1. 主要问题描述

古语“物以类聚”，生活中有很多需要聚类的地方，比如客户群的划分，可能需要对客户的历史交易记录进行查阅，分析，股票板块分析，需要对股票的历史价格和涨跌幅进行分析，公交站点的安放，需要对住宅区进行聚类，以安放在使用频率会最高的地方。生物领域中，还会遇到将动植物分类，对基因，蛋白质分类等问题。这些问题的核心其实都在聚类。聚类（Clustering）就是将数据分组成为多个类（Cluster）。在同一类内对象之间具有较高的相似度，在不同的类之间的对象差别较大。为了解决这个问题，很多学者试图利用计算机算法来解决，并提出了很多算法，本报告主要针对 Kmeans 和 DBSCAN 算法进行介绍。

2. 方法描述

对于一个数据，人们既可以对变量（指标）进行分类（相当于对数据中的列分类），也可以对观测值（事件、样品等）来分类（相当于对数据中的行分类）。前者为 R 型聚类，后者为 Q 型聚类。聚类是一种无监督学习。作为数据挖掘中的无监督学习，其可以作为其他数学算法的预处理步骤，获得数据分布状况，集中对特定的类做进一步的研究。聚类所说的类不是事先划定的，而是依据数据的相似性和距离来划分。聚类的数目和结构都没有事先假定（有时候数目需要手动指定）。聚类分析的目的是寻找数据中潜在的“自然分组结构”（a structure of natural grouping）和感兴趣的“关系”（relation）

统计学角度的相似性（similarity）度量：

R 型聚类：相似系数

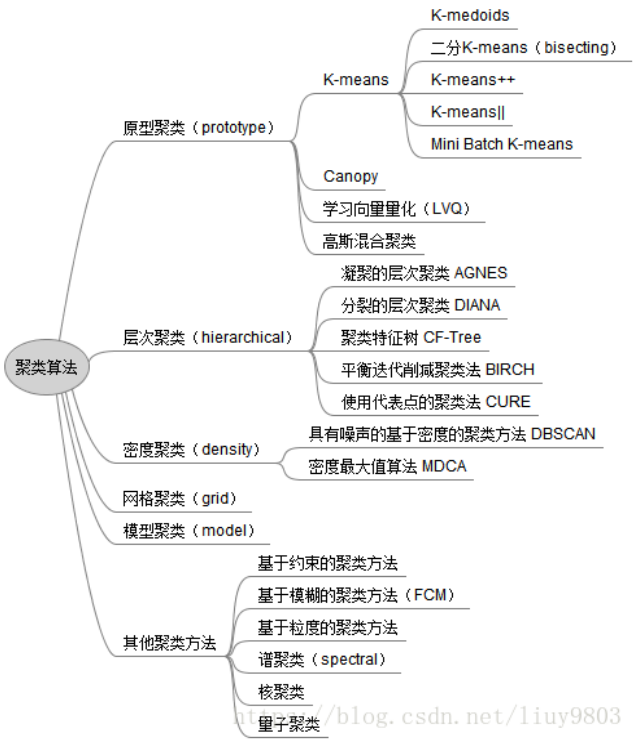
用于对变量（或指标）分类，可以用变量之间的相似系数的变形如 1-rij 定义距离。Pearson 相关系数：两个连续变量间呈线性相关。Spearman 相关系数：利用两变量的秩次大小作线性相关。Kendall 等级相关系数：利用两个变量的一致性作线性相关。Cosine 相关系数:利用两个变量的向量夹角余弦作线性相关

Q 型聚类：距离

用于对样品（或事件）分类。欧式距离：两点之间的直线距离。切比雪夫距离：两个向量之间的最大距离。曼哈顿距离:两个向量的绝对距离。马氏距离：数

据的协方差距离

聚类算法的分类如下：



划分聚类算法：任务是将数据划分成 K 个不相交的点集，使得每个子集中的点尽可能同质

基本思想：随机选择 K 个对象，每个对象初始地代表一个类的平均值或中心，对剩余每个对象，根据其到类中心的距离，被划分到最近的类；然后重新计算每个类的平均值。不断重复这个过程，直到所有的样本都不能再分配为止。

