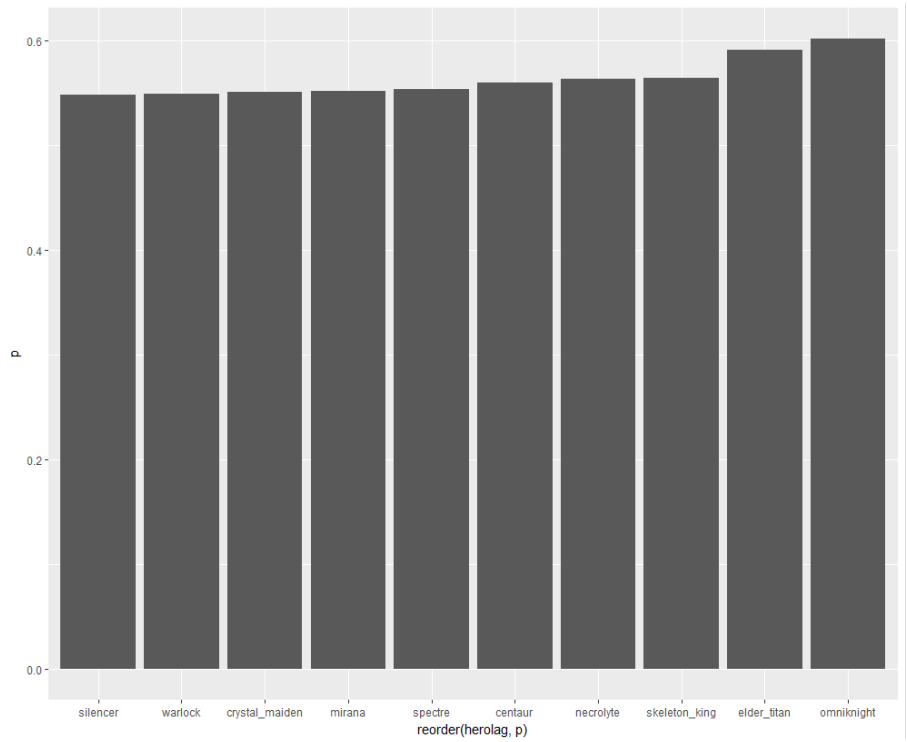

图 5 游戏类型 2 下的英雄胜率图

游戏类型 3 下的英雄胜率（只取胜率最高的前 10 位英雄）

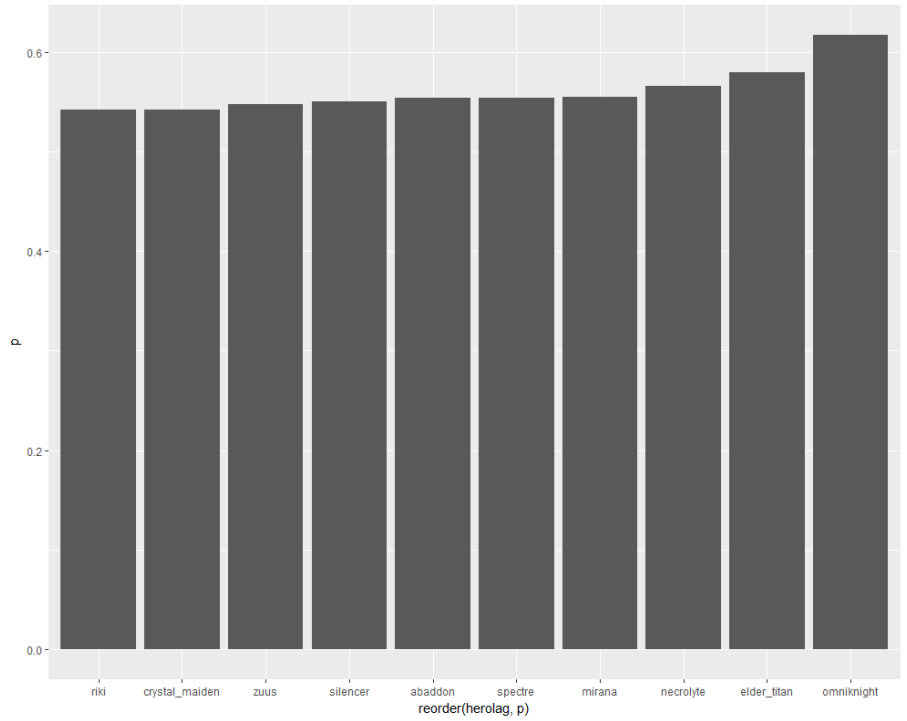

图 6 游戏类型 3 下的英雄胜率图

由这两幅英雄胜率图可以看出在两种游戏类型下面，omniknight（57 号英雄）和 elder_titan（103 号英雄）均是获胜率最高的英雄，而其他英雄的获胜率排序状况则不太相同。

