数据分类与聚类

实验报告

2018-12-12

钟春蒙 2018214147
崔浩 2018214160

目录

一、银行直销数据分类（钟春蒙） **3**

1. 作业要求 3

2. 分类评价方法 3

3. 实验环境 3

4. 决策树分类 4

5. 朴素贝叶斯分类 5

6. K-NN 分类 7

7. 逻辑回归分类 8

8. 四种分类方法比较 9

二、青蛙叫声聚类分析（崔浩） **11**

1. 实验环境 11

2. 实验设计 11

2.1 MFCC 与距离度量方式 11

2.2 降维和特征选择 13

2.3 算法 1：k-means+agnes 13

2.4 算法 2：dbscan+agnes 14

3. 实验结果 14

3.1 算法 1 参数对结果的影响 14

3.2 算法 2 参数对结果的影响 16

3.3 两种算法的对比 18

数据仓库大作业二

一、银行直销数据分类

1.作业要求

根据银行客户的属性判断客户是否会订阅某项定期存款业务，不限定分类算法，要求至少实现两种算法，并且对于算法精度进行分析比较

2.分类评价方法

实际 预测	Y	N
Y	a	b
N	c	d

如图所示，分别统计四种分类结果，并做如下整理：

$$
\begin{aligned}
Accuracy &= \frac{a+b}{a+b+c+d} \\
Precision &= \frac{a}{a+c} \\
Recall &= \frac{a}{a+b}
\end{aligned}
$$

考虑到本次分类的情景，为了尽可能多的找到购买服务的客户，应当在保证准确率的同时，尽可能增加真阳率，也就是增加类别为真的预测个数，甚至可以牺牲部分准确率来提高真阳率

3.实验环境

操作系统：Windows10 专业版 64位

处理器：Intel(R) Core(TM) i7-8700 @3.2GHZ

内存：16G

编程语言：Python 3.7

4.决策树分类

1）算法描述

对于所有的数值型属性，以平均值为轴，将所有的用例分为两类；对于所有的非数值型属性，每一个属性值分为一类。

以上述分类标准，为当前样例集合，计算当前属性集合中每一个属性的信息增益，找到信息增益最高的属性。如果是数值型属性，则按照大于平均值和小于等于平均值分出两棵子树；如果是非数值型数据，则为每一个属性值分出一棵子树。子树包含原样例集合中，符合分类标准的所有样例，属性集合中删除当前分类属性。递归为子树进行分类。

如果属性集合中属性数量为零，或者样例集合中样例数量为零，或者正负样例的比例达到了给定阈值，则不再对当前子树进行分类。

2）属性选择

选择了全部属性进行分类依据。

选择age， duration， campaign， pdays， previous， emp.var.rate， cons.price.idx， cons.conf.idx， euribor3m， nr.employed 作为数值型属性处理；

选择job， marital， education， default， housing， loan， contact， month， day_of_week， poutcome 作为非数值型属性处理

3）数据集选择和复杂度分析

设第i棵子树含有n_i个样例和k_i个属性，那么在找具有最大信息增益的属性时，必须遍历所有属性和所有样例，复杂度$n_i k_i$，在决策树的每一层中，所有子树的总体时间复杂度为$\sum n_i k_i = |K|n$，|K|代表当前子树的属性集合中的属性数目，因此算法总的时间复杂度是$O(nk^2)$。

复杂度较低，因此可以选择完整数据集。

4）正负例平衡方法

一般的决策树算法中，只有当子树中所有的样例都属于同一类时，也就是某一类的比例达到100%时，才停止子树的划分，但是在本次情景中，正样例非常少，因此当子树中样例集合的正例比例达没有达到100%时，在某一中间阈值，也可以停止子树的划分，并且将落到该子树的样例划分为正例。

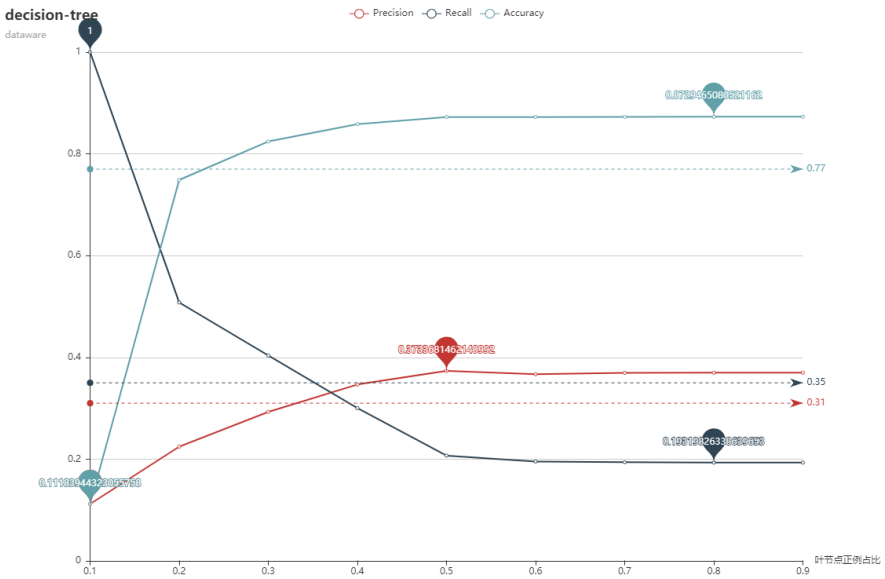
5）分类结果

对于大样本bank-additional-full，在正例占比0.6时停止子树分裂，准确度达到0.7440317229100913

对于小样本bank-additional，当正例的比例达到P时，停止子树的分类，分类的真阳率和准确度如下：

正例占比	真阳率$(a/(a+b))$	正确率
0.9	0.11940298507462686	0.866504854368932
0.6	0.11940298507462686	0.866504854368932
0.4	0.208955223880597	0.8519417475728155
0.3	0.26865671641791045	0.8252427184466019
0.2	0.3805970149253731	0.7459546925566343

