数据挖掘算法实现与实战

* 数据分析及实践exp4

PB19020499 桂栋南

目录

[1. KNN算法实现 3](#_Toc73101278)

[1.1. 一维KNN算法 3](#_Toc73101279)

[1.1.1. 算法思想 3](#_Toc73101280)

[1.1.2. 模型检验 3](#_Toc73101281)

[1.2. N维KNN算法 3](#_Toc73101282)

[1.2.1. 算法思想 3](#_Toc73101283)

[1.2.2. 模型检验 3](#_Toc73101284)

[1.3. 算法的图形化展示 4](#_Toc73101285)

[1.3.1. 原有数据的分布（以两队的总经济为例） 4](#_Toc73101286)

[1.3.2. KNN算法的展示 4](#_Toc73101287)

[1.4. 交叉验证 5](#_Toc73101288)

[1.5. 假设检验 5](#_Toc73101289)

[2. 游戏时长预测实战 5](#_Toc73101290)

[2.1. 数据预处理与分析 5](#_Toc73101291)

[2.2. 特征工程 6](#_Toc73101292)

[2.2.1. 特征选取 6](#_Toc73101293)

[2.2.2. 选取特征与预测目标的图形化展示 6](#_Toc73101294)

[2.2.3. 重大发现 6](#_Toc73101295)

[2.2.4. 数据分布的情况 8](#_Toc73101296)

[2.3. 线性回归 8](#_Toc73101297)

[2.3.1. 实现 8](#_Toc73101298)

[2.3.2. 训练结果 9](#_Toc73101299)

[2.4. KNN算法 9](#_Toc73101300)

[2.4.1. 实现 9](#_Toc73101301)

[2.4.2. K值选择 10](#_Toc73101302)

[2.4.3. 训练结果 11](#_Toc73101303)

[2.5. 决策树（C4.5）算法 11](#_Toc73101304)

[2.5.1. 实现 11](#_Toc73101305)

[2.5.2. 实验结果 11](#_Toc73101306)

[2.6. 贝叶斯方法 11](#_Toc73101307)

[2.6.1. 想法 12](#_Toc73101308)

[2.6.2. 实现 12](#_Toc73101309)

[2.6.3. 实验结果 12](#_Toc73101310)

# KNN算法实现

## 一维KNN算法

### 算法思想

由于一维的KNN算法与多维算法相比有一些为读书的特别，单独写了一个程序，对单一特征进行分类。

分类的根据是特征间距离的绝对值。也就是距离差的2范数。

根据特征间的距离找出前k项取label的平均值作为预测数据点的特征（以2分类问题为例）。

### 模型检验

利用两队的经济差这一特征来对一维的KNN算法进行验证。

使用4:1的比例进行验证，取前80%的数据作为训练集，剩余20%的数据作为测试集，检验KNN算法的正确率。

得到一维KNN算法在经济差这一特征上对team1\_win这一特征的预测正确率为98%。

## N维KNN算法

### 算法思想

根据输入的数据集的维数确定数据的维数，对长度不同或者维数不匹配的情况进行出错处理。

距离的刻画利用测试点和数据集中点（其实是两个向量）的欧氏距离来表示。

根据特征间的距离找出前k项取label的平均值作为预测数据点的特征（以2分类问题为例）。

### 模型检验

二维测试，测试特征是经济差和first\_inhibitor。

值得注意的是，对两个特征的值需要进行归一化。据我观察，似乎sklearn的包里包含了归一化的步骤，所以导致不能指定两个特征的权重，因为归一化后完全没有权重的概念。

而我的程序里，将归一化拿出了KNN程序，这样可以指定数据的权重，做一些自定义比例的缩放。虽然好像自己也没有进行验证。

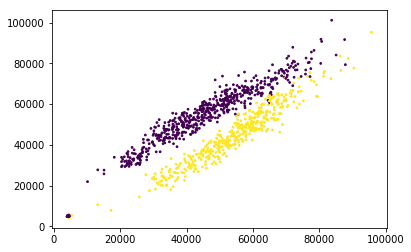
使用4:1的比例进行验证，取前80%的数据作为训练集，剩余20%的数据作为测试集，检验KNN算法的正确率，正确率为96%。

