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
   1.1. 一维 KNN 算法 3
      1.1.1. 算法思想 3
      1.1.2. 模型检验 3
   1.2. N 维 KNN 算法 3
      1.2.1. 算法思想 3
      1.2.2. 模型检验 3
   1.3. 算法的图形化展示 4
      1.3.1. 原有数据的分布（以两队的总经济为例） 4
      1.3.2. KNN 算法的展示 4
   1.4. 交叉验证 5
   1.5. 假设检验 5
2. 游戏时长预测实战 5
   2.1. 数据预处理与分析 5
   2.2. 特征工程 6
      2.2.1. 特征选取 6
      2.2.2. 选取特征与预测目标的图形化展示 6
      2.2.3. 重大发现 7
      2.2.4. 数据分布的情况 8
   2.3. 线性回归 9
      2.3.1. 实现 9
      2.3.2. 训练结果 9
   2.4. KNN 算法 10
      2.4.1. 实现 10
      2.4.2. K 值选择 10
      2.4.3. 训练结果 12
   2.5. 决策树（C4.5）算法 12
      2.5.1. 实现 12
      2.5.2. 实验结果 12
   2.6. 贝叶斯方法 12
      2.6.1. 想法 13
      2.6.2. 实现 13
      2.6.3. 实验结果 13

1. KNN 算法实现

1.1. 一维 KNN 算法

1.1.1. 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）。

1.1.2. 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%。

1.2. N 维 KNN 算法

1.2.1. 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）。

1.2.2. 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。

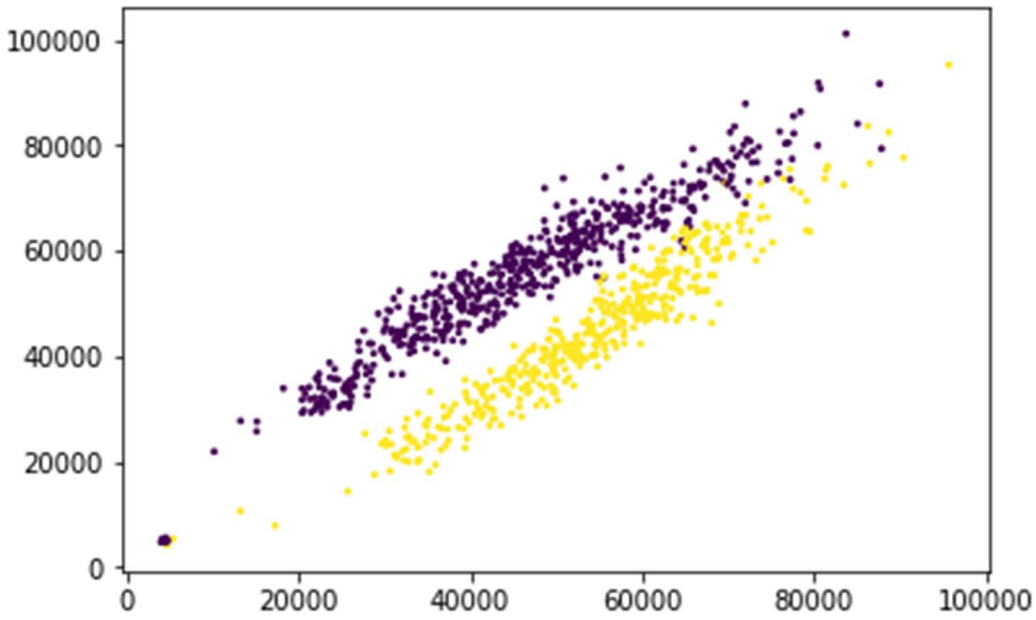
使用 4:1 的比例进行验证，取前 80%的数据作为训练集，剩余 20%的数据作为测试集，检验 KNN 算法的正确率，正确率为 96%。