对于大样本，当正例的比例达到P时，停止子树的分类，对应的Precision，Recall，Accuracy如下



5.朴素贝叶斯分类

1）算法描述

用A_{ij}表示A第i个属性的第j中值，为所有属性的所有值计算$P(A_{ij}|Y)P(A_{ij}|N)$

$$\vec{a} = [a_1, a_2, a_3, \ldots, a_{20}]$$
$$P(Y|\vec{a}) = \frac{P(a_1|Y)P(a_2|Y)\ldots P(a_{20}|Y)P(Y)}{P(a_1)P(a_2)\ldots P(a_{20})}$$
$$P(N|\vec{a}) = \frac{P(a_1|N)P(a_2|N)\ldots P(a_{20}|N)P(N)}{P(a_1)P(a_2)\ldots P(a_{20})}$$

如果$$P(Y|A_{ij}) >= P(N|A_{ij})$$，则将样例a分类为Y，否则分类为N

2）属性选择

选择了全部属性进行分类依据。

选择age，duration，campaign，pdays，previous，emp.var.rate，cons.price.idx，cons.conf.idx，euribor3m，nr.employed 作为数值型属性处理；

选择job， marital， education， default， housing， loan， contact， month， day_of_week， poutcome 作为非数值型属性处理

3）数据集选择和复杂度分析

在计算$P(A_{ij}|Y)$和 $P(A_{ij}|Y)$时，需要遍历所有n个样例的全部k个属性，时间复杂度$O(nk)$，在分类时，需要两次遍历所有样例的所有属性计算其为Y的概率和为N的概率，时间负责度为$O(2nk)$，因此总的时间负责度是$\Theta(nk)$

4）正负例平衡方法

在朴素贝叶斯分类中，正负样例的绝对数量并不影响分类的结果，因为计算$P(A_{ij}|Y)$和 $P(A_{ij}|Y)$时，已经包含了样本数量这个因素，样本数量少可能会导致容错性降低，不会导致系统性错误。

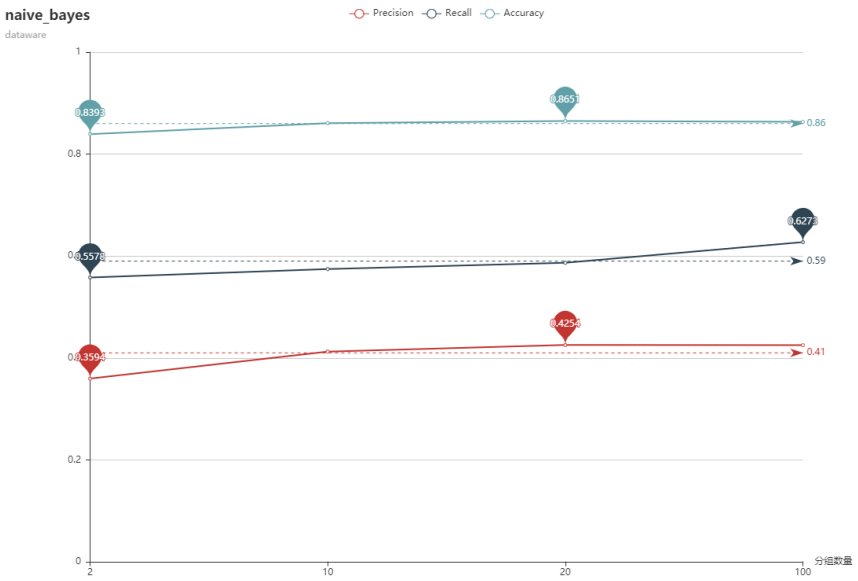
但是考虑到分类的实际情景，为了增大真阳率，适当的牺牲假阳率是可以的。因此可以通过改变分类条件，即将分类条件$$P(A_{ij}|Y) >= P(A_{ij}|N) * P,P\in [0,1)$$时将样例a分类为Y，来增大真阳率。

5）分类结果

对于小样本集bank-additional，分类正确可以达到0.8923948220064726

对于大样本集bank-additional-full，分类准确度可以达到0.6598038263571915

对于大样本，改变数值型属性的分组策略，会影响到最终的分类结果。下图是在等宽分组策略下，对应的Precision，Recall，Accuracy



6.K-NN分类

1）算法描述

选择和样例a距离最近的k个样例，如果正例占多数，则将a分类为Y，否则将a分类为N。

将样例的所有属性划分为数值型属性和非数值型属性，A_i^{max}表示第i个数值型属性的最大值，则样例a和b的在第i个属性生的距离定义为

$$dis = \begin{cases} (\frac{a_i - b_i}{A_i^{max}})^2 & 若\ A_i是数值型属性 \\ a_i == b_i ? 1 : 0 & 若\ A_i是非数值型属性 \end{cases}$$