游戏类型 2 下的使用频数（只取使用率最高的前 10 位英雄）

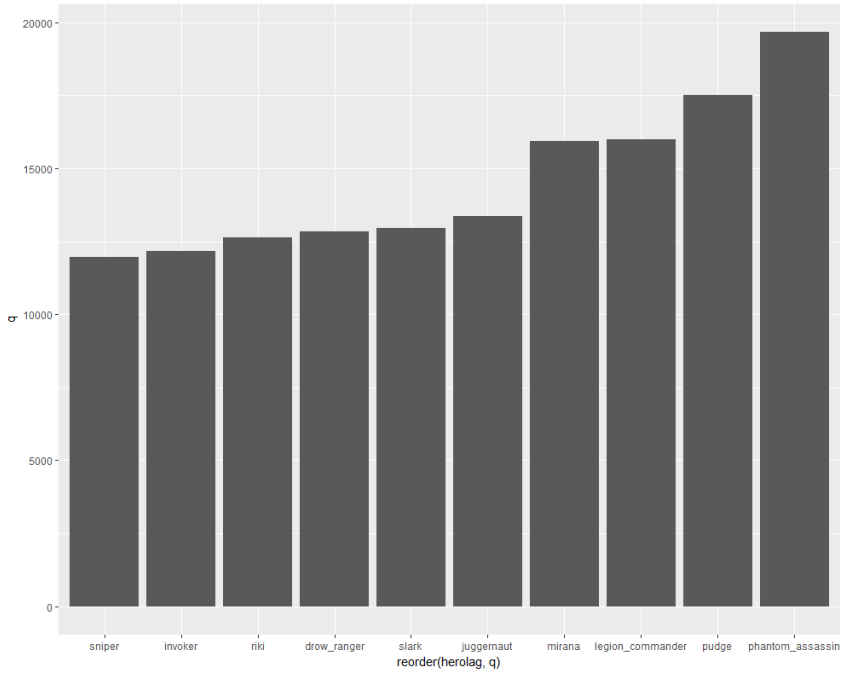

图 7 游戏类型 2 下的英雄使用频数图

游戏类型 3 下的使用频数（只取使用率最高的前 10 位英雄）

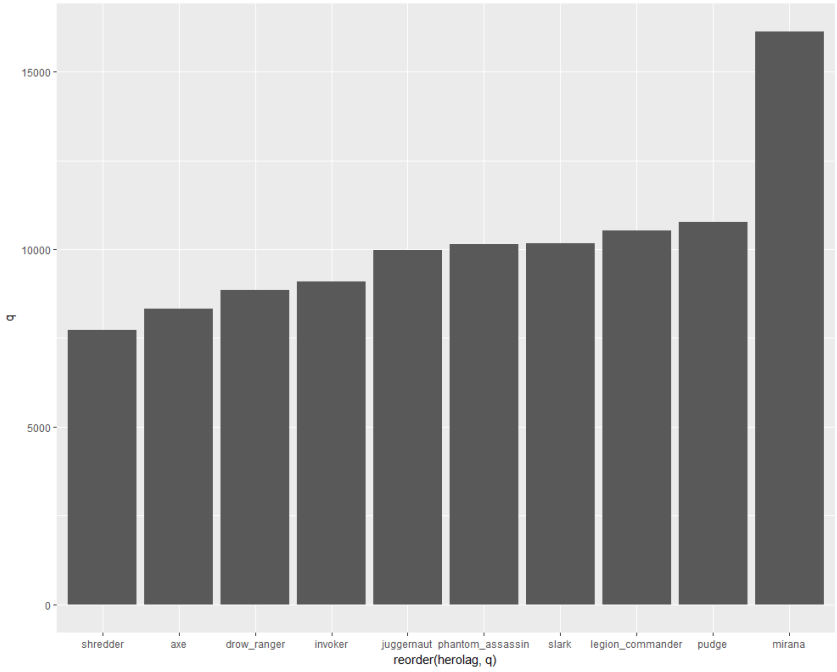

图 8 游戏类型 3 下的英雄使用频数图

由这两幅英雄使用频数图可以看出在两种游戏类型下面，phantom_assassin（44 号英雄）是游戏类型 2 最频繁使用的英雄，mirana（9 号英雄）是游戏类型 3 最频繁使用的英雄，并且使用频率明显高于第二名英雄 pudge（14 号英雄），pudge 在两种游戏类型下使用频率都位居第二。其他英雄使用频率情况在这两种游戏类型下也不太相同。

4 构建逻辑回归

在这一部分，我们将训练集随机拆分为 2:1 的建模训练集和建模测试集，通过对建模训练集构建模型，然后利用建模测试集进行回测，探究模型的准确率在不同惩罚项下的变化，并且将它们和总体模型（无惩罚项）一起对比，选择最优模型。