1.3. 算法的图形化展示

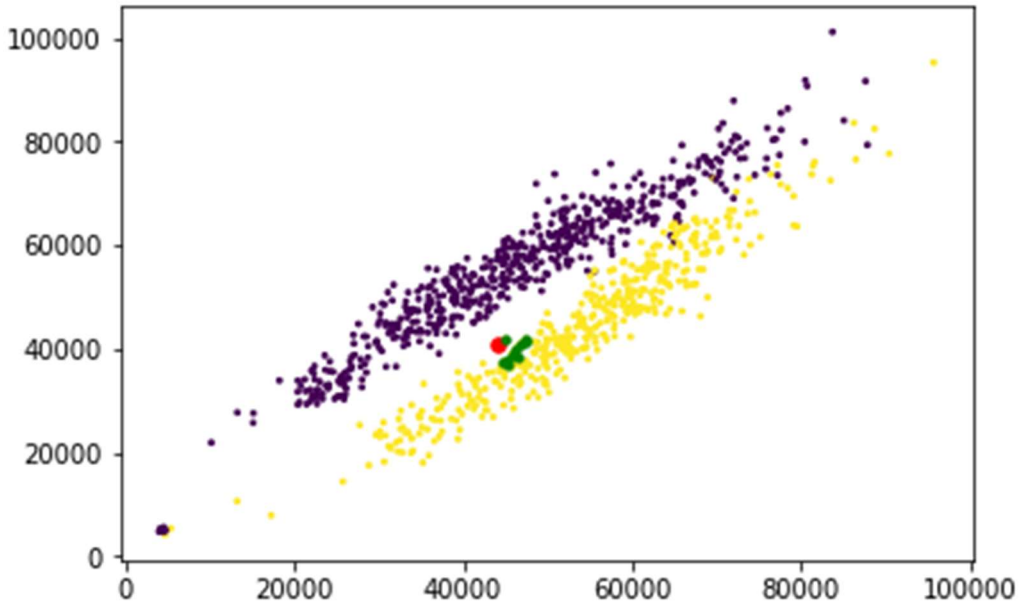
1.3.1. 原有数据的分布（以两队的总经济为例）

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。



1.3.2. 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。

1.4. 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%。

1.5. 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。