2）属性选择

选择了全部属性进行分类依据。

选择age， duration， campaign， pdays， previous， emp.var.rate， cons.price.idx， cons.conf.idx， euribor3m， nr.employed 作为数值型属性处理；

选择job， marital， education， default， housing， loan， contact， month， day_of_week， poutcome 作为非数值型属性处理

3）数据集选择和复杂度分析

对于测试集中的每一个样例，都需要计算其与训练集中每一个样例的距离，复杂度为$\Theta(Nk)$，因此为测试集中的每一个样例分类的复杂度是$\Theta(nNk)$，这个复杂度是很高的，因此K-NN分类算法在这一情景下只适用于小样本集

4）正负例平衡方法

在K-NN分类中，k值的选择是十分重要的，特别是在正负样例数量相差较大的情景下，因为如果k选择过大，那么k个最近邻样本中一定是负样例占多数。如果是为了增大真阳率而牺牲假阳率，那么还可改变分类条件，例如k近邻样本中正例占比达到40%就判断为正例。

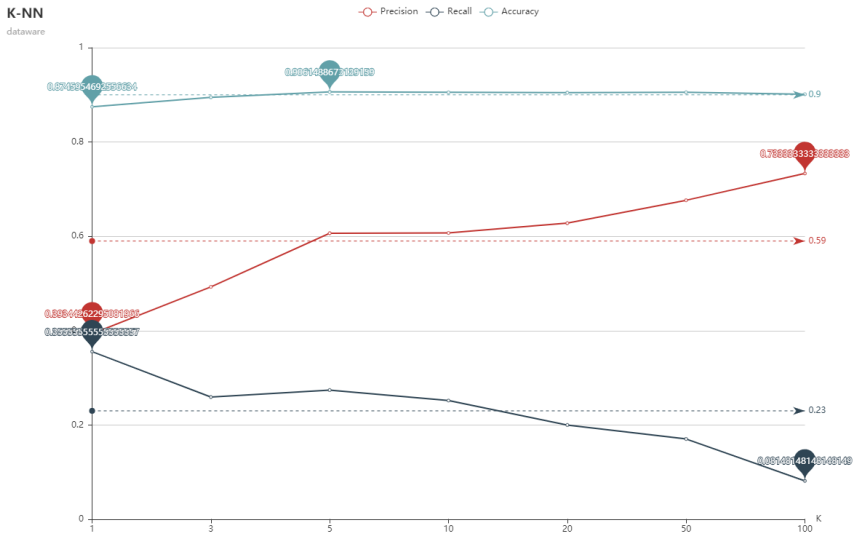
5）分类结果

k=10，只要有正例就判断为Y时，准确度为0.8358333333333333，真阳率(a/(a+b))为0.6259541984732825

k=100，正例占比达到10%就判断为Y时，准确度为0.8333333333333334，真阳率(a/(a+b))为0.7480916030534351

k=100，正例占比达到20%就判断为Y时，准确度为0.8933333333333333，真阳率(a/(a+b))为0.3969465648854962

对于大样本，在不同的K值时，K-NN分类对应的Precision，Recall，Accuracy如下



7.逻辑回归分类

1）算法描述

为每一个数值型属性设置一个权重值weight

$$result = sigmoid(\overrightarrow{cur}.transpose * \overrightarrow{weights})$$

result为1则分为Y，否则分为N

2）属性选择

逻辑回归方法仅适用于数值型数据，因此只使用了数值型属性，即age，duration，campaign，pdays，previous，emp.var.rate，cons.price.idx，cons.conf.idx，euribor3m，nr.employed 作为数值型属性处理。

3）数据集选择和复杂度分析

权重值的每一次更新迭代，都需要用当前的权重值计算每一个训练样本的类别，这一过程的复杂度是O(nk)，但是这一过程使用矩阵乘法可以快速完成。权重值的更新次数c是手动设置的，因此算法的总的时间复杂度是O(cnk)，可以用于大样本集

4）正负例平衡方法

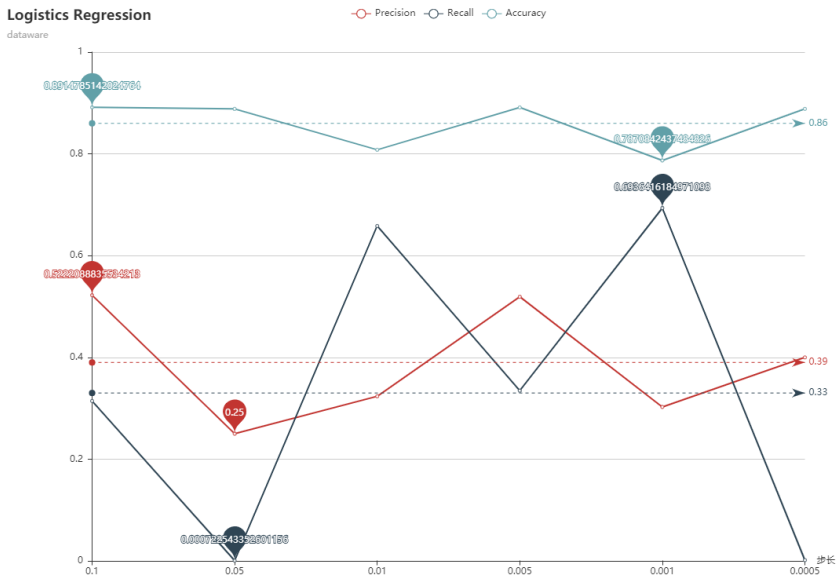
逻辑回归算法的根本思想是找到正负样例之间的分界，因此几乎不会受到样本数量的影响，同时也很难照顾样本数量因素，因此没有使用特别的方法平衡正负样例。