4.1 基于所有变量构建逻辑回归模型

根据前面对变量的数据处理，导入模型，构建逻辑回归，由于虚拟变量太多，近 300 个，因此在这里不把模型的系数以及 p 值列出，模型通过测试集进行验证，得到一组概率，然后分割点的选择是使得模型对训练集的预测结果中胜利的比例与真实训练集的胜利比例一致，这里计算结果为 0.516。我们通过召回率（recall）、精度（precision）、两者的调和平均的倒数（f1）、总体准确率（accuracy）和 AUC 面积来全方位衡量模型的有效性。

结果如下：

表 4 基于所有变量的逻辑回归模型预测结果表

predict/ture	failure	victory
failure	8320	6059
victory	6335	10169

此时模型测试集的召回率为 62.66%，精度为 61.99%，f1 为 62.32%，准确率为 59.87%。因此模型能够识别真正的胜利者的比例，以及模型判定的胜利者最终真正胜利的比例都较高，然而模型总体的准确率太低，不到 60%。由于这些比例受分割点的影响较大，因此这里结合 ROC 曲线进一步考虑整体效果。

如图所示，AUC 为 63.51%，所以模型总体效果一般。

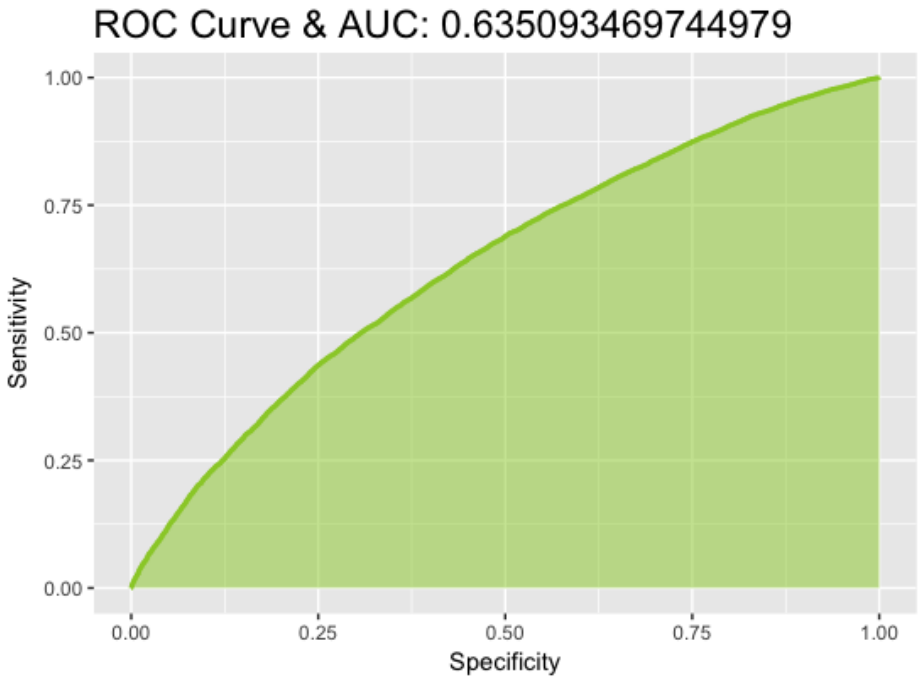

图 9 基于所有变量的逻辑回归模型 AUC 图

4.2 基于 LASSO 惩罚项的思想构建逻辑回归

鉴于 LASSO 的变量选择效果，因此在原来的逻辑回归里加上该惩罚用来进行变量选择，惩罚项存在一个调节参数，这里我们通过交叉验证来确定，结果调节参数设定为 0.000297，通过该参数仅对变量缩减了几个，因此虽然效果最好，但是考虑到模型的可解释性，我们调高调节参数达到 0.009，从而对模型的变量进行大幅度缩减，最终剩下的虚拟变量只是英雄选择与否的类型，国家地区、游戏模式和游戏类型并没有显著效应。

变量的系数表如下所示：

表 5 基于 LASSO 惩罚项的逻辑回归模型系数表

hero1usebyus	hero2usebyus	hero5usebyus	hero6usebyus	hero9usebyus
-0.0105	0.0027	0.0639	0.0690	0.1797
hero12usebyus	hero14usebyus	hero18usebyus	hero19usebyus	hero21usebyus
-0.0096	0.0424	0.0702	-0.0524	-0.1198
hero22usebyus	hero25usebyus	hero32usebyus	hero36usebyus	hero39usebyus
0.0463	-0.0769	0.0912	0.0880	-0.1425
hero42usebyus	hero46usebyus	hero48useby	hero53usebyus	hero57none

		us		
0.0652	-0.0057	0.0149	-0.1327	0.0221
hero57usebyus	hero58usebyus	hero59usebyus	hero61usebyus	hero67usebyus
0.3411	-0.0895	-0.0149	-0.1214	0.0007
hero74usebyus	hero75usebyus	hero80usebyus	hero82usebyus	hero84usebyus
-0.0380	0.0482	-0.1324	-0.0023	0.0139
hero91usebyus	hero100usebyus	hero101usebyus	hero103usebyus	hero106usebyus
-0.1271	-0.0552	-0.0050	0.1291	-0.1790

