期货交易分析及异常投资判别

班级:计算机卓越1601班　　姓名:徐永鑫　　学号:U201610002

摘要

期货（Futures）与现货完全不同，现货是实实在在可以交易的货（商品），期货主要不是货，而是以某种大众产品及金融资产为标的标准化可交易合约。我国的期货发展历史已有十多年，吸引了大量交易者的参与，但也有一些异常投资者，想通过自己对期货买卖的操作使得自己非正常获益。本文旨在对给出的交易数据进行分析，找到影响期货交易价格的成分与因子。并且通过聚类划分与数据清洗分析，找出异常投资者，分析他们的异常行为。

为了达到这样的目的，本文建立了一下几种数学模型对期货交易数据进行清洗、归类与分析。

1. 主成分分析模型，为了确定影响交易价格的因子，利用已有的数据，以天为单位的基础上，对成交数量、手续费、成交时间、成交日期进行主成分分析，将数据标准化，并获得相关系数表和特征方程，提取特征值大于 1 的前 4 个主成分，通过计算可得到每个主成分前的系数，即特征向量。计算可得出主成分表达式。最后可由主成分综合模型中根据每个因素的贡献率判定对价格的影响因素。

2. 基于密度聚类的交易特征刻画模型，本文尝试了(成交时间,成交价格)、(成交时间,成交数量)、(成交数量,成交价格)等多种组合，经过分析与筛选，最终选择了对于(成交数量，成交价格)的集合划分聚类，以刻画期货交易的特征。由于天数较多且许多天的交易情况较为类似，选择了具有代表性的三天进行了详细分析，试图完整的体现交易特征。

3. 基于密度聚类的异常投资判别模型，利用DBSCAN算法的特性，密度聚类在计算的过程中会自动清除一些“噪点”，达到数据清洗的效果，而这些“噪点”极有可能就是本文寻找的异常投资案例。

4. 基于特征时间分析与持仓量分析的异常投资判别模型，采用以下几种方案来确定异常投资现象：

a. 同一时间以相同价格买入卖出的投资者；

b. 每k笔交易中是否出现同价位大量买入卖出的投资者；

c. 关键时间(开盘、闭盘等)成交情况，对于大量买入、卖出者予以谨慎；

d. 通过投资者持仓量变化情况，分析异常投资者。

最后，本文对模型的建立以及实现进行了评估，对于优势和劣势进行了简练的分析，并对模型的优化提出了一些可行的建议。

关键词 主成分分析 基于密度的聚类 异常投资者

1. 题目分析

现有一个月的期货交易数据，本文将在题目提供的数据下根据数学模型的方法完成以下几项任务：

1. 分析月度的数据，从多种搭配组合尝试，找到最适合分析的指标，下文将通过尝试对(成交时间,成交价格)、(成交时间,成交数量)、(成交数量,成交价格)的二维分析，以及(成交数量,成交价格,成交时间)的三维分析，确定出需要分析的维度。

2. 选择五种给定的数据标签，建立五维矩阵，通过主成分分析，分析影响交易金额的因素。

3. 通过分析以日为单位的成交量和成交金额的关系，结合聚类分析方法得到数据清洗后的商品聚类，从而划分有共同特点的交易类型。

4. 通过聚类分析清洗出一些“噪点”，对于噪点的分析，推测是否是异常交易。并且通过对于关键时间点、极端交易数量等推测一些异常交易者。

2. 基本假设

1. 成交数量、手续费、成交时间、成交日期对成交价格均有一定影响。
2. 商品种类不收成交数量、手续费、成交时间、成交日期和成交价格指标的影响。换句话说，通过成交数量、手续费、成交时间、成交日期和成交价格指标不能划分出商品种类。
3. 异常交易者的异常交易操作是为了盈利或其他特殊目的的，不是无目的操作。
4. 开盘期，休盘期为关键成交时间，大量的交易为异常交易，自我买卖也为异常交易。
5. 在分析数据中，不存在交割结算

