# 问题定义

绝地求生是一款火遍国内外的大逃杀类写实射击游戏，简称“PUBG”。在每局比赛完结后，玩家能基于他们在比赛中的生存时间及伤害和杀死敌对玩家的数目，获得相应的游戏货币和分数以及第一名的荣誉。每一局游戏最多允许有100名玩家参与，且争取存活至最后，可以单排、双排、四排的游戏模式。而无论在哪种情况下，生存至最后的一人或团队都赢得比赛。在每局比赛开始时，玩家所控制的角色都会在没有携带任何东西的情况下跳伞至战场上。着陆后，玩家便能搜索建筑物或其他场所，从而获取随机分布在整个地图上的武器、车辆及其他装备。作为这款游戏的忠实爱好者，我决定建立一个线性回归模型，根据他们的最终统计数据预测玩家的比赛排名。

# 数据集

数据集的来源以下有两种，这两种来源的数据格式都是相同的。

1.PUBG的官网给出了大量匿名的PUBG游戏统计数据，数据格式为每行包含一个玩家的赛后统计数据。数据来自所有类型的比赛：单排，双排，四排和自定义（不保证每场比赛有100名玩家，每个小组最多4名玩家）。

2.Kaggle的官网上也有一个“绝地求生排名预测”竞赛。链接地址：

https://www.kaggle.com/c/pubg-finish-placement-prediction

# 数据分析

首先我们需要查看数据集的基本信息，从图3.1的输出结果可以得知数据集共有29列。各列的含义如下：

**DBNOs**：被击中的敌人玩家数量

**Assists：**被队友击杀的敌人数量

**Boosts：**使用过多少个提升性的物品

**damageDealt：**造成的总伤害-自己所受的伤害

**headshotKills**：击杀头部死亡的敌人数量

**heals：**医疗用品使用数量

**Id**：玩家ID

**killPlace**：杀死敌人数量的排名

**killPoints**- 基于杀戮的玩家外部排名。将其视为Elo排名，只有杀死才有意义。如果 rankPoints 中的值不是-1，那么 killPoints 中的任何0都应被视为“None”

**killStreaks**：短时间内杀死敌人的最大数量

**kills**：杀死敌人的数量

**longestKill**：玩家和死亡时被杀的玩家之间的最长距离。这可能具有误导性，因为击倒一个玩家并开车离开可能会导致一个较大的最长杀伤值

**matchDuration：**匹配用的秒数

**matchId**：比赛场次ID（每一局一个ID）

**matchType**：单排/双排/四排；标准模式是 “solo”，“duo”，“squad”，“solo-fpp”，“duo-fpp”和“squad-fpp”;其他模式来自事件或自定义匹配

**rankPoints**：类似 Elo 的玩家排名。这个排名是不一致的，在API的下一个版本中已被弃用，所以请谨慎使用。值“-1”将取代“None”

**Revives**：玩家救援队友次数

**rideDistance：**玩家使用交通工具行驶了多少米

**roadKills**：使用交通工具击杀人数

**swimDistance**：游泳距离

**teamKills**：该玩家击杀队友次数

**vehicleDestroys**：损坏交通工具次数

**walkDistance**：步行距离

**weaponsAcquired**：捡到的武器数量

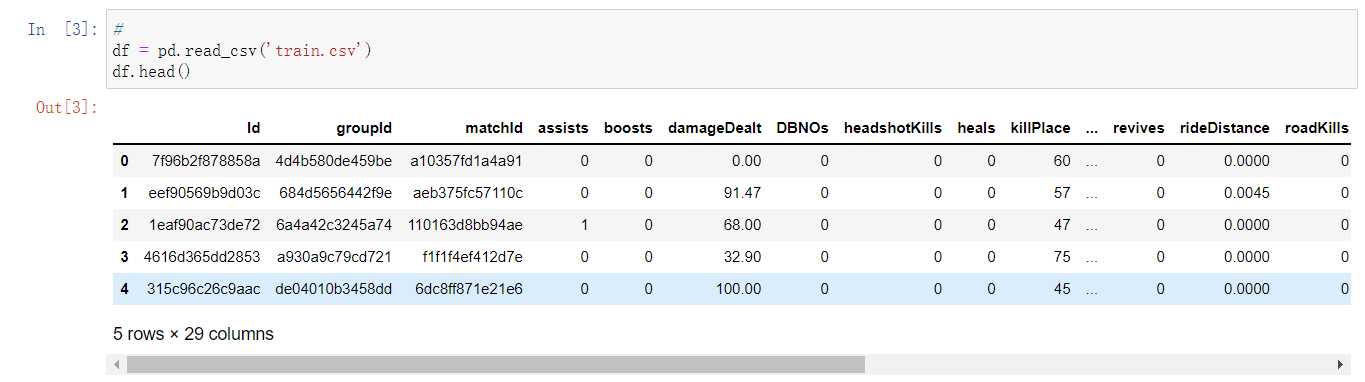
**winPoints：**基于胜利的玩家外部排名。(你可以把这想象成Elo排名中只有获胜才重要)如果rankPoints中的值不是-1，那么winPoints中的任何“0”都应该被视为“None”