该系数表反映了选择不同的英雄对己方胜率的正负影响，系数为正，表示选择该英雄可以提高胜率，系数为负则相反。以第一个系数为例，表示使用第一个英雄，可以降低对数发生比（log odds）0.0105，即表示胜率下降，具体下降幅度取决于原来的胜率。因此这个表格告诉我们为了提高胜率应该选择第 2、5、6、9、14、18、22、32、36、42、48、57、67、75、84、103、106 的英雄。

表 6 基于 LASSO 惩罚项的逻辑回归模型预测结果表

predict/ture	failure	victory
failure	7840	6819
victory	6841	9383

此时模型测试集的召回率为 57.91%，精度为 57.83%，f1 为 57.83%，准确率为 55.77%。因此模型能够识别真正的胜利者的比例，以及模型判定的胜利者最终真正胜利的比例都较高，然而模型总体的准确率太低，不到 60%。而且总体来看准确率比原来所有变量都有的模型差，由于这些比例受分割点的影响较大，因此这里结合 ROC 曲线进一步考虑整体效果。

如图所示，AUC 为 58.15%，所以模型总体效果一般。

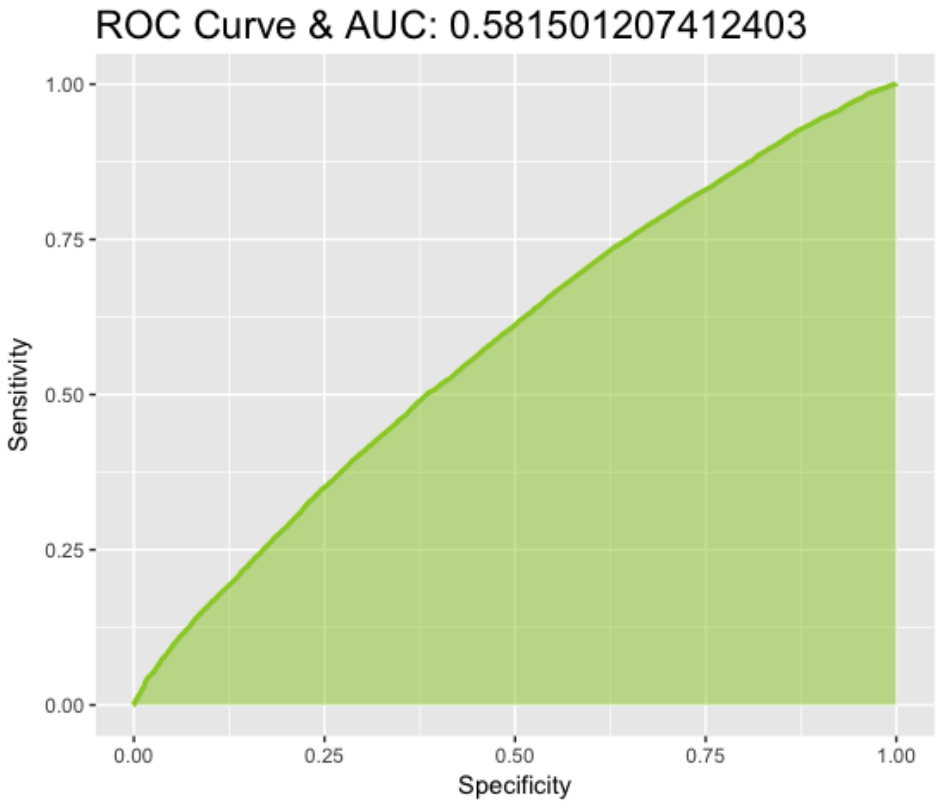

图 10 基于 LASSO 惩罚项的逻辑回归模型 AUC 图

4.3 基于 SCAD 惩罚项的思想构建逻辑回归

由于 LASSO 的有偏性，因此虽然可以进行变量选择，但是模型的系数有偏，因此学者之后提出了 SCAD 惩罚，该惩罚项既可以进行变量选择，也可以保持无偏，而且系数稀疏的过程可以连续，所以选择该惩罚项。在使用的过程也通过交叉验证选择其调节参数 0.0001，同样这样非零参数达到 238 个，这样的模型解释性很弱，因此选择调节参数为 0.0105，结果得到 28 个变量系数非零。

通过系数表可以发现，如今的系数个数比用 LASSO 少了 7 个，原因在于调节参数的选择，然而有趣的是 SCAD 方法选择的变量均被 LASSO 选择的变量中包含，说明这些变量确实是极大地解释了队伍的胜率。

变量的系数表如下所示：

表 7 基于 SCAD 惩罚项的逻辑回归模型系数表

hero1usebyus	hero2usebyus	hero5usebyus	hero6usebyus	Hero9usebyus
-0.0055	0.0141	0.0218	0.0371	0.2829
hero14usebyus	hero18usebyus	Hero21usebyus	Hero22usebyus	hero25usebyus