5）分类结果

在迭代步长为0.001，迭代次数为5000次时，对大样本的分类准确度是0.7651743226182384

在迭代步长为0.001，迭代次数为5000次时，对小样本的分类准确度是0.9087378640776699

对于大样本，不同的步长会对逻辑回归分类的结果造成影响，在不同步长下，对应的Precision，Recall，Accuracy如下



8.四种分类方法比较

1.适用范围

决策树算法，逻辑回归算法和朴素贝叶斯算法执行速度较快，适用于大数据集，而K-NN算法执行速度较慢，适用于小数据集。

决策树算法，朴素贝叶斯算法和K-NN算法都可以支持非数值型属性，但是K-NN算法不支持非数值型属性，但是必要时可以将非数值型属性转化为数值型属性。

2.准确度

四种分类方法的准确度比较

分类方法	准确度（大）	准确度（小）	真阳率（大）	真阳率（小）
决策树	0.8729465080521162	0.866504854368932	0.3812154696132597	0.25396825396825395
朴素贝叶斯	0.8609694909767743	0.88915857605178	0.3443868099296036	0.4919786096256685
$K-NN$	–	0.9053398058252428	–	0.3565217391304348
逻辑回归	0.8912357368293274	0.9087378640776699	0.3333333333333333	0.6119402985074627

3.真阳率

决策树算法、K-NN算法对正负样本的比例敏感，在正例较少的情况下，容易造成真阳率低，但是这两种算法都能够通过调整分类细节的方式来降低正负样本比例的影响。

朴素贝叶斯算法受正负样本比例影响较小，但是也能够通过调整分类细节达到牺牲假阳率，提高真阳率的效果。

逻辑回归算法受正负样本比例影响较小，且很难通过调整提高真阳率。

4.可解释性

四种分类方法均有较强的解释性。

决策树算法根据每一个非叶节点的判断标准可以知道如何一步一步进行分类，这些在每一个维度的分类标准合起来就是对整个样本的分类标准；

K-NN算法找到的是与测试样本最为相近的训练样本，根据与其最为相似的样本的类别来决定当前样本的类别；

逻辑回归算法中每一个属性的权重值与属性值范围综合考虑，代表了某一个属性的重要程度和对分类结果的贡献；

朴素贝叶斯算法直接给出了每一个属性对最终分类结果的贡献。

青蛙叫声聚类分析实验报告

一、实验环境

操作系统：Windows10 专业版 64 位
处理器：Intel(R) Core(TM) i7-8700 @3.2GHZ
内存：16G
编程语言：Python 3.7
集成环境：Anaconda3
库文件： fastdtw 0.3.2 (pip install fastdtw)

二、实验设计

1. MFCC 与距离度量方式

参考资料：

（1） Mel-frequency cepstrum https://en.wikipedia.org/wiki/Mel-frequency_cepstrum
（2） 音频特征(MFCC)提取 https://blog.csdn.net/yunnangf/article/details/78965446
（3） 梅尔频率倒谱系数 https://blog.csdn.net/zouxy09/article/details/9156785

MFCC 是一种音频特征的表现形式。声音的形状可以通过语音短时功率谱的包络中显示出来。MFCC 是描述这个包络的一种特征。

提取 MFCC 特征的步骤为：

（1） 对音频信号预加重、分帧和加窗
（2） FFT(快速傅里叶变换)得到频谱
（3） 频谱通过 Mel 滤波器组得到 Mel 频谱
（4） 在 Mel 频谱上面进行倒谱分析得到 MFCC 特征

基于以上音频特征，在计算音频数据距离度量时分别使用以下两种方法：

1.1 余弦相似度

在比较两条 MFCC 特征数据时使用该方法。将 MFCC 特征数据理解为多维空间向量，两个向量夹角的余弦值作为衡量两个个体间差异的大小。余弦值越接近 1，就表明夹角越接近 0 度，也就是两个向量越相似。

向量 x 和向量 y 的夹角的余弦计算公式为：

$$\cos(\theta) = \frac{\sum_{i=1}^{n}(x_i \times y_i)}{\sqrt{\sum_{i=1}^{n}(x_i)^2} \times \sqrt{\sum_{i=1}^{n}(y_i)^2}}$$

具体在项目中，按以下步骤计算 MFCC 特征向量 x 和 y 的距离：

（1） 求 $\mathrm{num} = \mathrm{x.T} \times \mathrm{y}$

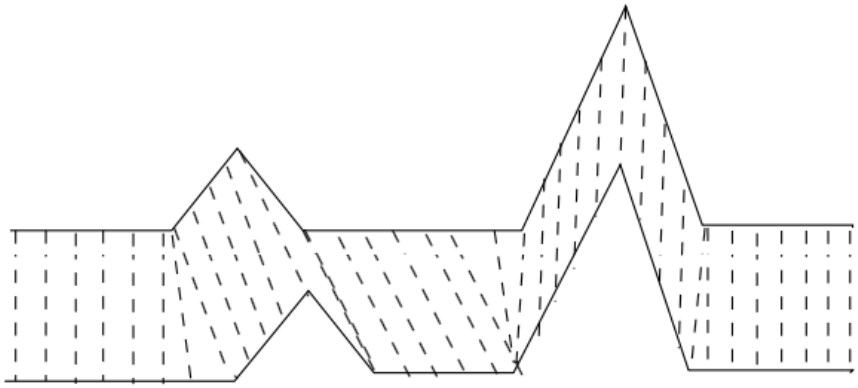
（2） 求 x 与 y 范数的乘积$\mathrm{norm} = norm(x) \times norm(y)$

