**机器学习作业1-PUBG排名预测**

**郭远帆，唐荣俊，罗乙然**

目录

[任务介绍 1](#_Toc528182075)

[任务说明 1](#_Toc528182076)

[数据集的使用 2](#_Toc528182077)

[任务目标与规划 2](#_Toc528182078)

[实现方案 2](#_Toc528182079)

[编程环境说明 2](#_Toc528182080)

[基本特征工程（线性模型） 3](#_Toc528182081)

[模型评估方法 3](#_Toc528182082)

[实验结果 4](#_Toc528182083)

[线性模型 4](#_Toc528182084)

[模型说明 4](#_Toc528182085)

[实验结果 4](#_Toc528182086)

[决策树模型 5](#_Toc528182087)

[决策树定义 5](#_Toc528182088)

[模型使用及结果 5](#_Toc528182089)

[随机森林（Random Forest） 6](#_Toc528182090)

[ 随机森林定义 6](#_Toc528182091)

[ 随机森林的使用与结果 6](#_Toc528182092)

[结论与后续工作 7](#_Toc528182093)

[附录 7](#_Toc528182094)

[参考文献 7](#_Toc528182095)

[代码 7](#_Toc528182096)

# 任务介绍

## 任务说明

本次任务来源于Kaggle数据科学竞赛网站提供的比赛项目:**PUBG Finish Placement Prediction （绝地求生:大逃杀玩家排名预测）**

比赛说明网站: <https://www.kaggle.com/c/pubg-finish-placement-prediction> 网站提供了该任务的详细说明，输出要求以及数据（包括训练集和测试集）

这是一个非常有趣的项目，竞赛网站与PUBG制作商蓝洞公司合作，获取了超过65，000场游戏的玩家数据，要求竞赛参加者根据这些数据预测玩家在比赛中的排名。

## 数据集的使用

本项目原数据集相当庞大，训练集与测试集数据量一共有超过650万条数据，每一条数据拥有25个特征。在实际使用中,对于某些模型(如随机森林，决策树等)我们需要对数据进行采样来测试模型的性能，以缩短训练时间，从而在短时间内了解多个模型的特点以及调参方法。

由于数据集庞大，提交作业时不一并提交，可于百度网盘上下载

## 任务目标与规划

我们小组以此项目作为机器学习第一次作业，该问题属于回归问题。而由于该项目本身是数据科学竞赛项目，其中必然存在数据分析与特征提取的过程（但并非主要的），我们将任务划分为几个阶段:

第一阶段: 数据缺失处理，进行简单的数据可视化分析

第二阶段: 参考竞赛网站其它竞赛参加者提供的特征工程方案，提取简单特征

第三阶段: 构建不同的机器学习模型训练，并且进行模型的评估

其中第三为主要的工作。

通过本次任务，我们将学习到:

1. 利用pandas对数据集进行预处理
2. 利用seaborn和matplotlib对数据集进行简单分析
3. 如何调用sklearn中的机器学习模型，并理解其实现原理
4. 自己实现简单的模型并与sklearn库进行比较

## 小组分工

郭远帆:

完成基本特征工程以及针对线性模型优化的特征工程

实现梯度下降法线性回归模型

利用sklearn的线性回归模型进行多次实验并进行比较分析

唐荣俊:

完成对决策树模型的特征工程

利用sklearn的决策树模型进行多次实验和调参，并进行比较分析

罗乙然:

完成对随机森林模型的特征工程

利用sklearn的随机森林模型进行多次实验和调参，并进行比较分析

# 实现方案

## 编程环境说明

编程语言: Python3.6.4

使用的工具包:

Numpy: 线性代数库，支持强大的矩阵运算

pandas: 数据处理库

Matplotlib: Matlab风格的画图工具

Seaborn: 统计画图工具

Sklearn: 最大的机器学习库，包含了一些常用的机器学习模型。

xgboost: 梯度提升库（集成学习方法）

## 基本特征工程（线性模型）

线性回归模型以模型简单、速度快而在工业界广泛应用。但是由于模型过于简单，最小二乘法的线性简单模型需要更多的特征提取工作，否则非常容易欠拟合而得到不满意的结果。在对训练集进行数据预处理以及相关性分析后，我们得到下图:

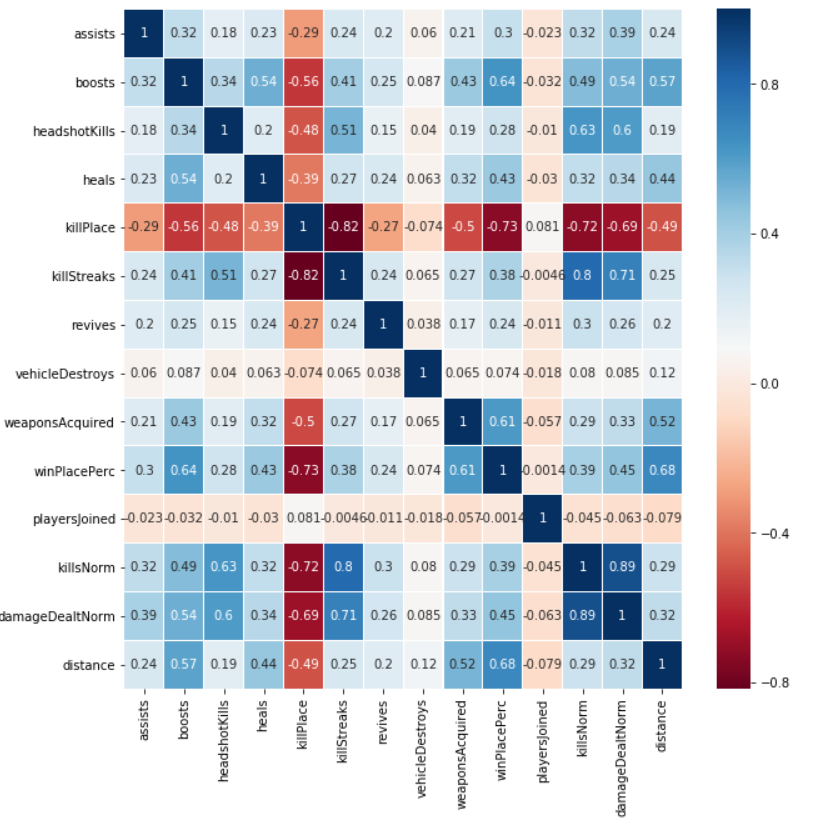


Figure1-特征相关性图

其中winPlacePerc为目标值(y)，从图中可以看到经过预处理后的特征的相关性，这些特征即线性模型中最终保留的特征。

## 模型评估方法

根据竞赛要求，该任务最终的模型评估方法为平均绝对值误差(mean absolute error)，根据模型不同还有不同的评价方式，详见实验结果中的相关描述。

# 实验结果

## 线性模型

### 模型说明

sklearn库提供了多种线性模型，在本次实验中使用到其中三种: LinearRegression, Lasso, Ridge。

LinearRegression的原理十分简单，设**X**为输入数据集，**y**为预测结果，**w**为各特征权重，其优化目标为:

在sklearn库中，其训练复杂度为, 其中n为样本数量，p为特征量。根据[1]中对sklearn的 LinearRegression()源码解读，解线性回归方程参数时，首先判断训练集**X**是不是稀疏矩阵，如是，就用Golub & Kahan双对角线化过程方法来求解；否则就调用C库LAPACK中的用基于分治法的奇异值分解来求解。

sklearn中的Lasso和Ridge是引入了L1和L2正则化项的线性回归模型，即优化目标分别为:

和LinearRegression一样，在sklearn中采用双对角线化 或者 奇异值分解来进行优化，其计算复杂度也为

由于sklearn中的线性模型没有使用梯度下降法实现，为了实践课内学习到的理论，自己实现梯度下降法(GDLinear.py)并在下一部分与sklearn实现的模型进行比较。

### 实验结果

在完成一轮特征工程后，数据集中留下来的特征拥有较好的独立性，并且对于简单模型来说降低了数据的复杂度，因此也降低了拟合的难度。我们采用不同的线性模型进行拟合并观察实验结果，并尝试着对实验结果进行解释。

调用sklearn 中的几个线性模型进行训练以及预测（采用全局数据）并对结果进行比较:

**实验结果**:

|  |  |  |  |
| --- | --- | --- | --- |
| 模型: | 数据集 | 测试集比例 | M（平均绝对误差） |
| LinearRegression | train.csv(49M) | 0.25 | 0.095 |
| Lasso | train.csv(49M) | 0.25 | 0.095 |
| Ridge | Train.csv(49M) | 0.25 | 0.095 |