0.0283	0.0847	-0.0871	0.0450	-0.0937
Hero32usebyus	Hero36usebyus	hero39usebyus	Hero42usebyus	Hero46usebyus
0.1057	- 0.0438	-0.0489	0.0477	-0.0101
hero48usebyus	hero91usebyus	hero100usebyus	hero53usebyus	hero57none
0.0211	-0.1198	-0.0287	-0.1410	0.02812
hero57usebyus	hero58usebyus	hero103usebyus	hero61usebyus	hero106usebyus
0.4391	-0.0324	0.1595	-0.1992	-0.1148
hero74usebyus	hero75usebyus	hero80usebyus		
0.0277	0.0742	-0.0989		

表 8 基于 SCAD 惩罚项的逻辑回归模型预测结果表

predict/ture	failure	victory
failure	7343	6435
victory	7387	9718

此时模型测试集的召回率为 60.12%，精度为 56.81%，f1 为 58.42%，准确率为 55.24%。因此模型能够识别真正的胜利者的比例，以及模型判定的胜利者最终真正胜利的比例都较高，然而模型总体的准确率太低，不到 60%。而且总体来看准确率比原来所有变量都有的模型差，但精度和召回率高于 LASSO，总体准确率略低于 LASSO，由于这些比例受分割点的影响较大，因此这里结合 ROC 曲线进一步考虑整体效果。

如图所示，AUC 为 56.63%，比 LASSO 低，所以模型总体效果相对更差。

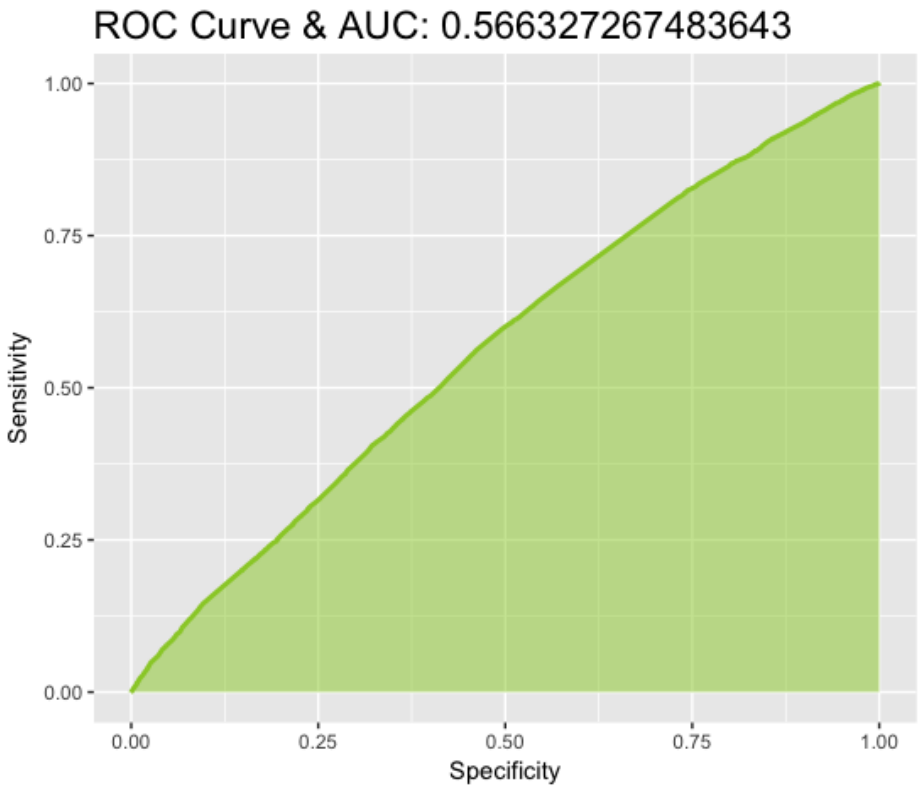

图 11 基于 SCAD 惩罚项的逻辑回归模型 AUC 图

4.4 基于 MCP 惩罚项的思想构建逻辑回归

之前提到 LASSO 的有偏性，因此虽然可以进行变量选择，但是模型的系数有偏，因此学者之后提出了 SCAD 惩罚，之后另外还有学者提出了 MCP 惩罚，该惩罚项和 SCAD 具有类似的效果，既可以进行变量选择，也可以保持无偏，而且系数稀疏的过程可以连续，所以选择该惩罚项。在使用的过程也通过交叉验证选择其调节参数 0.001，同样这样非零参数达到 198 个，这样的模型解释性很弱，因此选择调节参数为 0.0098，结果得到 31 个变量系数非零。