（3） 计算余弦值 $\cos\theta = num \ / \ norm$

（4） 计算距离$\mathrm{distance} = 0.5 + 0.5 \times \cos\theta$

1.2 动态时间规整 DTW

DTW 常被用在语音识别领域，求解两模板匹配时累计距离最小所对应的规整函数，该算法基于动态规划的思想，解决了发音长短不一的模板匹配问题。MFCC 特征值是从基于时间序列的音频信号中提取的特征，也是与时间相关的数据。DTW 把短的序列线性放大到和长的序列一样的长度再比较，使用动态规划来实现该算法。



算法伪码如下所示：

```
DTW-DISTANCE (X, Y):
    let D be a new array of size n * m
    for i=1 to n
        D[i][0] = INFINITY
    for i=1 to m
        D[0][i] = INFNITY
    D[0][0]=0
    for i=1 to n
        for j=1 to m
            cost=|X[i]-Y[j]|
            D[i][j]=cost + min(D[i-1][j], D[i][j-1], D[i-1][j-1])//插入、删除和匹配
    return D[n][m]
```

但是这样的方法还是有效率问题，fastdtw 基于论文的研究成果，将复杂度控制到线性时间和空间复杂度，因此直接使用已有的方法。

论文：

Stan Salvador, and Philip Chan. "FastDTW: Toward accurate dynamic time warping in linear time and space." Intelligent Data Analysis 11.5 (2007): 561-580.

2. 降维和特征选择

由于原始数据的正确标注结果为字符串，所以需要将原始标注结果转换为数值型数据，由于本实验只针对科进行分类，因此定义：

```
FAMILY = ['Bufonidae','Dendrobatidae','Hylidae','Leptodactylidae']
```

使用 pandas 导入数据，并将对应的值进行修改：

```
data_frame['Family']
      = data_frame['Family'].map(lambda x: FAMILY.index(x))
```

由于原始数据维度过多，因此考虑使用 **PCA** 进行降维。PCA 通过计算数据矩阵的协方差矩阵，得到协方差矩阵的特征值和特征向量，选取特征值最大的 N 个特征所对应的特征向量组成的矩阵，可以将数据转到新的空间，实现数据特征降维。本实验实现了 PCA 算法，并封装为模块进行调用。

本算法的实现流程为：

（1） 减去平均值
（2） 计算协方差矩阵
（3） 计算协方差矩阵的特征值和特征向量
（4） 对特征值排序
（5） 选取前 N 个最大的特征向量
（6） 转换数据到前 N 个特征向量构建的空间

3. 算法 1：k-means + agnes

算法的思路是使用 k-means 划分多个类，再使用 agnes 层次聚类算法聚类到 FAMILY 指定的 4 类。

k-means 是一种划分聚类方法，使用最简单的 k-means 算法：

（1） 随机选择 K 个随机的点
（2） 对与数据集中的每个数据点，按照距离 K 个中心点的距离，将其与距离最近的中心点关联起来，与同一中心点关联的所有点聚成一类；
（3） 计算每一组的均值，将该组所关联的中心点移动到平均值的位置
（4） 重复执行（2）-（3）步，直至中心点不再变化

在 k-means 执行过程中，使用**余弦相似度**来衡量 MFCC 特征值之间的距离。

agnes 是一种层次聚类方法，agnes 使用自底向上的策略进行簇之间的聚类。在前一步中，数据集已经按照 k-means 算法划分为多个簇，使用 agnes 计算簇之间的距离，由下而上合并距离最小的两个簇，直到最终只剩下 4 个分类。

agnes 的流程为：

（1） 将 k-means 聚类结果转化为初始簇
（2） 计算每两个簇之间的距离，找到距离最近的两个簇
（3） 合并两个簇
（4） 如果簇的数量大于 4，重复（2）-（4），否则算法结束

在这一步的聚类过程中，使用 **DTW** 衡量特征值之间的距离。

4. 算法 2： dbscan + agnes

算法的思路是使用 dbscan 划分多个类，再使用 agnes 层次聚类算法聚类到 FAMILY 指定的 4 类。

dbscan 是一种基于密度的聚类算法，在具有噪声的空间数据库中发现任意形状的簇，它将簇定义为密度相连的点的最大集合。

算法的流程为：

（1） 随机选取一个核心点

（2） 对该核心点进行扩充，遍历该核心点邻域内的所有核心点，寻找与这些数据点密度相连的点，直到没有可以扩充的数据为止

（3） 将这些点标记为一类

（4） 如果仍有核心点，重复以上过程

在算法进行过程中，第（2）步为遍历该核心点邻域内的所有核心点，为了能提高找到邻域内核心点的速度，引入 **KD-Tree** 数据结构。

Kd-Tree，即 K-dimensional tree，是一种高维索引树形数据结构，经常使用于在大规模的高维数据空间进行近期邻查找(Nearest Neighbor)和近似近期邻查找(Approximate Nearest Neighbor)，比如图像检索和识别中的高维图像特征向量的 K 近邻查找与匹配。