常用聚类算法：K-Means，K-Medoids

1.K-Means:硬聚类；计算每个类的中心值（平均值）

2.K-Medoids：K-Means 算法的改进；计算过程类似于 K-Means

特点：k 事先确定，创建初始划分再迭代，不必确定距离矩阵，计算量较小适合大型聚类，适用于发现球状聚类。

缺点：初值对结果影响很大，有些算法结果与输入顺序有关，用爬山算法寻找最优解容易陷入局部最优。

层次聚类算法（系统聚类算法）：对给定的数据进行层次分解，分为凝聚和分类

两类

基本思想：

1.凝聚的方法:自底向上，一开始将每个对象作为单独的一组，然后根据同类相近，异类相异的原则，合并对象，直到素有的组合并成一个，或达到一个终止条件。

2.分裂的方法：自顶向下，一开始将所有对象置于一类，在迭代的每一步中，一个类不断地分为更小的类，直到每个对象在单独的一个类中，或达到一个终止条件。

常用聚类算法：BIRCH，ROCK，Chameleon

1.BIRCH:利用层次方法的平衡迭代规约和聚类

2.ROCK：分类属性的层次聚类算法

3.Chameleon：利用动态建模的层次聚类算法

密度聚类算法：

基本思想：只要临近区域的密度超过一定的阈值，就继续聚类

常用聚类算法：DBSCAN，OPTICS，DENCLUE

1.DBSCAN：一种基于高密度连通的聚类

2.OPTICS：通过点排序识别聚类结构

3.DENCLUE：通过密度分布函数的聚类

特点：可以过滤噪声点和孤立点（outlier），发现任何形状的类

网格聚类算法：

基本思想：把样本空间量化为有限数目的单元，形成一个网络结构，聚类操作都在这个网格结构（即量化空间）上进行。

特点：处理速度很快，其处理时间通常独立于数据对象的数目，仅依赖于量化空间中每一维的单元数目。

常用聚类算法：STING，WaveCluster

1.STING：利用网格单元保存数据统计信息，从而实现多分辨率的聚类

2.WaveCluster：利用小波变换的聚类

模型聚类算法：

基本思想：为每个类假定一个模型，寻找数据对应模型的最佳拟合。

常用聚类算法：EM，COBWEB，SOM

1.EM：每个对象按照权重指派到每个簇，其中权重表示对象的隶属概率，优化隶属概率

2.COBWEB：一个通用的概念聚类方法，它用分类树的形式表现层次聚类

3.SOM：由外界输入不同的样本到人工的自组织映射网络中，一开始时，输入样本引起输出兴奋细胞的位置各不相同，但自组织后会形成一些细胞群，它们分别代表了输入样本，反映了输入样本的特征

几种常用的聚类算法从可伸缩性、适合的数据类型、高维性、异常数据的抗干扰度、聚类形状和算法效率等 6 个方面进行综合性能评价。

算法名称	可伸缩性	适合的数据类型	高维性	异常数据的抗干扰性	聚类形状	算法效率
WaveCluster	很高	数值型	很高	较高	任意形状	很高
ROCK	很高	混合型	很高	很高	任意形状	一般
BIRCH	较高	数值型	较低	较低	球形	很高
CURE	较高	数值型	一般	很高	任意形状	较高
K-Prototypes	一般	混合型	较低	较低	任意形状	一般
DENCLUE	较低	数值型	较高	一般	任意形状	较高
OptiGrid	一般	数值型	较高	一般	任意形状	一般
CLIQUE	较高	数值型	较高	较高	任意形状	较低
DBSCAN	一般	数值型	较低	较高	任意形状	一般
CLARANS	较低	数值型	较低	较高	球形	较低

3. 算法流程

3. 1. K-Means 算法：

输入-K：簇的数目，D：包含 N 个对象的数据集

输出-K 个簇的集合

方法：

1.从 D 中任意选择 K 个对象作为初始质心；

2.Repeat

3.根据簇中对象的均值，讲每个对象指派到最近的簇；

4.更新簇均值，即计算每个簇中对象的均值；

5.Until 不在发生变化

算法伪代码：

$$
\begin{aligned}
&\textit{repeat until clusters do not change}\\
&\quad \textit{for } j=1..\#clusters\\
&\qquad c_j = \{x_i : \textit{for all } f_l \textit{ and } d(x_i, f_j) <= d(x_i, f_l)\ \}\\
&\quad \textit{for } j=1..\#clusters\\
&\qquad f_j = mean(c_j)
\end{aligned}
$$