表1 基本假设表

3. 符号说明

表2 符号说明表

p	分析维度
x_i	第i个原始数据维随机向量
Z_{ij}	标准矩阵的第i行j列号元素
r_{ij}	本相关矩阵第i行j列号元素
R	本相关矩阵
λ_j	第j个特征向量
U_k	第k主成分
C_m	第m个聚类
ϵ	邻域大小(邻域参数1)
$N_\lambda(x_q)$	第q个数据在ϵ邻域中的点个数
$Minpts$	使得对象成为核心对象的最小邻域点数(邻域参数2)

4. 指标选择说明

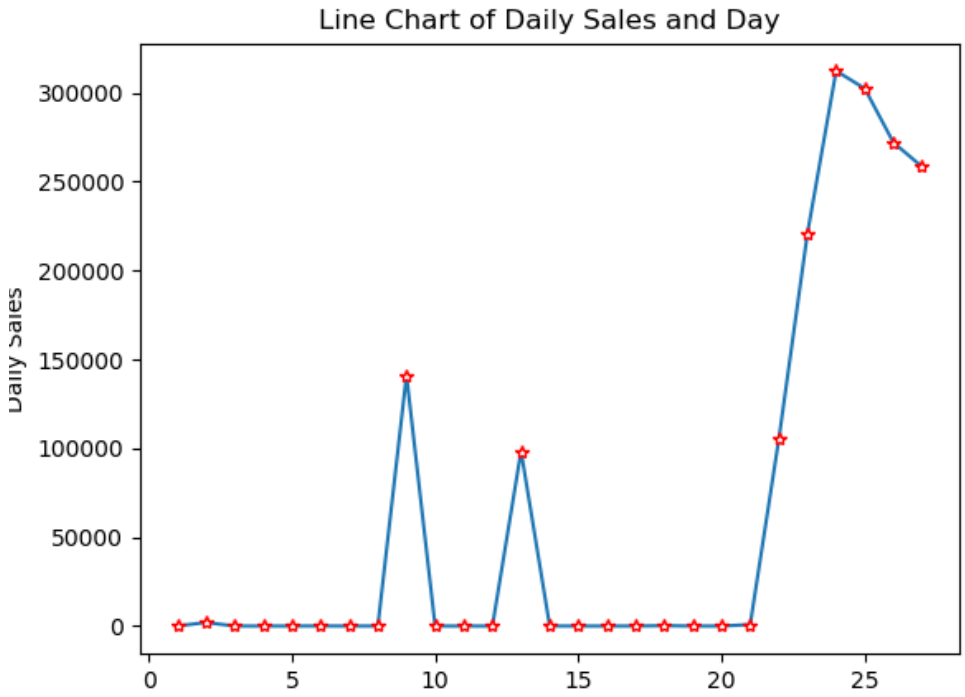

图1 每日销售量图标

由于以每次交易的时间(分钟)为单位，数据量过多过杂，难以直接看出规律，于是选择以天为单位，分析其余的指标。

以上是每日的销售量图标，分析数据可以发现，8月1日到8月31日中，共有4天休盘(分别是8月7日、8月14日、8月21日和8月28日)。交易的数量从开始的每天都很少，到中间有两个交易小高潮(8月10日和8月15日，以及到月末的交易成线性高速上升。