三、实验结果

1. 算法 1 参数对结果的影响

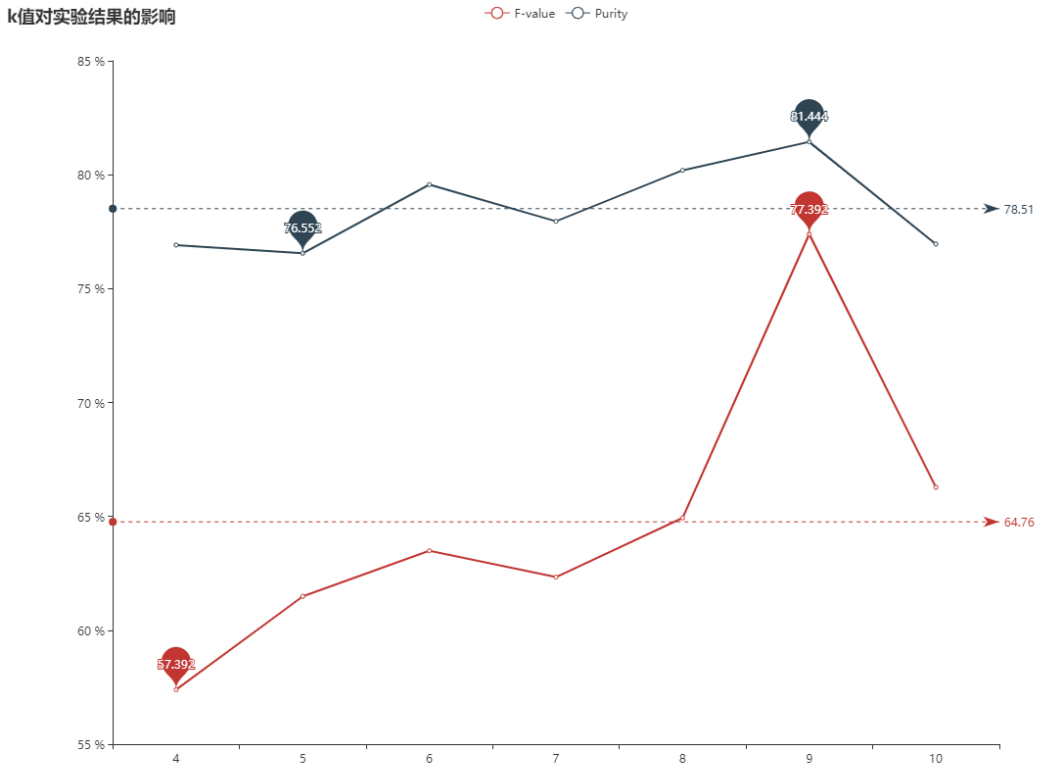
1.1 k 值对实验结果的影响

在算法 1 中，第一步是使用 k-means 划分为多个簇，第 2 步是对划分的簇进行层次聚类，第一步中 k 值对最后总结果的影响如下（每组取 5 次实验最高值）：

k	F-value	Purity
4	57.392%	76.912%
5	61.491%	76.552%
6	63.489%	79.566%
7	62.337%	77.957%
8	64.931%	80.193%
9	77.372%	81.444%
10	66.278%	76.955%

尽管 k-means 结果对随机生成的中心点较为敏感，但多次试验后仍然可以看出趋势，那就是随着 k 值的上升，最终结果越来越准确，这是因为前期划分较为精细，便于后期层次聚类时再次进行聚类。但是 k 值达到 10 后效果却出现了下滑，多次实验仍然保持这一趋势，这是因为 k-means **循环次数设置为 10 次，**k 值上升后需要的循环次数需要更多次才能收敛，