通过系数表可以发现，如今的系数个数比用 LASSO 少了 4 个，原因在于调节参数的选择，然而和之前两个惩罚不一样的是，尽管大多数选择的变量相同，但是存在一些变量是之前两个惩罚没有选择的。因此 MCP 惩罚在实际应用中与前两个差异较大。

表 9 基于 MCP 惩罚项的逻辑回归模型系数表

hero1usebyus	hero100usebyus	hero5usebyus	hero6usebyus	hero9usebyus
-0.0452	-0.1467	0.1186	0.1114	0.2437

Hero21usebyus	hero14usebyus	hero18usebyus	hero19usebyus	hero21none
-0.4854	0.02814	0.0666	-0.0524	-0.2276
Hero34usebyus	hero25usebyus	hero32usebyus	hero36usebyus	hero39usebyus
-0.0381	-0.1503	0.1406	0.1045	-0.1555
hero42usebyus	hero102usebyus	hero48usebyus	hero53usebyus	hero57none
0.1167	-0.1467	0.0318	-0.2107	0.3205
hero57usebyus	hero58usebyus	hero59usebyus	hero61usebyus	hero67usebyus
0.8277	-0.0456	-0.0226	-0.1491	0.0053
hero74usebyus	hero75usebyus	hero80usebyus	hero103usebyus	hero106usebyus
-0.0828	0.0565	-0.0589	0.2305	-0.18963
hero91usebyus				
-0.1170				

此时模型测试集的召回率为 59.87%，精度为 57.75%，f1 为 58.79%，准确率为 55.94%。因此模型能够识别真正的胜利者的比例，以及模型判定的胜利者最终真正胜利的比例都较高，然而模型总体的准确率太低，不到 60%。而且总体来看准确率比原来所有变量都有的模型差，但精度和召回率高于 LASSO，总体准确率略低于 LASSO，略高于 SCAD，由于这些比例受分割点的影响较大，因此这里结合 ROC 曲线进一步考虑整体效果。

表 10 基于 MCP 惩罚项的逻辑回归模型预测结果表

predict/ture	failure	victory
failure	7572	6506
victory	7100	9705

如图所示，AUC 为 58.07%，比 LASSO 低，比 SCAD 高，所以模型总体效果相对 SCAD 好一些。

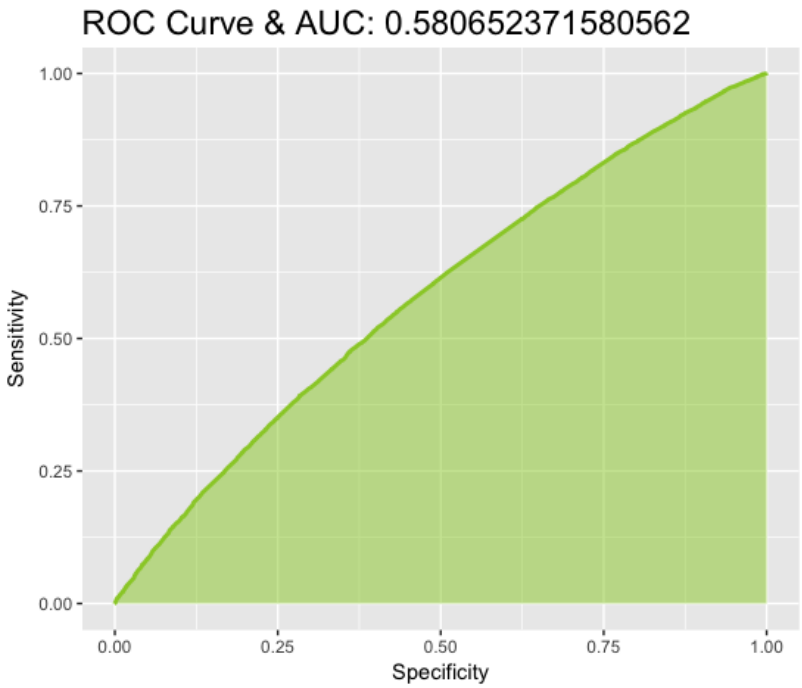

图 12 基于 MCP 惩罚项的逻辑回归模型 AUC 图

4.5 基于三类惩罚项的逻辑回归对比分析

尽管利用所有变量建立逻辑回归的效果最好，然而各项指标也只比有惩罚项的回归高 5%以内，这种差距并不明显，但变量却多了近 100 个，因此我们有理由相信更多的变量虽然提供了更多信息，但主要是噪音，否则也不会提升较少，因此我们将三类惩罚回归进行对比，尽管差异不大，但仍然可以筛选出相对较好的模型，然后基于该模型用测试集测试！

表 11 对比分析

model	recall	precision	f1	accuracy	AUC
LASSO	57.91%	57.83%	57.83%	55.77%	58.15%
SCAD	60.12%	56.81%	58.42%	55.24%	56.63%
MCP	59.87%	57.75%	58.79%	55.94%	58.07%