K-Means 算法的特点：

算法复杂度：O（K*N*D*T)

优点：实现简单、广泛应用于实际问题

缺点：

1.K 值即输入参数选择对结果影响大且需要人工给定

2.对噪声和离群点敏感

3.很难处理非球状簇

3. 2. DBSCAN 算法：

概念

1.核心点：给定对象的半径 Eps 领域内样本数量超过阈值 MinPts

2.边界点：落在核心点的领域内，在半径 Eps 内点的数量下小于 MinPts

3.噪声点：既不是核心点也不是边界点

4.直接密度可达：如果对象 q 在核心对象 p 的 Eps 邻域内，则称 q 从 p 出发是直接密度可达的。

5.密度可达：集合中的对象链 p1、p2、p3、...、pn，如果每个对象 pi+1 从 pi 出发都是直接密度可达的，则称 pn 从 p1 出发是密度可达的。

6.密度相连：集合中如果存在对象 o 使得对象 p 和 q 从 o 出发都是密度可达的，

则称对象 p 和 q 是互相密度相连的。

输入-D：包含 N 个对象的数据集，Eps：扫描半径，MinPts：密度阈值

输出-簇的集合，满足密度要求

方法：

1.Repeat；

2.从数据集中抽出一个未处理的对象；

3.IF 抽出的对象是核心点（核心对象），THEN 找出所有从该点密度可达的对象，形成一个簇；

4.ELSE 抽出的点是边缘点（非核心对象），跳出本次循环，寻找下一个对象；

5.Until 所有的对象都被处理。

伪代码如下：

```
输入：数据对象集合D，半径Eps，密度阈值MinPts
输出：聚类C
DBSCAN（D, Eps, MinPts）
Begin
init C=0; //初始化簇的个数为0
for each unvisited point p in D
mark p as visited; //将p标记为已访问
N = getNeighbours (p, Eps);
if sizeOf(N) < MinPts then
mark p as Noise; //如果满足sizeOf(N) < MinPts，则将p标记为噪声
else
C= next cluster; //建立新簇C
ExpandCluster (p, N, C, Eps, MinPts);
end if
end for
End
```

其中ExpandCluster算法伪码如下：

```
ExpandCluster(p, N, C, Eps, MinPts)
add p to cluster C; //首先将核心点加入C
for each unvisited point p' in N
mark p' as visited;
N' = getNeighbours (p', Eps); //对N邻域内的所有点在进行半径检查
if sizeOf(N') >= MinPts then
N = N+N'; //如果大于MinPts，就扩展N的数目
end if
if p' is not member of any cluster
add p' to cluster C; //将p' 加入簇C
end if
end for
End ExpandCluster
```

3. 3. 两种算法对比：

与 K-Means 相比，DBSCAN 不需要事先知道要形成的簇类的数量

与 K-Means 相比，DBSCAN 可以发现任意形状的簇类

同时，DBSCAN 能够识别出噪声点

DBSCAN 对于数据样本的顺序不敏感，即 Pattern 的输入顺序对结果的影响不大。但是，对于处于簇类之间边界样本，可能会根据哪个簇类优先被探测到而其归属有所摆动