## 算法的图形化展示

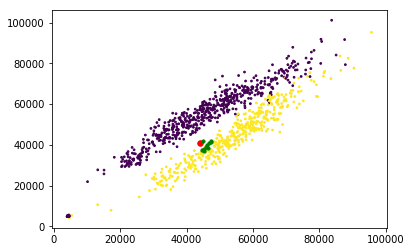
### 原有数据的分布（以两队的总经济为例）

首先利用散点图画出两队总经济的分布情况。根据team1\_win=True和team1\_wni=False两种情况进行分类，如下图所示。

横纵坐标分别为两队的总经济值。为了更清晰的展示，以前1000条数据作图。



### KNN算法的展示



图中的红色点是测试的点，绿色的点是训练集中距离该测试点最近的k个点，所以可以知道，红色点所表示的测试点的训练结果为绿色点的label，也即黄色点的label，也即team1\_win=False。

## 交叉验证

试用k折交叉验证来对KNN算法准确性进行预测。

以单一特征经济差作为测试的特征（这一特征的正确性很高）。

由于k折检验对于80000条数据跑的速度有些慢，故采取前8000条数据进行检验。

每次随机抽取1-1/k的数据作为训练集，剩余的1/k的数据作为测试集，利用测试集检验KNN算法的准确性。

实验的结果表明在10折检验中，KNN算法的准确性达到97.8%。

## 假设检验

获取k折检验得到的k个准确性进行假设检验。采用方差未知的单样本t检验。

调用了scipy.stats.ttest\_lsamp包，得到的结果是有97%的把握认为预测准确，但没有98%的把握认为预测准确。

也即有把握认为预测模型有97%的准确率。

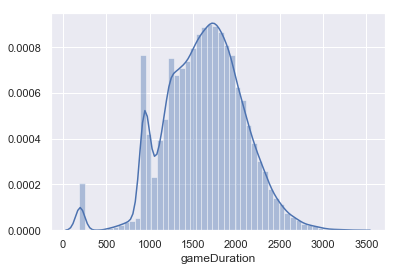
# 游戏时长预测实战

## 数据预处理与分析

首先导入数据，并利用merge函数合并数据。

显然任务目标是通过20000~80000的数据的游戏时长来预测0~20000的数据的游戏时长。

首先看一下游戏时长的分布情况。



## 特征工程

### 特征选取

由游戏的常识可知，游戏时长越长，总的经济量越高，总击杀数、辅助数、死亡数等也越多。

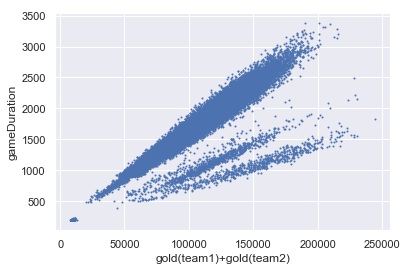
而关于team的所有数据对于预测游戏时长无用，因为两队总有一个队伍赢、一个队伍拿下一血，龙，水晶，与游戏时长无关。