**groupIdID**：队伍的ID。如果同一组玩家在不同的比赛中比赛，他们每次都会有不同的GroupId

**numGroups**：在该局比赛中有玩家数据的队伍数量

**maxPlace**：在该局中已有数据的最差的队伍排名（可能与该局队伍数不匹配，因为数据收集有跳跃）

**winPlacePerc：**预测目标，是以百分数计算的，介于 0-1 之间，1 对应第一名，0 对应最后一名。 它是根据maxPlace计算的，而不是 numGroups ，因此匹配中可能缺少某些队伍



**图3.1数据信息预览**

我们所要预测的列为 winPlacePerc，它是综合排名的指数，是归一化的结果。它的范围是0-1之间。

# 具体过程

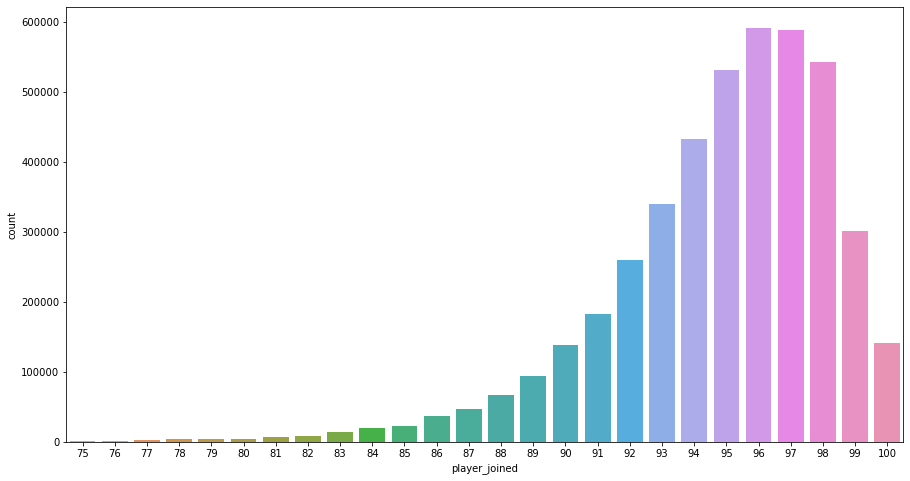
## 1.数据预处理

（1）处理缺失数据

我们首先需要将训练集中winPlacePerc缺失的数据删除掉

（2）处理参与比赛人数数据

参与玩家人数较少的比赛的数据没有训练的意义，因此我们需要统计出玩家参与人数大于75人的比赛场次。参赛人数对于击杀量、伤害量、存活时间等具有一定的影响，因此我们需要将数据调整到相当于参赛人数为100人的标准下，使得模型更加公平。