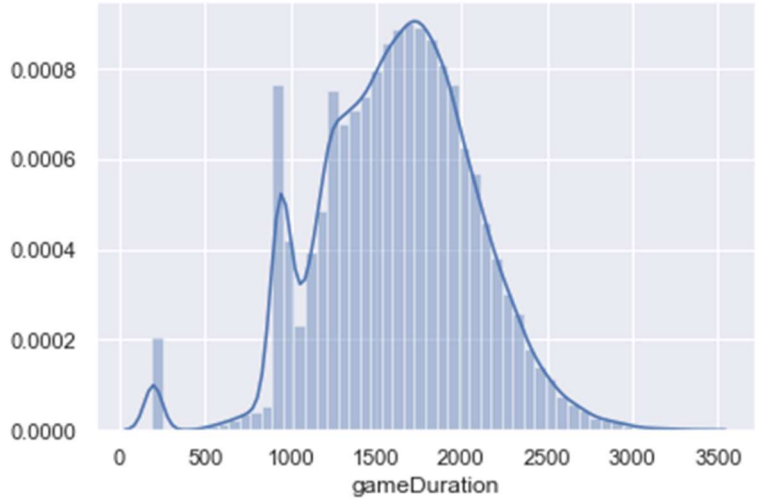
2. 游戏时长预测实战

2.1. 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。



2.2. 特征工程

2.2.1. 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。

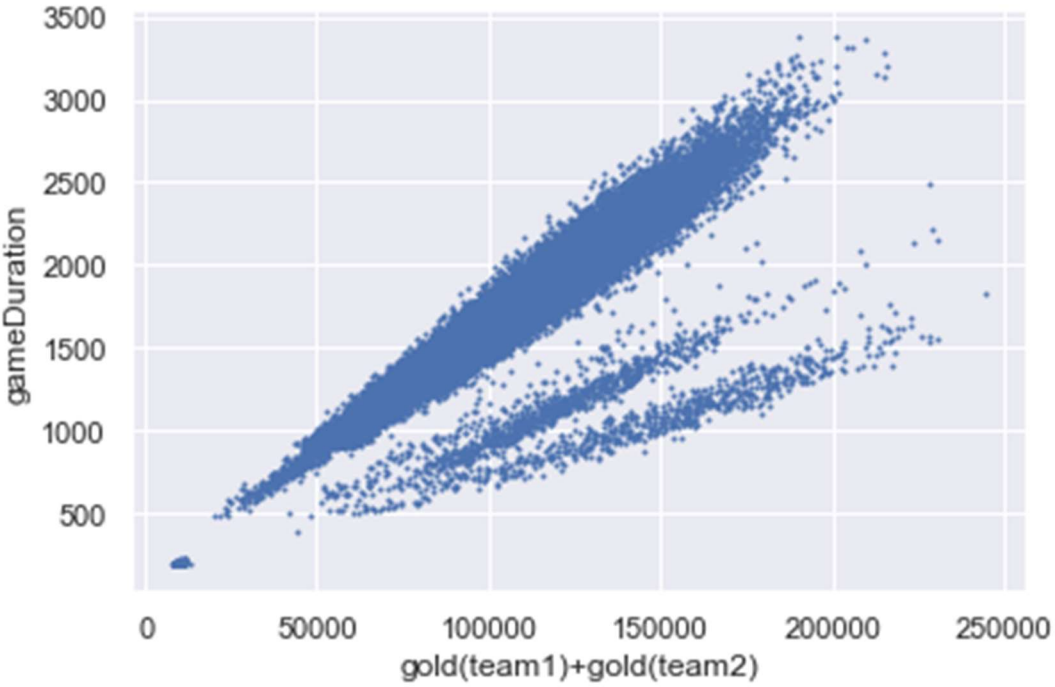
而关于 team 的所有数据对于预测游戏时长无用，因为两队总有一个队伍赢、一个队伍拿下一血，龙，水晶，与游戏时长无关。