由于击杀数、死亡数等与游戏时长的关系没有十分明显，且变动范围较小。故采用总经济作为特征，而又消除了不同role对预测的影响。

更具体地说，一维特征采取两队的总经济，而二维特征，则分别采取两队的总经济。

### 选取特征与预测目标的图形化展示

下图即为两队总经济这一单一特征与游戏时长之间的关系。采用散点图展示。

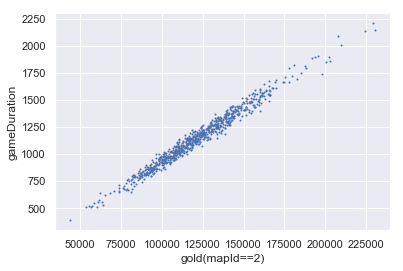


### 重大发现

上图中好像产生了不同的线性分布情况。所以猜测是不同的mapId导致了不同斜率的回归现象。

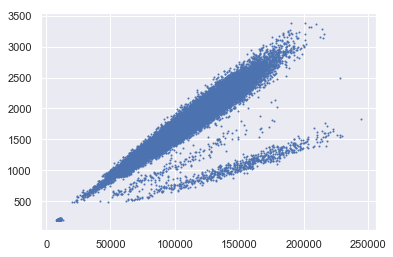
从而检验一下不同的mapId数据。

在数据较少的map12，发现了良好的线性关系。



但是mapId==11的数据仍然是不同的分布。

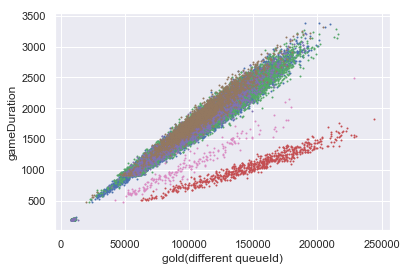
只是相比于之前的图像，中间的线性趋势点减少了。



一个想法，可能是queueId的不同导致了这种现象，于是分离各个queueId的数据。

所以分离了不同的queueId数据，得到了下图。

Amazing！竟然各个queueId的数据都呈现了比较明显的线性分布，可以更加精确地做预测了。



### 数据分布的情况

为了防止某些queueId只在后60000条数据中出现，大致看一下不同queueId的分布情况。

结论：

queueId为420的分布为: 数据集37107个，测试集13038个

queueId为450的分布为: 数据集764个，测试集235个

queueId为430的分布为: 数据集16148个，测试集4818个

queueId为900的分布为: 数据集677个，测试集346个

queueId为440的分布为: 数据集3142个，测试集874个

queueId为700的分布为: 数据集1953个，测试集641个

queueId为1020的分布为: 数据集208个，测试集48个

分布还是比较均匀的~~（我甚至怀疑助教人为打乱了原始顺序数据集）~~。

所以可以放心的进行调包预测了。

## 线性回归

### 实现

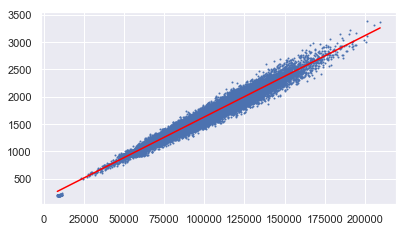
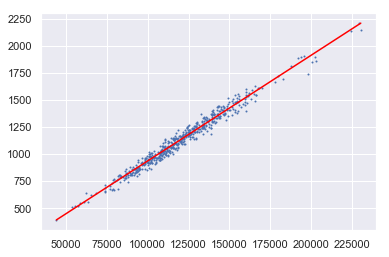
直接调用sklearn.linear\_model. LinearRegression库。

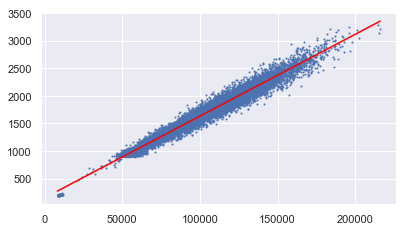
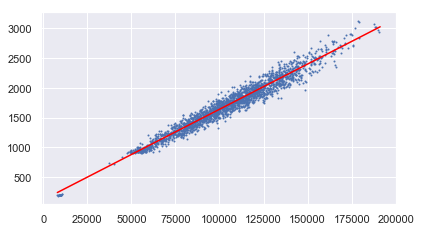
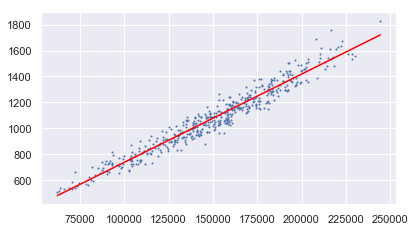
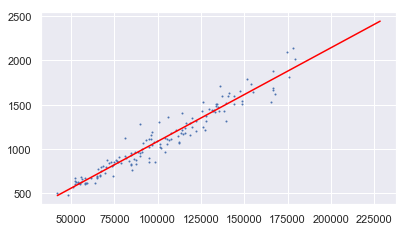
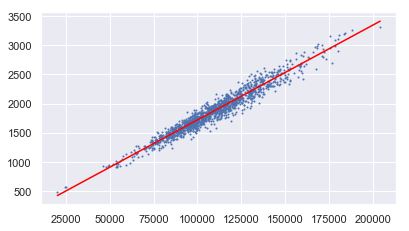
Reg.fit()进行训练集训练，reg.predict()进行测试集的测试。训练和测试都是在整个的训练集上划分出来的子类，不是真正的预测测试集。