所有效果出现了下滑。

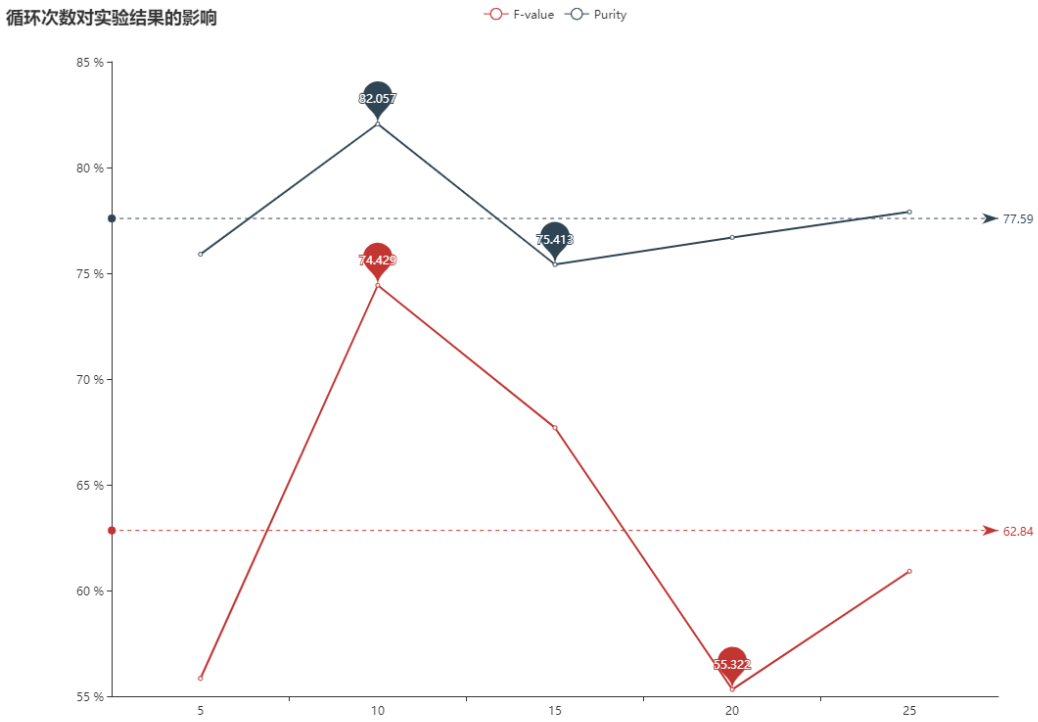


1.1 k-means 循环次数对实验结果的影响

为了提高性能，在进行 k-means 操作时并没有等到完全收敛才停止聚类，而实验效果表示等到完全收敛对整个算法的结果影响反而有可能更差，取 **k 值为 9**，循环次数与最终实验结果的统计如下（取 5 次实验的最高结果）：

N	F-value	Purity
5	55.841%	75.899%
10	74.429%	82.057%
15	67.697%	75.413%
20	55.322%	76.964%
25	60.911%	77.900%

理论上来讲，循环次数越多，k-means 分类越来越接近收敛，因此划分效果也会越好，但是实验结果证明需要的循环次数远小于收敛次数，只需要进行 10 次循坏进行简单划分就可以了，循环次数越多，划分结果越收敛不一定能引起最终结果的提升，反而增加了算法运行时间，从算法运行效率的角度来讲，k-means 循环只需要进行 10 次就足够了。



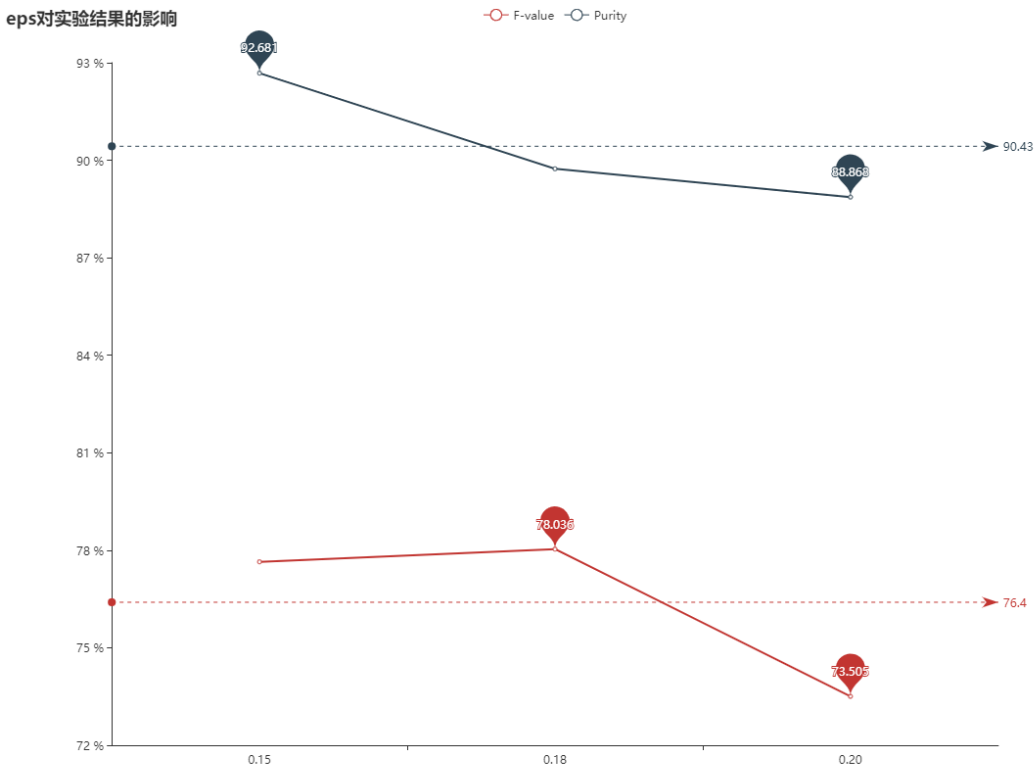
2. 算法 2 参数对结果的影响

由于 dbscan 的特殊性，有部分点被标记为异常点，因此在该部分的数据中**不包含识别出的异常点**，所有统计都是基于 dbscan 识别出的**非异常点**来进行的。算法需要调整的参数主要包括 eps 和 min_pts。

2.1 eps 对实验结果的影响

给定 min_pts=10，讨论 eps 的值与最终结果的关系。这里的 eps 指的是 KD-Tree 的参数 eps，而不是实际的半径距离衡量。

EPS	F-value	Purity
0.15	77.645%	92.681%
0.18	78.036%	89.736%
0.20	73.505%	88.868%