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。

2.2.2. 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。

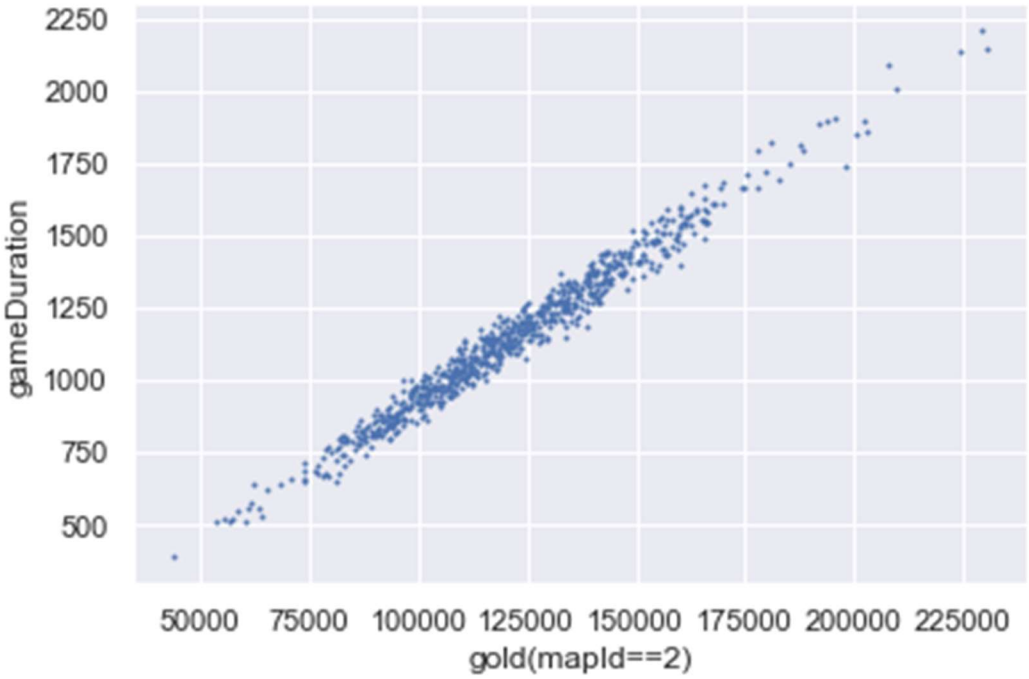


2.2.3. 重大发现

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。

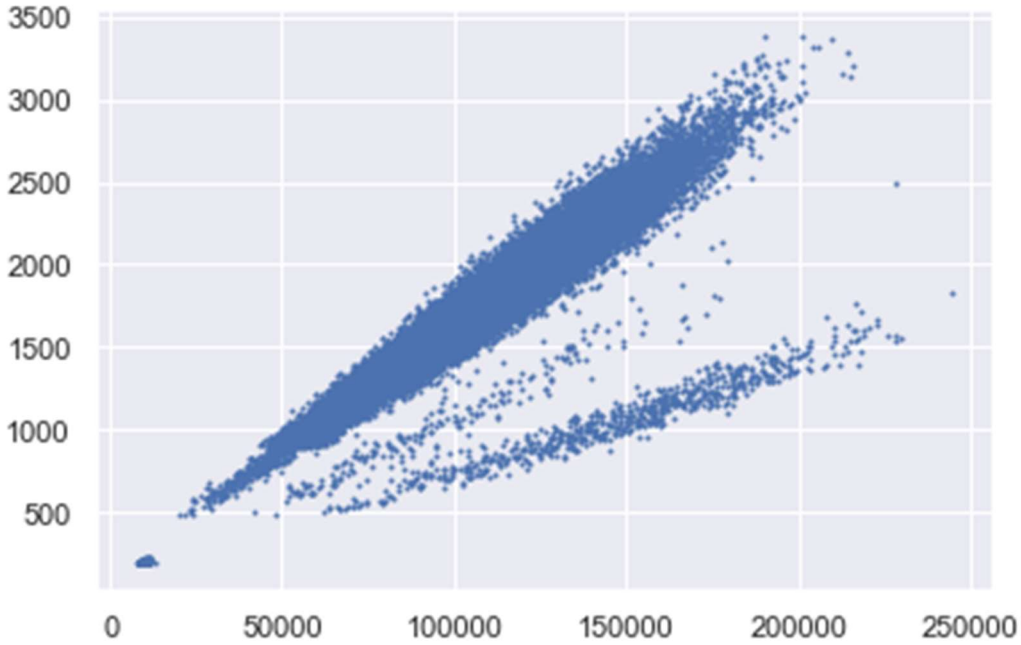
从而检验一下不同的 mapId 数据。

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。

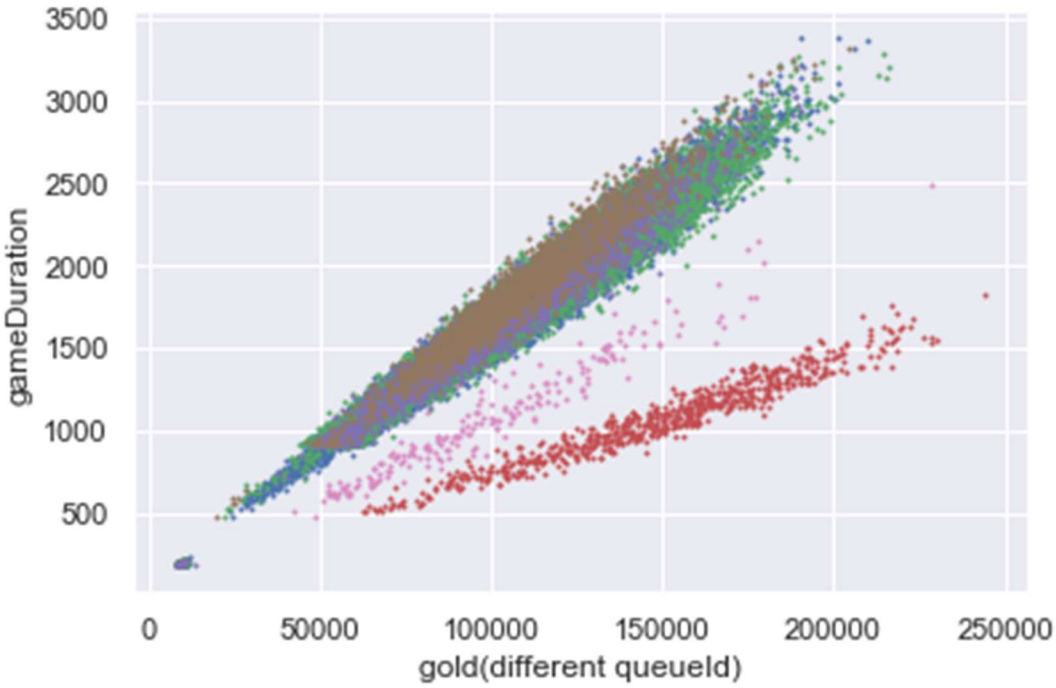
只是相比于之前的图像，中间的线性趋势点减少了。



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。



2.2.4. 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。

2.3. 线性回归

2.3.1. 实现

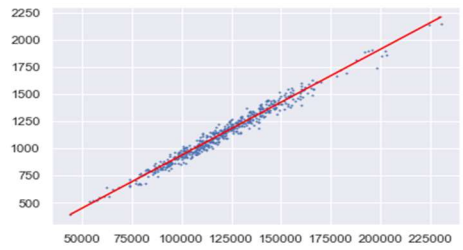
直接调用 sklearn.linear_model. LinearRegression 库。

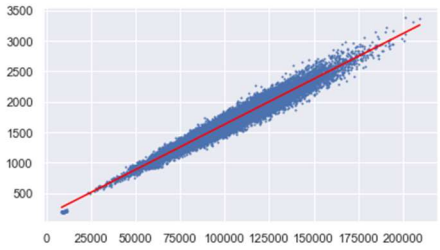
Reg.fit()进行训练集训练，reg.predict()进行测试集的测试。训练和测试都是在整个的训练集上划分出来的子类，不是真正的预测测试集。