**图4.1 玩家超过75人的比赛场次**

（3）处理开挂的数据

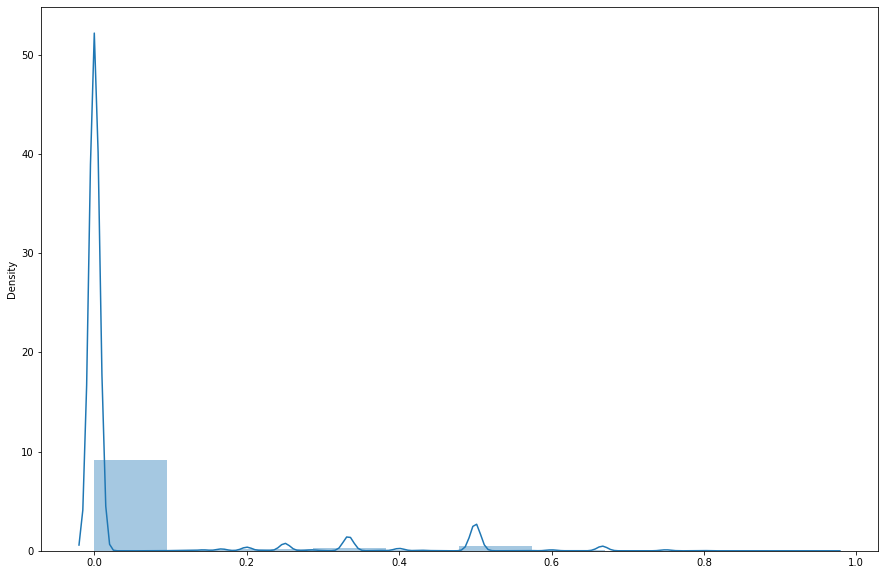
由于数据量非常庞大而且含有一些异常数据，因此我们需要进行数据清洗，剔除开挂的异常数据，在这里我们选定了一些剔除条件