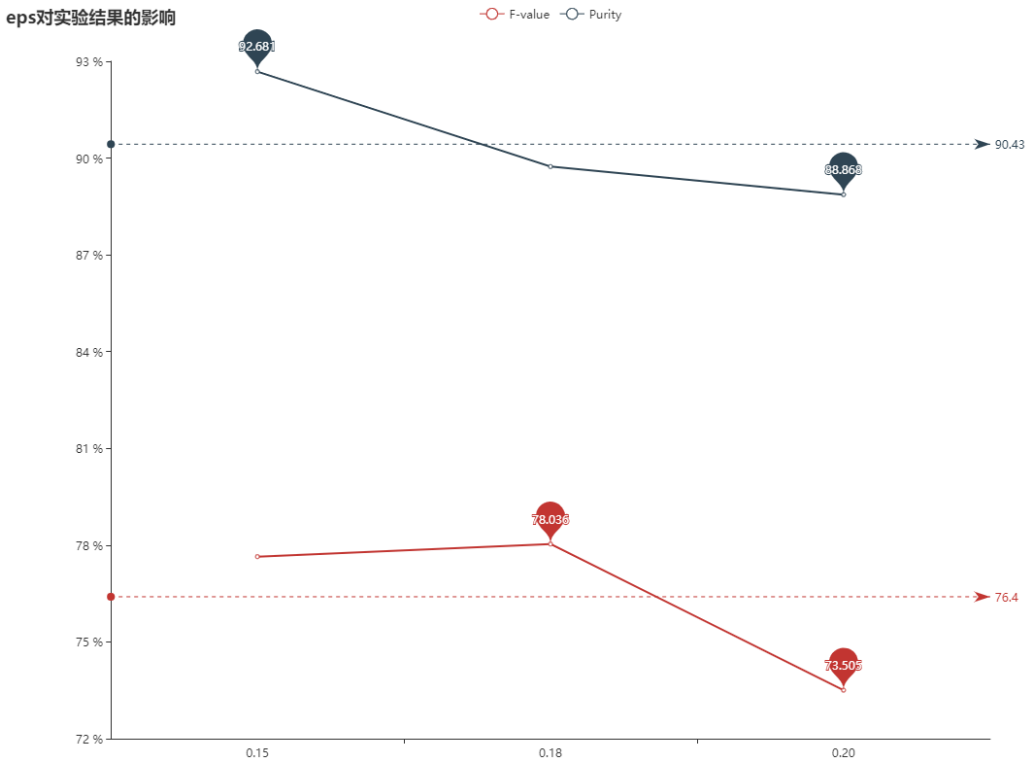
考虑到 **eps 过小会有太多的异常点**，综合考虑下确定 eps 设置为 0.18。

2.2 min_pts 对实验结果的影响

给定 eps=0.18， 讨论 min_pts 对实验结果的影响：

min_pts	F-value	Purity
8	76.755%	89.552%
10	79.221%	90.163%
12	85.713%	90.019%

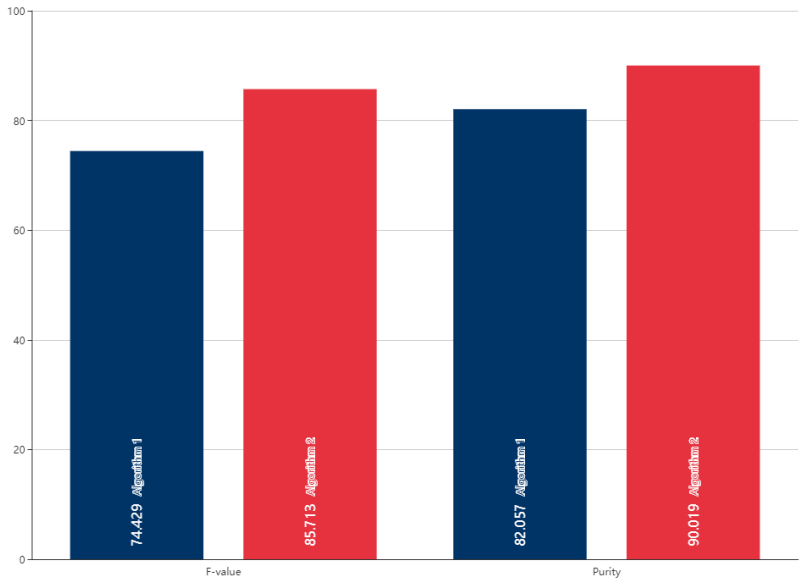
由实验结果可知，随着 min_pts 增加，F 值缓慢增加，而 Purity 效果并不明显。由于 min_pts 增加意味着固定半径内需要的点数更多，也就意味着异常点的数量会更多。为了控制异常点的数量，选取 min_pts 为 12。



3. 两种算法的对比

3.1 两种算法聚类效果的对比

算法 1 选取的参数：k=9，　循环次数=10
算法 2 选取的参数：eps=0.18，　min_pts=12



算法	K-value	Purity
算法 1：k-means + agnes	74.429%	82.057%
算法 2：dbscan + agnes	85.713%	90.019%

3.2 两种算法时间的对比

算法	所用时间
算法 1：k-means + agnes	88.472742456s
算法 2：dbscan + agnes	700.721409846s

3.3 两种算法的总结

在聚类效果上，算法 2 在数据上要优于算法 1，但是算法 2 排除了大量异常点，这些异常点在实际中不一定是异常点。此外，算法 2 需要的时间远远超过算法 1，在运行时间上还有不少问题。算法 1 虽然在聚类效果数据上略低于算法 2，但是有比较好的时间效率，在每个类的表现上也较为出色。