通过该表可以看出，由于 f1 是对召回率和精度的综合指标，因此 MCP 的效果最好，准确率也是 MCP 最高，因此基于该模型对测试集进行预测。

4.6 对开始的测试集进行模型预测

测试集一共有 10293 条数据，利用 MCP 模型对其进行预测，得到混淆矩阵：

表 12 基于 MCP 惩罚项的逻辑回归模型对测试集的预测

predict/ture	failure	victory
failure	2715	2548
victory	2076	2954

此时测试集的召回率为 53.69%，精度为 58.73%，f1 为 56.09%，准确率为 55.08%。AUC 面积为 57.30%。

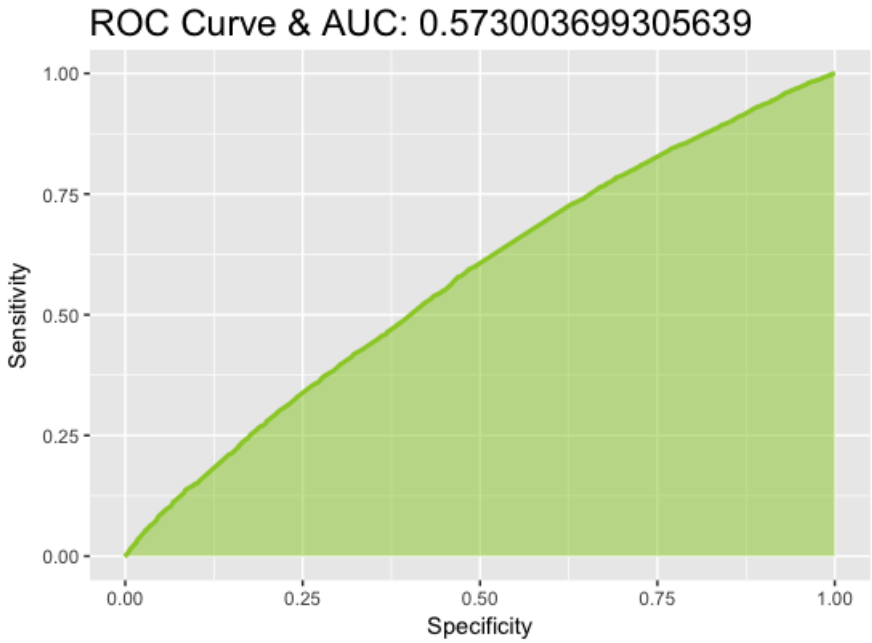

图 13 基于 MCP 惩罚项的逻辑回归预测 AUC 图

5 构建 SVM 模型

支持向量机（supportvectormachine，SVM）最初提出是为了解决二类分类问题，现在被广泛用于解决多类非线性分类问题和回归问题，是支持向量分类器的一个扩展，扩展的结果是支持向量机使用了一种特殊的方式，即核函数（kernel）来扩大特征空间。一般来讲，核函数越复杂，模型越偏向于拟合过度。在软边缘参数 C 方面，它可以看作是 LASSO 算法中的 lambda 的倒数，C 越大模型越偏向于拟合过度，反之则拟合不足。常用的核函数有如下种类：Linear，polynomial，Radial basis 等。这里本模型使用高斯核函数（Radial basis），它是当下比较流行且易于选择。SVM-type 选用 C-classification。Cost 参数用来设置观测穿过间隔的

成本，如果 cost 参数设置较小，那么间隔就会很宽，许多支持向量会落在间隔上或者穿过间隔，如果 cost 参数设置较大，那么间隔就会很窄，更少的支持向量会落在间隔上或者穿过间隔。这里本模型取 cost 为 1。Gamma 参数取值为 0.004。通过建立 SVM 模型，可得到拟合结果。共有 83263 个支持向量，其中 41591 个是一类，另外 41672 个是一类。

通过拟合模型对测试集数据进行预测，得到如下结果：

表 13　SVM 模型预测结果表

predict/ture	failure	victory
failure	2518	1863
victory	2274	3639

由结果可知预测准确率达到了 59.81%，相较于 lasso 模型准确度基本一致，仅低于基于所有变量构建的逻辑回归模型，但高于其他加入惩罚项的模型。同时得出 AUC 曲线图，AUC 为 63.1%，也仅低于基于所有变量构建的逻辑回归模型，模型总体效果一般。