由此，可以分析3个日期结点，来概括整个月的期货交易特征，一种是很少交易的平缓期，一种是中间的两个小高潮日，还有一个是月末的至高点的分析。

为了弄清为什么造成了8月10日和8月15日的销量急剧增高，继续分析交易量和日期的关系，图形如下：

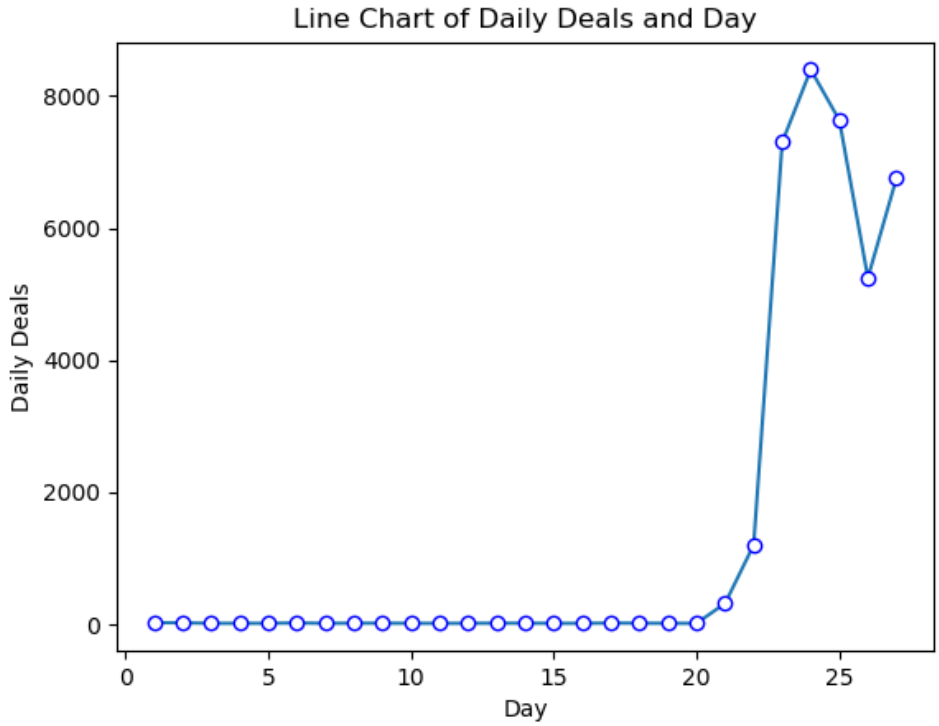

图2 每日交易单数

可以看出，在第0~21日(除了8月10日和8月15日)，每日交易单数和每日销售量图标基本吻合，之后的上升也基本吻合。所以，在8月10日和8月15日，可能有个人用户一单进行了大笔交易，在之后的分析中要加以关注。

下面尝试组合(成交时间,成交价格)、(成交时间,成交数量)、(成交数量,成交价格)对交易数据进行分析，由于成交时间与成交数量的图，图一已经分析，于是不再进行重复画图。由于它的异常明显的可以看出，除了月底的高密度交易外也很难体现交易倾向，所以不使用(成交时间,成交数量)做为交易行为分析的指标。而由于单日的成交价格无法确定(商品种类未知)，且不能草率地求平均数得到成交价格，所以(成交时间,成交数量)的二元组被舍弃。

所以选择(成交数量、成交价格)的二元组作为交易特征分析的指标。

5. 主成分分析模型

5.1 主成分分析模型原理

主成分分析(Principal Component Analysis)也称主分量分析，旨在利用降维的思想，把多指标转化为少数几个综合指标[5]。本文利用主成分分析的目的是确定成交数量、手续费、成交时间(具体时间)、成交日期对于**成交价格**特征的影响，划分出主成分，并计算出每个主成分的方差贡献率 ，即权重

本文运用的主成分分析方法将按以下几个步骤执行：

1. 原始数据的标准化采集p维随机向量(这里p为影响成分。这里等于5，$x = (x_1, x_2, \ldots, x_5)^T$，以此为基础取n个样本$x_i = (x_{i1}, x_{i2}, \ldots, x_{i5})^T, i = 1,2,...,n$。其中$n > p$，构造样本本阵，对样本阵元进行如下标准化变换：

$$\text{标准矩阵元素} Z_{ij} = \frac{x_{ij} - \overline{x_j}}{s_j}, i = 1,2,...,n; j = 1,2,...,5,$$

$$\text{其中} \overline{x_j} = \frac{\sum_{i=1}^{n}(x_{ij})}{n},\ s_j{}^2 = \frac{\sum_{i=1}^{n}(x_{ij} - \overline{x_j}^2)}{n-1},\ \text{得到标准化矩阵Z}$$