4. 核心算法代码

4. 1. Kmeans 算法代码

```
private int findNearestStock(double[][] center, double[] point){
    double m = -1000000;
    int id = -1;
    for (int j = 0; j < K; ++j) {
        double dis = MyMath.coefOfAssociation(center[j], point);
        if (dis > m) {
            m = dis;
            id = j;
        }
    }
    return id;
}

public void clusterStock(ArrayList<String> names) {
    double[][] center = new double[K][dim];
    Random random = new Random();
    for (int i = 0; i < K; ++i) {
        double[] tmp = new double[dim];
        for (int j = 0; j < dim; ++j)
            tmp[j] = random.nextDouble();
        center[i]=tmp;
    }

    boolean done = false;
    int clk=0;
    while (!done && clk<200) {
        System.out.printf("第%d 次迭代\n",++clk);
        ArrayList<ArrayList<double[]>> clusters  = new ArrayList<>(
);//可视化部分
```

```
        for(int i=0;i<center.length;++i) clusters.add(new ArrayList
<>());

        double[][] nextCenter = new double[K][dim];//记录和，计算下次
的中心点
        int[] count = new int[K];//记录属于每个中心点的点数量，方便求平均
        for (double[] point : data) {
            int id=findNearestStock(center,point);
            for (int j = 0; j < dim; ++j)
                nextCenter[id][j] += point[j];
            ++count[id];
            clusters.get(id).add(point);
        }

        for (int i = 0; i < K; ++i) {
            for (int j = 0; j < dim; ++j)
                //count 为 0 的话，直接除会出错
                if(count[i]!=0)
                    nextCenter[i][j] /= count[i];
                else
                    nextCenter[i][j] = random.nextDouble();
        }

        double MDelta=-1;//记录最小距离的平方
        for(int i=0;i<K;++i){
            double dis=MyMath.getDistance(nextCenter[i],center[i]);
            MDelta=Math.max(MDelta,dis);
            System.out.printf("%d 号点移动%.2f 距离\n",i,dis);
            center[i]=nextCenter[i];
        }

        //终止条件
        if(Math.sqrt(MDelta)<1e-9){done = true;}
    }

    for(int i=0;i<K;++i){
        System.out.printf("第%d 个板块:",i);
        for(int j=0;j<data.size();++j){
            if(findNearestStock(center,data.get(j))==i){
                System.out.printf("%s ",names.get(j));
            }
        }
        System.out.println();
    }
```

```
}
```

4. 2. DBSCAN 算法代码

```
//并查集合并
private void Merge(int x, int y,int[] p){
    x=findSet(x,p);
    y=findSet(y,p);
    if(x == y) return ;
    data.get(x).neighbor.addAll(data.get(y).neighbor);
    data.get(y).neighbor.clear();
    p[y]=x;
}

public void clusterStock(ArrayList<String> names){
    for(int i=0;i<data.size();++i){//枚举每个点
        Dpoint dp1 = data.get(i);
        dp1.neighbor.add(dp1.v);//自己放进自己的 neighbor
        for(int j=0;j<data.size();++j) if(i!=j){//枚举其他的点，检查是
否在范围内
            Dpoint dp2 = data.get(j);
            if(MyMath.coefOfAssociation(dp1.v,dp2.v) >= e){//符合相
关性要求
                dp1.neighbor.add(dp2.v);//加入集合
            }
        }
    }

    int[] p = new int[data.size()];
    for(int i=0;i<data.size();++i) p[i]=i;//初始化并查集

    for(int i=0;i<data.size();++i) if(data.get(i).isKey()){
        for(int j=0;j<data.size();++j)
            if(data.get(j).isKey() && MyMath.coefOfAssociation(data
.get(i).v,data.get(j).v)>=e){
                Merge(i,j,p);//合并互相在 e 范围内的关键点
            }
    }

    int cnt=0;
    Set<String> vis = new HashSet<>();
    for(int i=0;i<data.size();++i) if(p[i]==i){
```

```
        ArrayList<String> tmp = new ArrayList<>();
        for(double[] t : data.get(i).neighbor){
            int id=-1;
            for(int j=0;j<data.size();++j) if(Arrays.equals(data.ge
t(j).v, t)){
                id=j;break;
            }
            String name = names.get(id);
            if(vis.contains(name)) continue;
            vis.add(name);
            tmp.add(name);
        }
        if(tmp.size()==0) continue;
        System.out.printf("第%d 个集合:",++cnt);
        tmp.sort(String::compareTo);
        System.out.println(tmp);
    }
}
```

5. 实验结果截图及分析

5.1. 结果评估方法

使用纯度(purity)评估方法。对于一个聚类 i，首先计算P_{ij}。指的是聚类 i 中的成员（member）属于类（class）j 的概率，$P_{ij} = \frac{m_{ij}}{m_i}$。其中$m_i$是在聚类 i 中所有成员的个数，$m_{ij}$是聚类 i 中的成员属于类 j 的个数。我们将聚类 i 的 purity 定义为$P = \max(P_{ij})$。整个聚类划分的 purity 为$purit = \Sigma_{i=1}^{K}\left(\frac{m_i}{m}\right)P_i$ ，其中 K 是聚类（cluster）的数目，m 是整个聚类划分所涉及到的成员个数。也就是说选择能使当前聚类结果正确最多的类别当作当前类别。