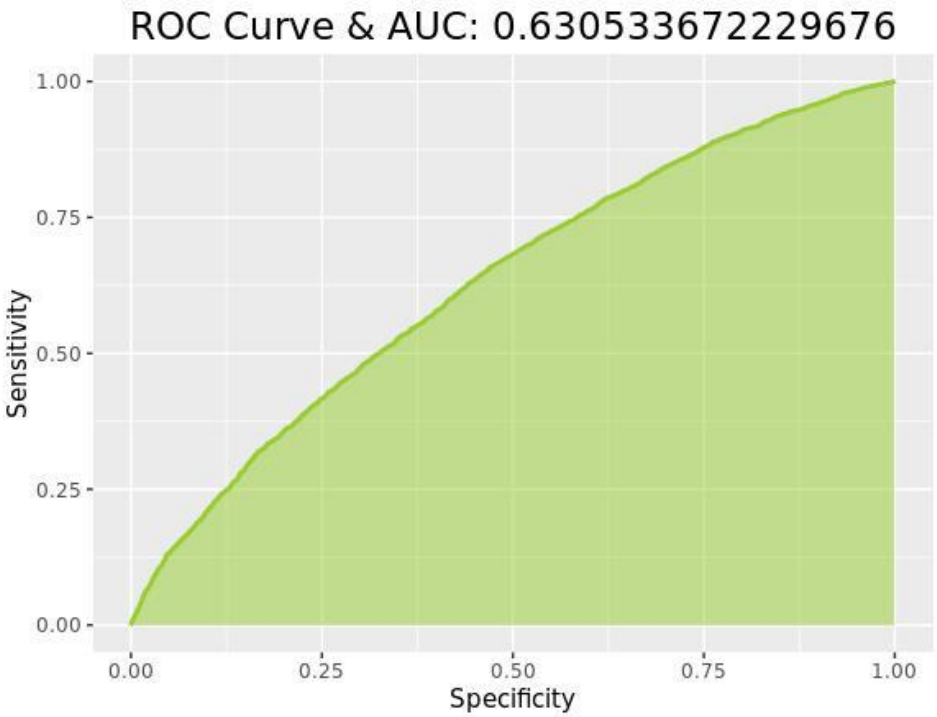

图 14 SVM 模型 AUC 图

6 构建 KNN 模型

KNN 算法有许多优点，如简单有效，对数据的分布没有要求，训练阶段很快。使得拟合过程快速高效。KNN 算法的思路是：如果一个样本在特征空间中的 k 个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别，则该样本也属于这个类别。简言之，就是将未标记的案例归类为与它们最近相似的、带有标记的案例所在的类。KNN 算法中，所选择的邻居都是已经正确分类的对象。该方法在定类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。本模型选取了 k 为 1 和 2 以及有权重的 KNN 算法（kknn）。分别得到他们的预测结果。

表 14 KNN 模型（K=1）预测结果表

predict/ture	failure	victory
failure	2355	2647
victory	2437	3035

表 15 KNN 模型（K=2）预测结果表

predict/ture	failure	victory
failure	2255	2391
victory	2537	3111

表 16 KKNN 模型（K=1）预测结果表

predict/ture	failure	victory
failure	2358	2434
victory	2543	2959

这 3 种情况差别均不大，预测准确率分别为 52.36%，52.12%，以及 51.65%。其中当 K 取值为 1 时准确率最高，有权重的 knn 算法最低。相比于以上逻辑回归预测以及 SVM 预测算法，准确度略显不足，但也基本维持在一个水准之下。

7 结果解读

在本次研究中，我们通过描述性统计进行探索性数据分析，发现无论是国家地区、游戏模式还是游戏类型，均对比赛的胜负没有显著影响，这也可以从后面惩罚项逻辑回归中，这些变量均被压缩为 0 进一步佐证；另外，使用 9 号英雄的队伍胜率最高，使用 44 号英雄的队伍胜率最低，这可以从后面逻辑回归中 9 号英雄的正系数较大来进一步说明，而 44 号英雄并没有出现在逻辑回归筛选变量之后，因此该英雄的边际效应可能是其他因素导致的。

我们还通过传统的统计模型和机器学习的方法来探究一个合理预测模型，结果表示，无论是传统的统计模型，还是机器学习的方法，总体效果均不好；然而两者相比还是机器学习方法的效果相对好一些；除此之外，逻辑回归的变量越多效果越好，然而好的程度很有限，因此采用三类惩罚项进行变量筛选，发现 MCP 在这三类模型中的效果最好。

表 17 MCP 模型变量选取

英雄 1	英雄 2	英雄 5	英雄 6	英雄 9
antimage	axe	crystal_maiden	drow_ranger	mirana
英雄 12	英雄 14	英雄 18	英雄 19	英雄 21
phantom_lancor	pudge	sven	tiny	windrunner
英雄 22	英雄 25	英雄 32	英雄 36	英雄 39
zuus	Lina	riki	necrolyte	queenofpain
英雄 42	英雄 44	英雄 46	英雄 48	英雄 53
skeleton_king	phantom_ assassin	templar_ assassin	luna	furion
英雄 57	英雄 58	英雄 59	英雄 61	英雄 67
omniknight	enchantress	huskar	broodmother	spectre
英雄 74	英雄 75	英雄 80	英雄 82	英雄 84
invoker	silencer	lone_druid	meepo	ogre_magi
英雄 91	英雄 100	英雄 101	英雄 103	英雄 106
wisp	tusk	skywrath_mage	elder_titan	ember_spirit