2. 对标准化阵z求相关系数矩阵：

$$R = [r_{ij}]_p xp = \frac{Z^T Z}{n-1},$$

$$\text{其中，} r_{ij} = \frac{\sum z_{kj}{}^2}{n-1}, i, j = 1,2,...,5$$

3. 解样本相关矩阵 R 的特征方程 ,得到5个特征根，确定主成分

$$\text{按} \frac{\sum_{j=1}^{m}\lambda_j}{\sum_{j=1}^{p}\lambda_j} \geq 0.95$$

确定m的值，使得信息的利用率达到95%以上，对于每个$\lambda_j, j = 1,2,...,m$解方程组$Rb = \lambda_j$得到单位特征向量b_j^o。

4. 将标准化后的指标变量转换为主成分

$$\text{主成分} U_{ij} = z_i^T b_j^o, j = 1,2$$

其中U_1称为第一主成分，U_2称为第二主成分。对应较小的一些特征向量被舍弃了，这是降维导致的结果，但舍弃这部分信息是必要的：一方面，舍弃这部分信息之后能使样本的采样密度增大，这正是降维的动机；另一方面，当数据收到噪声影响时，最小的特征值所对应的特征向量往往与噪声有关，将他们舍去能在一定程度上起到降噪的效果。

5. 对 m 个主成分进行综合评价。对 m 个主成分进行加权求和，即得最终评价值，权数为每个主成分的方差贡献率。

5.2 主成分分析模型在期货交易分析中的实例化

下面是利用SPSS Statistical软件进行主成分分析得到的一些表格。

	交易日期	成交价格	成交数量	手续费	成交时间
交易日期	1	0.679	-0.052	-0.026	1
成交价格	0.679	1	-0.059	0.040	0.683
成交数量	-0.052	-0.059	1	0.205	-0.052
手续费	-0.026	0.040	0.205	1	-0.025
成交时间	1	0.683	-0.052	-0.025	1

表2. 实例化模型的相关矩阵R

	初始	提取
成交价格	1.000	0.941
交易日期	1.000	0.700
成交数量	1.000	0.594
手续费	1.000	0.613
成交时间	1.000	0.944

表3. 运用主成分分析方法获得的公因子方差表

	初始特征值			提取平方和载入		
成分	合计	方差的%	累积%	合计	方差的%	累积%
1	2.59	51.807	51.807	2.59	51.807	51.807
2	1.202	24.043	75.85	1.202	24.043	75.85
3	0.799	15.971	91.821			
4	0.409	8.172	99.993			
5	0	0.007	100			