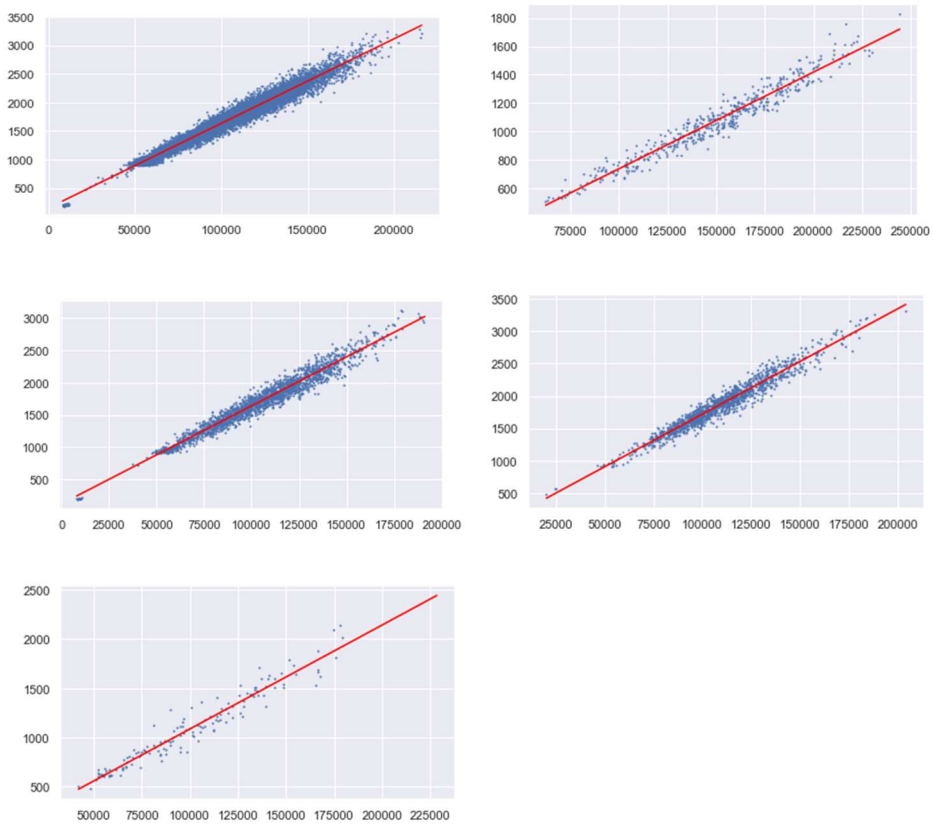
2.3.2. 训练结果

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。

2.4. KNN 算法

2.4.1. 实现

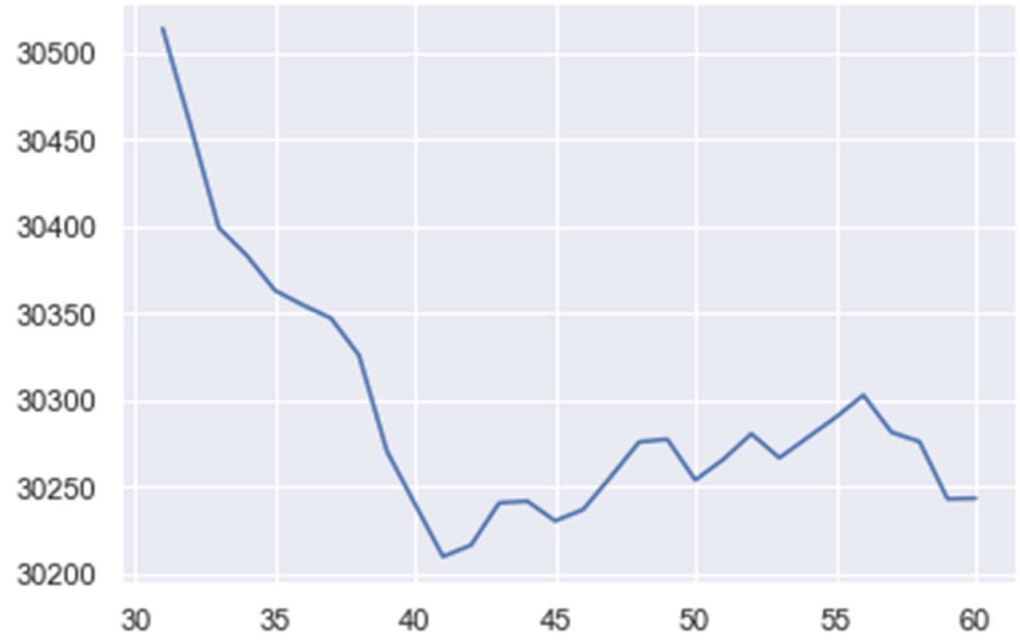
直接调用 sklearn.neighbors. KNeighborsRegressor 方法。