从表中可得，线性回归模型平均将引入9.5%的误差。正则化项对模型的影响不大

鉴于该模型的低成本性，9.5%误差的结果可以接受。

接下来考虑自己实现的梯度下降法的线性回归模型（未引入正则化）:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 学习速率( | 数据集 | 测试集比例 | 最大迭代次数 | M(平均绝对误差) |
| 0.000001 | Train.csv.sample(1000) | 0.25 | 5000 | 0.0835 |
| 0.00001 | Train.csv.sample(1000) | 0.25 | 5000 | 0.1209 |
| 0.001 | Train.csv.sample(1000) | 0.25 | 5000 | NaN |

自己实现的梯度下降法线性模型还需要非常大的改进，目前的梯度下降法只能在小型数据集上进行测试，无法像sklearn中实现的线性回归模型一样高效快速。而且，即使对数据集进行了归一化处理，学习速率的调整依然非常困难，很容易出现无法收敛的情况。

从与sklearn中的线性模型对比，可以得出:

1. 理论计算和实际中数值解的求取还是有差距的，sklearn中的方法十分具有实际意义和工程性,其基于数值解的优化方式值得学习
2. 梯度下降法虽然实现简单，但拥有许多缺点，例如对学习速率超参数的强烈依赖。

## 决策树模型

### 决策树定义

决策树(Decision Tree）是在已知各种情况发生概率的基础上，通过构成决策树来求取净现值的期望值大于等于零的概率，评价项目风险，判断其可行性的决策分析方法，是直观运用概率分析的一种图解法。由于这种决策分支画成图形很像一棵树的枝干，故称决策树。[机器学习](https://baike.baidu.com/item/%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0" \t "_blank)中，决策树是一个预测模型，代表的是对象属性与对象值之间的一种映射关系。树中每个节点表示某个对象，而每个分叉路径则代表的某个可能的属性值，而每个叶结点则对应从根节点到该叶节点所经历的路径所表示的对象的值。

### 模型使用及结果

决策树模型在机器学习中既可以用作分类算法，也可以用作回归算法。这里我们采用它的回归形式来对结果做出预测，并使用scikit-learn中的DecisionTreeRegressor库进行回归预测。该库使用的是CART算法，是最优的决策树算法。

通过查阅资料，我们发现决策树回归模型有很多个参数需要调节。一些参数，如criterion、splitter等，采用默认的参数即可。而一些参数则需要根据我们数据的实际情况进行调节。由于我们的数据量大（共几百万条），学习的时间成本是必须考虑的问题。而且我们经过了数据预处理，忽略掉了一些显然无用的特征，例如groupId、winPoints，也按一定规则合并和归一化了一些特征，例如kills，实际上的特征量已经减少了很多，故而无需考虑最大特征数。所以经过考虑，我们最终确定了决策树最大深度max\_depth、内部节点再划分所需最小样本数min\_samples\_split、叶子节点最少样本数min\_samples\_leaf等三个参数进行调节。并且通过查阅资料，最终我们确定了它们的值分别是15、10、10。

模型在测试集上的表现为M=0.06420，M为平均绝对值误差。误差较小，可以尝试继续改进算法及参数降低误差。

## 随机森林（Random Forest）

**随机森林定义**

随机森林是集成学习的一种，是Bagging的一个扩展变体。随机森林在以决策树为基学习器构建Bagging集成的基础上，进一步在决策树的训练过程中引入了随机属性选择。具体来说，传统决策树在选择划分属性是是从当前结点的属性集合中选择最优属性；而在随机森林中，对基决策树的每个结点，先从该结点色属性集合中随机选出一些结点，再从选出的这些结点中选择最优的属性用于划分。因此，个体学习器的“独立性”得到了提升，所以泛化能力得以提高。

**随机森林的使用与结果**

Python的机器学习库scikit-learn中提供了用于回归和分类的随机森林的模型（RandomForestRegressor与RandomForestClassifier），由于本次任务是回归问题，所以我们采用RandomForestClassifier。

在第一个线性模型中我们进行了一定量的特征工程，并且丢掉了相当一部分特征，这是因为线性模型对于特征的独立性有一定要求，而在随机森林当中，因为要保证每棵决策树之间有差异性，所以保留更多在线性模型中被丢掉的重要程度次之的特征反而会提高模型的性能。我们的试验也验证了这一点：在模型的参数完全一样的前提下，使用丢掉很多“不重要”特征的训练集训练出的模型性能上是不如保留更多特征的训练集的。这也使我们更加明白针对不同模型的特点要采用不同的特征工程。