1. 没有移动距离就有击杀人数的数据
2. 开车杀人大于10的数据
3. 击杀量大于30的数据



**图4.2 击杀数据图**

1. 爆头率为1的数据



**图4.3爆头率数据图**

1. 远击杀距离大于1000米的数据



**图4.4 远距离击杀数据图**

1. 步行移动距离大于10000米的数据



**图4.5步行移动距离数据**

1. 骑行移动距离大于15000米的数据



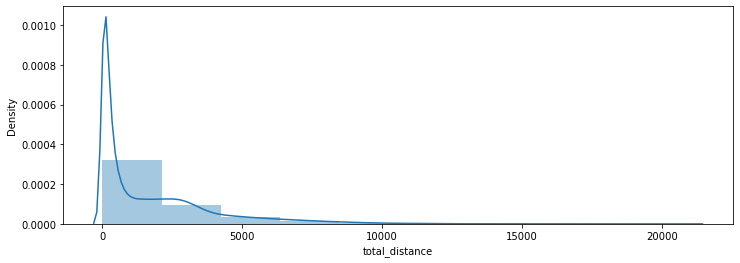
**图4.6骑行移动距离数据**

1. 游泳移动距离大于1000米的数据



**图4.7游泳移动距离数据**

1. 总移动距离大于15000米的数据



**图4.8总移动距离数据**

1. 武器获得50把以上的数据



**图4.9获得武器数据**

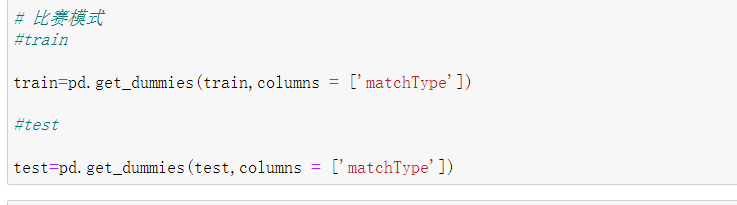
1. 治疗量超过40的数据



**图4.10玩家治疗量数据**

（4）类别变量处理

因为比赛模式有很多种，我们需要处理好类别变量。



**图4.11类别变量处理**

## 2.建立线性回归模型

（1）特征选择

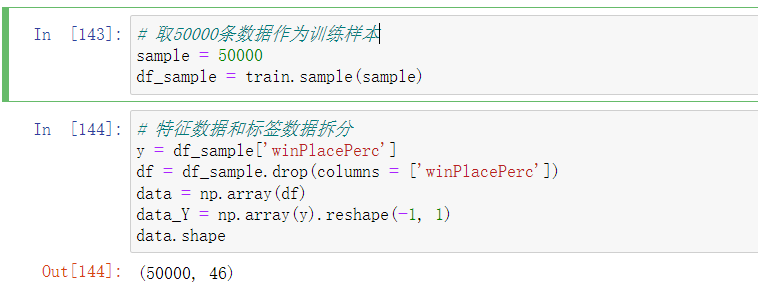
将没有用的数据信息以及标签数据删除得到特征表，并将最终排名作为标签



**图4.12删除无用列**

（2）数据集划分

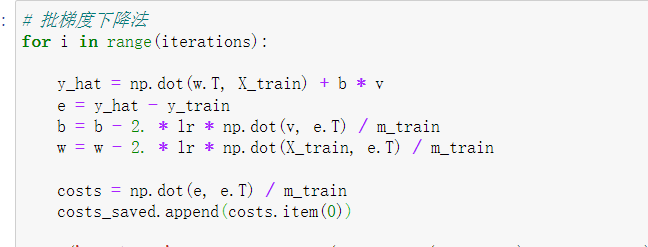
由于该数据集的数据量非常庞大，所以我们随机选取50000条数据作为训练样本，按照4:1的比例划分数据集。



**图4.13数据集划分**

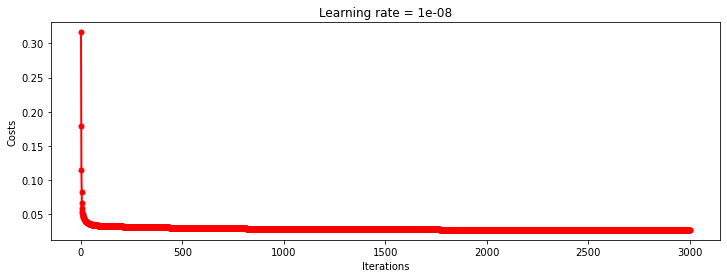
（3）模型训练

我们采用批梯度下降法训练线性回归模型，迭代次数选取3000次，学习率设为0. 00000001，epochs为1.



**图4.14 批梯度下降法训练模型**

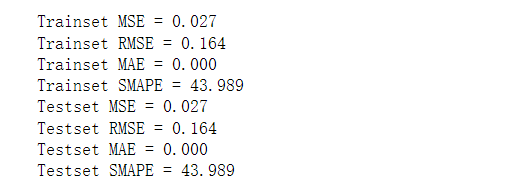
下图为均方误差代价函数的值对迭代次数变化的曲线。



**图4.15损失函数曲线图**

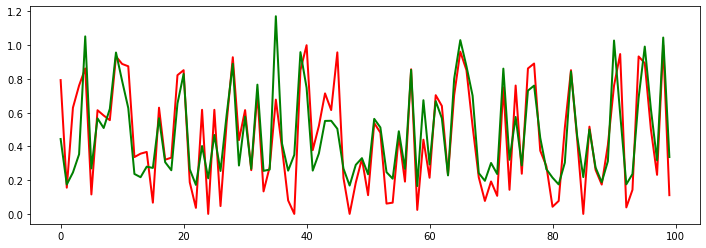
## 3. 模型结果分析及可视化

首先选取了4个评价指标MSE，RMSE，MAE以及SMPAE来对模型进行评价，评价指标如下所示。



**图4.16模型评价指标**

从中可以看到，线性回归模型存在着一定误差，我们选取前100个数据将真实值和预测值绘制成折线图，观察真实值与预测值之间的差距，如下图所示



**图4.16线性回归预测值与真实值的对比**

红色折线代表真实值，绿色折线代表预测值，从中可以发现，当真实值的胜率偏大时，预测值低于真实值，然而当真实值的胜率偏低时，预测值高于真实值，这说明当利用线性回归模型进行预测时，预测的数字偏于稳定。

# 改进

从模型评估来看，我们发现线性回归模型的预测值并没有很好的拟合真实值，甚至在预测时出现了预测值大于1以及小于0的情况。

我在网上看到一个博主利用sklearn工具包，采用MLP模型、随机森林模型和线性模型进行PUBG排名预测，三者相比较之下，发现随机森林模型的各个指标相比较来说都是最优的，但其实MLP模型和随机森林模型的评价指标相差并不大。在极端情况下，随机森林的效果要更好。虽然随机森林模型也存在着一定误差，但是在各方面相比较之下，随机森林模型最适合解决PUBG玩家排名预测问题。

# 程序源代码

详见PUBG\_Rank\_Prediction.ipynb