Knn.fit(training_data)+knn.predict(test_data)即可。

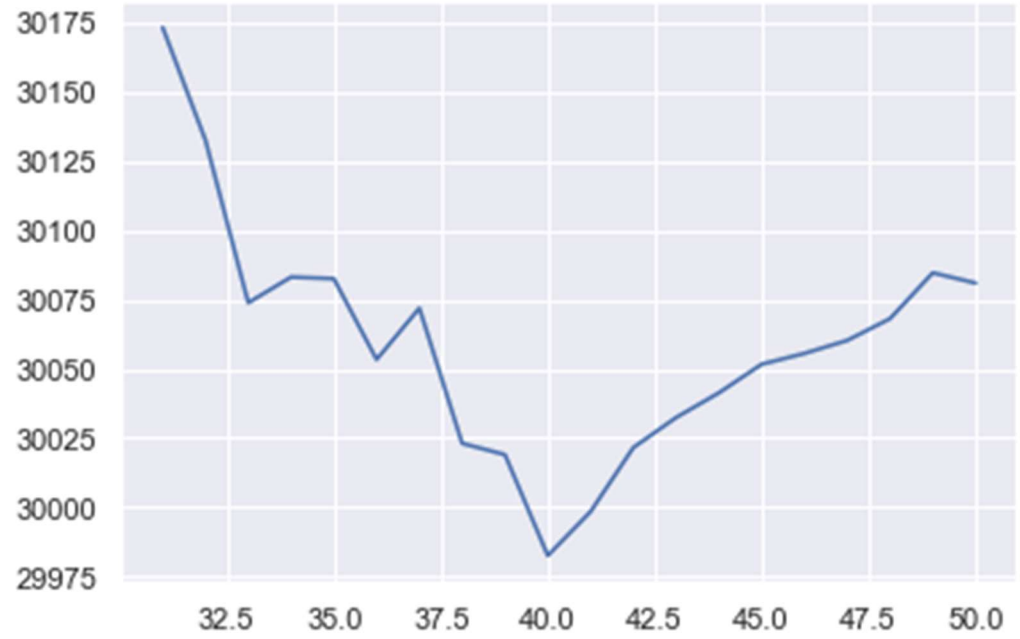
2.4.2. K 值选择

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

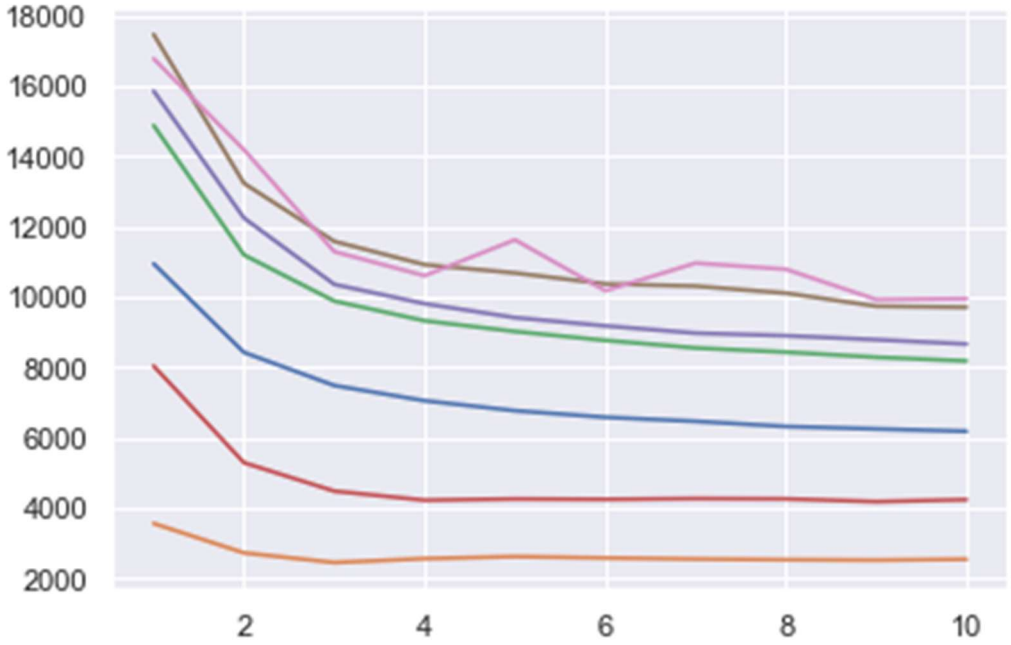


对于双特征：选取 k=40 左右



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。



2.4.3. 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。

2.5. 决策树（C4.5）算法

2.5.1. 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。

2.5.2. 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。

2.6. 贝叶斯方法

2.6.1. 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。

2.6.2. 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。

2.6.3. 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。