RandomForestClassifier模型中有若干参数，其中n\_estimators（随机森林中的决策树数量），max\_features（对每个结点寻找最优属性时要考虑的特征数量），min\_samples\_split（拆分内部节点所需的最小样本数）与min\_samples\_leaf（叶子结点所需最小样本数）几个参数对模型性能影响较大，因此需要进行调参。

n\_eatimators理论上说越多越好，因为随着个体学习器数目的增加，随机森林通常会收敛到更低的泛化误差。但是由于处理器计算能力有限，更多的个体学习器会耗费更多的时间来训练模型。在进行了多次试验后，我们选择n\_estimators=50，在保证模型性能的前提下，有相对较短的训练时间。

max\_features也会影响到模型的性能以及训练时间，如果取值太小每一棵决策树的效果不佳，而如果取值太大，则会导致随机森林中树缺乏随机性，使泛化能力下降。西瓜书中提到一般情况下推荐使用log2，但我们实际使用下来发现效果并不好，原因是我们的数据集中特征数量本来不多（小于30），取log2之后只有4个特征，这对一棵决策树来说肯定是不够的。因此我们排除了log2，同样也排除了sqrt（算术平方根），经过多次试验，最终取值max\_feature=20。

min\_samples\_split和min\_samples\_leaf这两个参数限制了决策树的生长。如果取值太大则会让决策树生长受限，性能不佳，而如果取值太小则会使模型更容易捕捉到训练数据中的噪声，同样影响模型性能。根据训练集的数量级并经过多次试验，最终选值min\_samples\_split=100，min\_samples\_leaf=50。

最后，关于随机森林中的模型性能评估，我们选择用两个值来评估模型的性能：一个是oob\_score（袋外得分），他利用了随机森林在生成不同训练集时约有36.8%的训练集不会被选中的特点，将未被选中作为训练集的数据作为测试集，用来评估模型的性能；另一个mean\_absolute\_error（平均绝对误差），是手动使用scikit-learn库中的train\_test\_split函数将训练集的30%划分为测试集，在模型训练成功以后，使用测试集预测最终排名，并与真实排名比较，求平均绝对误差。

最终使用以上参数的模型oob\_score得分0.925，mean\_absolute\_error为0.05997，相比于线性模型和决策树有一定提升。

# 总结与感受

### 总结与后续工作

本文介绍了我们小组使用的不同非深度机器学习模型在同一数据集上的表现以及对其原理的简单分析，总结可得线性模型速度最快，但相应地需要很好的特征工程，与此对比，更加复杂的模型则可以对无关特征的容忍度高一些，但需要以计算时间为代价。

### 小组成员第一次大作业感受

**郭远帆:**

本次作业可以说是收获良多啊！原本只是偶然在Kaggle数据科学竞赛网站上看到这么一个有趣的竞赛，就想着机器学习第一次大作业就做这个吧，没想到竟然找到了两个队友并且一起完成了这个作业。

在本次作业中我的分工是基本特征工程以及线性模型。原本只是想着利用sklearn中的线性模型进行求解并且评估其性能，当发现sklearn中的线性模型并非基于梯度下降法之后，就打算自己实现一个使用梯度下降法进行训练的线性回归模型并与sklearn的模型进行比较。

实际证明自己被sklearn完爆！自己设计的模型在1000个数据上都需要跑接近5分钟，而sklearn基于数值解的优化方法能够在完整数据集(490万条数据)上有着10秒内跑完的优秀结果，这中间差距还相当的大！在实现的过程中也是漏洞百出，调了不少bug最终才实现了一个普普通通的模型。

但不管怎么样，通过本次作业，我们小组每个成员都对sklearn机器学习库有了一定的了解，对各个模型的调参方法懂了不少。从工程意义上来说，这是一次十分成功的学习(即使在自己实现模型的过程中遇到了不少坎坷)，我们看到了实际和理论的差距，并且尝试着去深入理解，收获了不少知识。

# 附录

## 参考文献

[1]<https://blog.csdn.net/ybdesire/article/details/67701289> sklearn中的LinearRegression关键源码解读

[2]<http://scikit-learn.org/stable/user_guide.html> sklearn库官方文档

[3]<https://en.wikipedia.org/wiki/Singular_value_decomposition#Solving_homogeneous_linear_equations> 维基百科-奇异值分解

[4] <https://blog.csdn.net/y0367/article/details/51501780?utm_source=blogxgwz2> 随机森林算法

[5] <https://blog.csdn.net/qq_16633405/article/details/61200502> 随机森林算法以及参数调优