表4. 解释的总方差

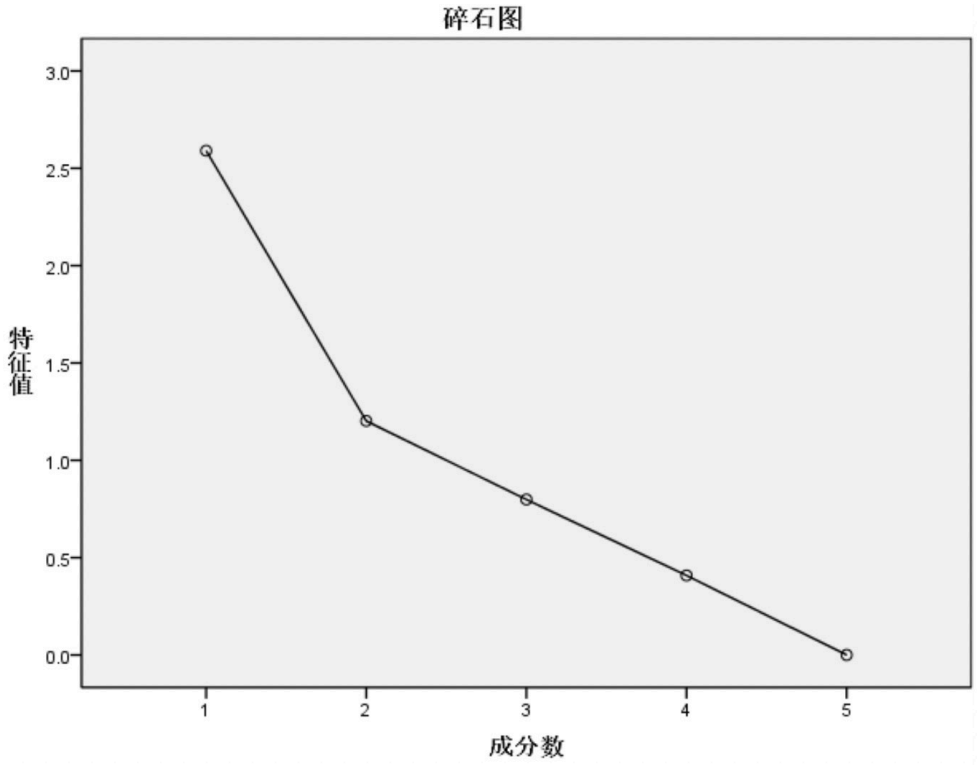

图3. 乘分数数和特征值折线图

	成分	
	1	2
交易日期	0.97	0.02
成交价格	0.834	0.065
成交数量	-0.098	0.764
手续费	-0.023	0.783
成交时间	0.971	0.02

表5 主成分分析放下下的成分矩阵

	成分	
	1	2
交易日期	0.375	0.016
成交价格	0.322	0.054
成交数量	-0.038	0.636
手续费	-0.009	0.651
成交时间	0.375	0.017

表6 构成得分

经过计算得到主成分综合模型：

$$F_1 = 0.375z_1x_1 + 0.322z_2x_2 - 0.038z_3x_3 - 0.009z_4x_4 + 0.375z_5x_5$$
$$F_2 = 0.016z_1x_1 + 0.054z_2x_2 - 0.636z_3x_3 - 0.651z_4x_4 + 0.017z_5x_5$$

6. 密度聚类的分析模型

6.1 密度聚类的模型原理

密度聚类又称基于密度的聚类(Density-based clustering)，此算法假设聚类结构能通过样本分布的紧密程度确定。

DBSCAN算法(Density-Based Spatial Clustering of Application with Noise)，它基于一组“邻域”参数$(\epsilon, Minpts)$来刻画样本分布的紧密程度。本文也将运用此特征，将抽取一些特征天数的的(成交单价, 成交数量)做密度聚类，以此来刻画交易的类型特征。

给定数据集$D = x_1, x_2, \ldots, x_m$，定义下面几个概念。

ϵ-邻域：对$x_j \in D$，其ϵ-邻域包含样本即D中与x_j的距离不大于ϵ的样本，即$N_\epsilon(x_j) = \{x_j \in D \mid dist(x_i, x_j) \leq \epsilon\}$。

核心对象:若x_j的ϵ-邻域至少包含$Minpts$个样本，即$N_\epsilon(x_j) \geq Minpts$，则 x_j 是一个核心对象

密度直达: 若 x_j 位于 x_i 的 ϵ - 邻域中，且x_i是一个核心对象，则成 x_j 到 x_i 密度直达。

密度可达: 对 x_i 与 x_j，若存在样本$p_1, p_2, \ldots p_n$，其中$p_1 = x_i$，$p_n = x_j$，且对每一个i，p_i 到 p_{i+1}密度直达，则称 x_i 与 x_j 密度可达。

密度相连: 对 x_i 与 x_j，若存在 x_k，使得 x_i 与 x_j 均有 x_k 密度可达，则称 x_i 与 x_j 密度相连。

下图给出了上述概念的直观显示[5]