5.2. Kmeans 的聚类结果

参数 K=9，聚类结果如下，夹在同一对方括号里的股票为一类

[BCC,LOW]

[LSI,MU,TXN]

[AHC,SOTR]

[AA,ABK,ABS,ABT,ADM,AEP,AET,AFL,AGN,AIG,ALL,AMGN,AMR,AOC,APA,APC,APD,APOL,ASH,ASO,AT,AVP,AVY,AXP,AZO,BA,BAC,BAX,BC,BCR,BDK,BDX,BEN,BHI,BJS,BK,BLL,BLS,BMC,BMY,BNI,BOL,BR,BSC,BSX,BUD,CA,CAG,CAT,CB,CBE,CC,CCE,CCL,CCU,CI,CIN,CL,CLX,CMA,CMS,COF,COL,CPB,CSC,CSX,CTAS,CTB,CTL,CTX,CUM,D,DAL,DCN,DD,DDS,DE,DG,DIS,DJ,DLX,DNY,DOV,DOW,DRI,DTE,DUK,EC,ED,EFX,EK,EMN,EMR,EOG,EQR,ETN,ETR,F,FCX,FD,FDC,FDO,FDX,FISV,FITB,FLR,FNM,FON,FPL,FRE,FRX,G,GAS,GCI,GD,GDT,GDW,GE,GIS,GLK,GLW,GM,GP,GPC,GPS,GR,GT,GWW,HAL,HAS,HBAN,HCR,HD,HDI,HI,HIG,HLT,HMA,HNZ,HON,HPC,HRB,HRC,HSY,HUM,IBM,IFF,IGT,IP,IPG,IR,ITW,JCI,JCP,JNJ,JNY,JP,JPM,K,KEY,KMB,KMG,KO,KR,KRB,KRI,LEG,LEH,LIZ,LLY,LMT,LNC,LPX,LTD,LTR,LUV,LXK,MAR,MAS,MAT,MBI,MCD,MDT,MEL,MER,MHP,MIL,MIR,MMC,MMM,MO,MOLX,MOT,MRK,MRO,MSFT,MTG,MYG,NBR,NCC,NEM,NI,NKE,NOC,NOVL,NSC,NTRS,NUE,NWL,ODP,OMC,ONE,OXY,PBI,PCAR,PCG,PCL,PD,PEG,PEP,PFE,PG,PGR,PH,PLL,PNC,PNW,PPG,PPL,PX,QTRN,R,RATL,RBK,RDC,RIG,S,SAFC,SBC,SBUX,SCH,SFA,SGP,SHW,SIAL,SLB,SLE,SLM,SLR,SNA,SNV,SO,SPC,SPG,STI,STT,SUN,SVU,SWK,SWY,SYY,T,TE,TEK,THC,TIF,TIN,TJX,TMK,TMO,TNB,TOY,TRB,TRW,TXT,TXU,TYC,UCL,UNH,UNM,UNP,UPC,UST,UTX,VFC,WAG,WB,WEN,WFC,WHR,WIN,WMB,WMT,WWY,WY,X,XRX]

[ADCT,BGEN,BMET,GENZ,LLTC,MXIM,PMTC,PSFT,TLAB,UIS]

[BBY,CSCO,INTU,SPLS,SUNW]

[AAPL,ADI,AMD,ANDW,APCC,CMCSA,CMVT,CPWR,EMC,HET,INTC,KLAC,NAV,NSM,PAYX,QCOM,TER]

[ALTR,AMAT,BBBY,CTXS,ERTS,NVLS,ORCL,XLNX]

[ADBE]