### 训练结果

分别对不同queueid的数据进行了拟合，拟合的效果r^2值都达到了90%以上，线性拟合的效果比较良好。

下面是拟合示意图，横坐标分别是不同queueId的总经济，纵坐标都是游戏时长。



且最后的训练误差的MSE为6000+，相当于平均预测误差70~80s，在可以接受的范围。

最后输出真实的预测结果。

## KNN算法

### 实现

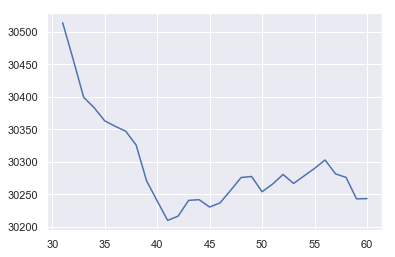
直接调用sklearn.neighbors. KNeighborsRegressor方法。

Knn.fit(training\_data)+knn.predict(test\_data)即可。

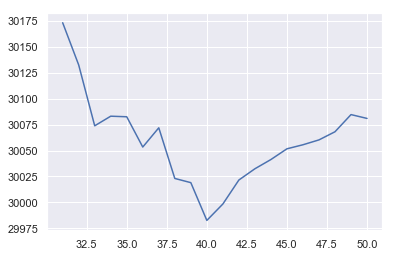
### K值选择

对于不同的特征，选取的k值不同。所以对每个特征画出不同k值与MSE之间的关系，从而确定合理的k值。

对于单特征：选取k=41左右

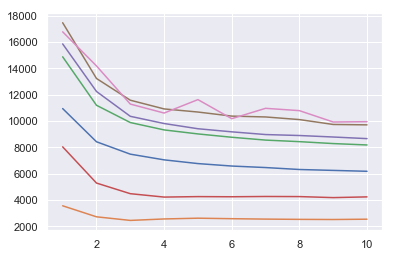


对于双特征：选取k=40左右



但是很明显，上述两者的纵坐标——误差太大了。

对于每个queeuId使用：可以看出，在k=10左右，误差已经基本上稳定下来，所以最终采取k=10作为参数跑KNN算法。



### 训练结果

单一特征两队总经济，效果太差，总误差30000+。

特征是两队总经济和queueId，效果优于单特征的KNN算法,但还是很差。

对于每个queueId应用KNN算法，效果比较理想，与线性回归算法相似。

最后对真正的预测集合预测，保存文件。

## 决策树（C4.5）算法

### 实现

直接调用sklearn.tree.DecisionTreeClassifier即可。

采用c45.fit(training\_data),利用c45.predict(test\_data)作为预测。

### 实验结果

单特征和双特征误差太大，50000+，20000+。

分别对不同queueId采用双特征，分别使用gini和entropy来作为信息增益的评价指标来进行预测任务。

Gini的误差9000+，entropy的误差10000+，效果都不如前两种方法，故舍弃。

## 贝叶斯方法

### 想法

可以看到gameDuration的分布可能可以使用正态分布来进行拟合，故采用GaussianNB方法来进行贝叶斯方法的建模。

### 实现

直接调用sklearn.naive\_bayes. GaussianNB即可。

Gnb.fit(training\_data)训练数据，gnb.predict(test\_data)预测数据。

### 实验结果

对不同的queueId分别使用，效果不如KNN和linear regression方法，误差9000+，故舍弃。