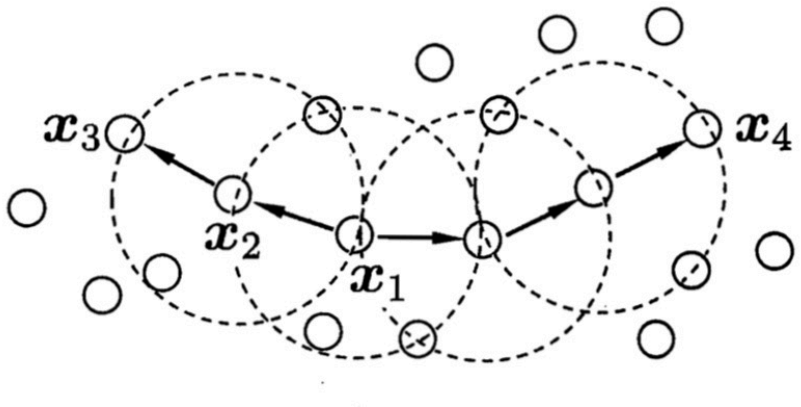

图4 密度聚类例图

上图DBSCAN定义的基本概念(图中$Minpts = 3$): 虚线显示出 ϵ - 邻域，x_1 是核心对象，x_2 由 x_1 密度直达，x_3 由 x_1 密度可达，x_3 与 x_4 密度相连。

基于以上的概念，DBSCAN将聚类定义为：由密度可达的关系导出最大的密度相连样本集合。形式化地说，给定邻域参数$(\epsilon, Minpts)$，聚类$C \in D$是满足以下性质的非空样本子集：

连接性: $x_i \in C, X_j \in C \Rightarrow x_i$ 与 x_j 密度相连;

最大性: $x_i \in C, x_i$ 由 x_j 密度可达 $\Rightarrow x_j \in C$。下面是DBSCAN算法实现的伪代码：

输入: 样本集D = $\{x_1, x_2, \ldots, x_m\}$;

邻域参数$(\epsilon, Minpts)$

算法过程:

1. 初始化核心对象集合: $\Omega \in \varnothing$
2. **for** $j = 1,2,...,m$ **do**
3. 确定样本x_j的邻域$N_\epsilon(x_j)$
4. **if** $N_\epsilon(x_j) \geq Minpts$ **then**

5. 将样本x_j的加入核心对象集合: $\Omega = \Omega \bigcup \{x_j\}$;

6. **endif**

7. **end for**

8. 初始化聚类个数: k = 0

9. 初始化未访问样本集合: $\Gamma \in D$

10. **while** $\Omega \neq \varnothing$ **do**

11. 记录当前未访问集合: $\Gamma_{old} = \Gamma$

12. 选取一个核心对象$o \in \Omega$，初始化队列$Q = < o >$

13. $\Gamma = \Gamma \ \{o\}$

14. **while** $Q \neq \varnothing$ **do**

15. 取出队列Q的首个样本q

16. **if** $N_\epsilon(q) \geq Minpts$ **then**

17. 令$\Delta = N_\epsilon \bigcap \Gamma$

18. 将Δ中的样本加入队列Q

19. $\Gamma = \Gamma \setminus \Delta$

20. **end if**

21. **end while**

22. $k = k + 1$，生成聚类$C_k = \Gamma_{old} \setminus \Gamma$

23. $\Omega = \Omega \setminus C_k$

24. **end while**

25. 输出聚类划分$C = C_1, C_2, \ldots, C_k$

图5 DBSCAN伪代码算法

DBSCAN算法先任选数据集中的一个核心对象为“种子”，再由此出发确定相关的聚类划分，算法如伪代码所示，在第1~7行中，算法先根据给定的邻域$(\epsilon, Minpts)$找出所有的核心对象；然后在10~24行中，以任一核心对象作为出发点，找出由其密度可达的样本生成聚类簇，直到所有核心对象被访问到为止。