评估后每个类别分别代表：

Services

Technology

Services

Services

Technology

Technology

Technology

Technology

Technology

总纯度为：0.26086≈26.1%

5. 3. DBSCAN 聚类结果

DBSCAN 聚类结果如下：参数 e=0.6，minp=3

[AA, AAPL, ABK, ABS, ABT, ADI, ADM, AEP, AET, AFL, AGN, AHC, AIG, ALL, AMGN, AMR, AOC, APA, APC, APD, ASH, ASO, AT, AVP, AVY, AXP, AZO, BA, BAC, BAX, BC, BCC, BCR, BDK, BDX, BEN, BHI, BJS, BK, BLL, BLS, BMC, BMY, BNI, BOL, BR, BSC, BSX, BUD, CA, CAG, CAT, CB, CBE, CC, CCE, CCL, CCU, CI, CIN, CL, CLX, CMA, CMCSA, CMS, COF, COL, CPB, CSC, CSCO, CSX, CTAS, CTB, CTL, CTX, CUM, D, DAL, DCN, DD, DDS, DE, DG, DIS, DJ, DLX, DNY, DOV, DOW, DTE, DUK, EC, ED, EFX, EK, EMC, EMN, EMR, EOG, EQR, ETN, ETR, F, FCX, FD, FDC, FDO, FDX, FITB, FLR, FNM, FON, FPL, FRE, FRX, G, GAS, GCI, GD, GDT, GDW, GE, GIS, GLK, GLW, GM, GP, GPC, GPS, GR, GT, GWW, HAL, HAS, HBAN, HCR, HD, HDI, HET, HI, HIG, HLT, HMA, HNZ, HON, HPC, HRB, HRC, HSY, HUM, IBM, IFF, IGT, INTC, IP, IPG, IR, ITW, JCI, JCP, JNJ, JNY, JP, JPM, K, KEY, KMB, KMG, KO, KR, KRB, KRI, LEG, LEH, LIZ, LLY, LMT, LNC, LOW, LPX, LSI, LTD, LTR, LUV, LXK, MAR, MAS, MAT, MBI, MCD, MDT, MEL, MER, MHP, MIL, MIR, MMC, MMM, MO, MOLX, MOT, MRK, MRO, MSFT, MTG, MU, MYG, NBR, NCC, NEM, NI, NKE, NOC, NSC, NTRS, NUE, NWL, ODP, OMC, ONE, ORCL, OXY, PAYX, PBI, PCAR, PCG, PCL, PD, PEG, PEP, PFE, PG, PGR, PH, PLL, PNC, PNW, PPG, PPL, PX, R, RBK, RIG, S, SAFC, SBC, SCH, SGP,

SHW, SIAL, SLB, SLE, SLM, SLR, SNA, SNV, SO, SOTR, SPC, SPG, STI, STT, SUN, SUNW, SVU, SWK, SWY, SYY, T, TE, TEK, THC, TIF, TIN, TJX, TMK, TMO, TNB, TOY, TRB, TRW, TXN, TXT, TXU, TYC, UCL, UNH, UNM, UNP, UPC, UST, UTX, VFC, WAG, WB, WEN, WFC, WHR, WIN, WMB, WMT, WWY, WY, X, XRX]

[ADBE]

[ADCT]

[ALTR]

[AMAT]

[AMD]

[ANDW]

[APCC]

[APOL]

[BBBY]

[BBY]

[BGEN]

[BMET]

[CMVT]

[CPWR]

[CTXS]

[DRI]

[ERTS]

[FISV]

[GENZ]

[INTU]

[KLAC]

[LLTC]

[MXIM]

[NAV]

[NOVL]

[NSM]

[NVLS]

[PMTC]

[PSFT]

[QCOM]

[QTRN]

[RATL]

[RDC]

[SBUX]

[SFA]

[SPLS]

[TER]

[TLAB]

[UIS]

[XLNX]

评估后每个类别代表实际类别：

Technology

Technology

Technology

Technology

Technology

Technology

None

Services

Services

Services

None

None

None

Technology

None

Services

Technology

Services

Healthcare

Technology

Technology

Technology

Technology

Consumer-Goods

Technology

Technology

Technology

Technology

None

Technology

None

None

Basic-Materials

Services

Financial

Services

Technology

Technology

Technology

Technology

最终的纯度结果：0.11594≈11.6%

5.4. 比较分析

在总体上来看，Kmeans 更胜一筹，但是也不是很理想，会出现某一个聚类包含

很多股票，而有几个聚类只有一支股票的情况。这种情况主要是由维数太高以及股票噪声较大引起的。而 DBSCAN 就更极端了，完全是一个聚类很多，剩下的全部都只有一个，试了很多参数，效果都不理想。

6. 总结

这个实验的难点可能主要在数据集，因为股票数据噪声大，不好处理。然后维数高了的话，算距离就容易都差不多，就聚在一块儿去了。也没有想到什么好的办法，也许可以先使用特征筛选，选一部分影响比较大的特征出来，或者也许可以用决策树，随机森林什么的试一试，也许会有不一样的结果。