不是核心对象的独立点将被作为噪点被删除，达到了数据清洗的效果。

6.2 利用密度聚类划分交易聚类

按照前面的分析，选择了具有代表性的三天(平稳状态下的第五天、突然增长的第九天和最高点第二十四天进行分析)

分析的指标是每单交易的成交价格与成交数量。

画出的散点图如下：

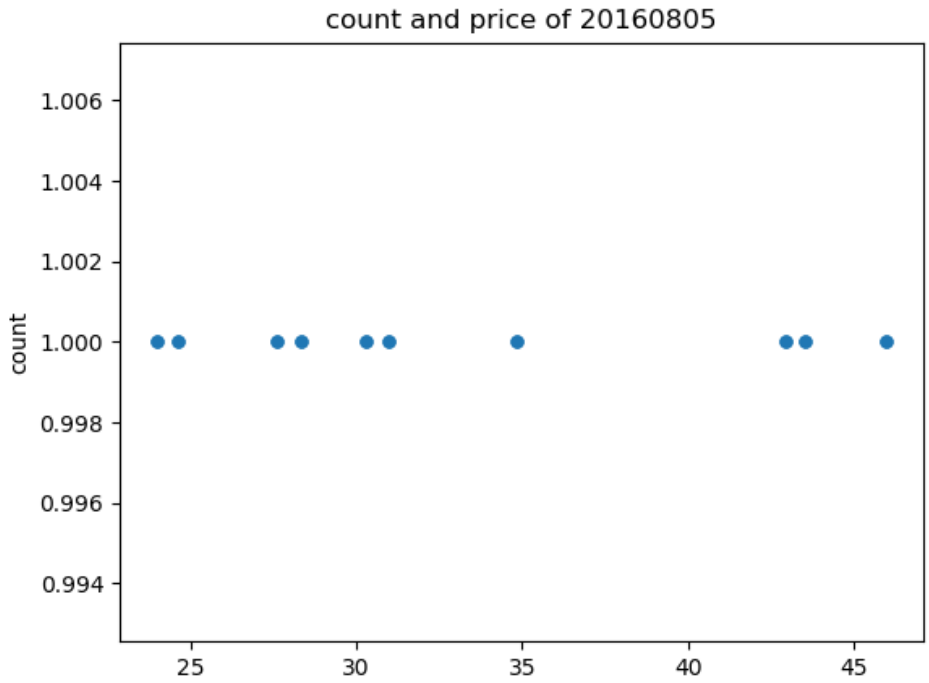

图6 第5天的交易数量和交易价格的关系

可以看出，第五天(2016年8月5日）的交易数量非常少，没有划分聚类的必要，每一笔的成交量只有一，可以看出，在月初，交易者普遍都保守交易，很少交易。

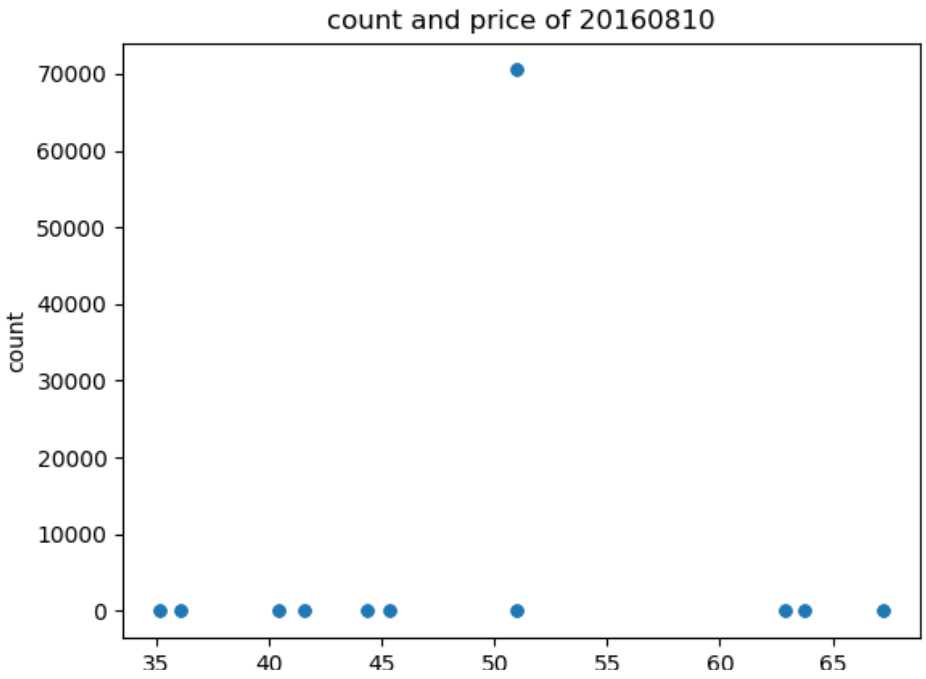

图7 第9天的交易数量和交易价格的关系

这一天（八月十日）有一单非常大的交易，一笔交易量达到了70515，其他的交易数仍然为1，查询这单的编号，这一单的详细交易如下：

交易日期	投资者编号	买/卖	成交价格	成交数量	成交金额	手续费	成交时间
20160810	566661638521	买	51.01	70515	3596970.15	0	8/10/16 15:00
20160810	566603263583	卖	51.01	70515	3596970.15	359.7	8/10/16 15:00

表7 第一单异常交易

由于比较直观，这一天仍然不需要划分聚类。

下面是最高峰的交易分布情况：

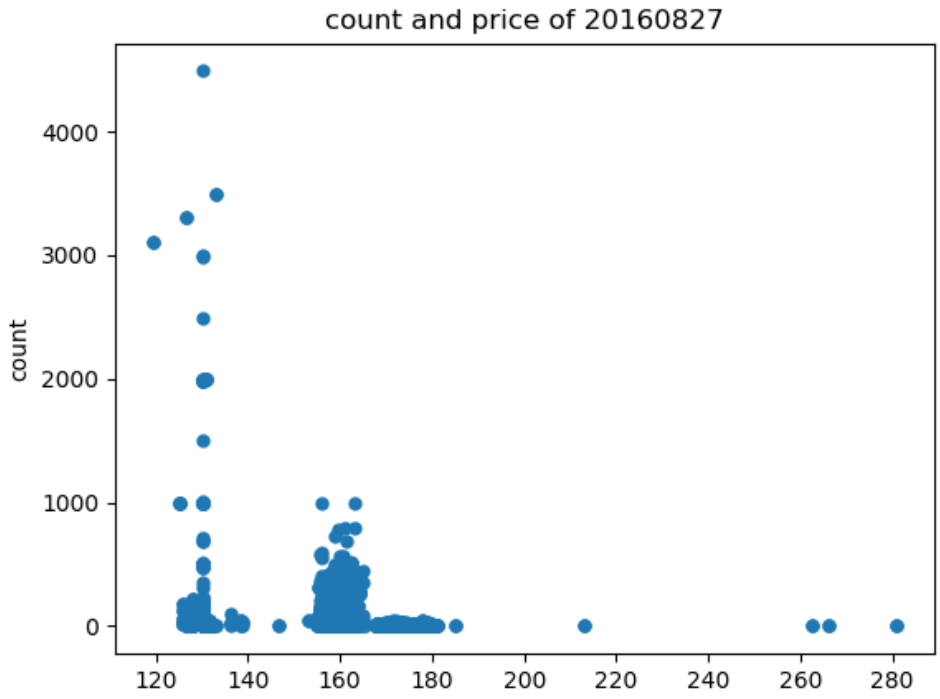

图8 第24天的交易数量和交易价格的关系

位于月末，交易数额非常之多，尽管在图上可以看出一些孤立的点，但难以用肉眼划分聚类，于是利用前文所述的DBSCAN算法进行聚类划分，划分的结果如下：

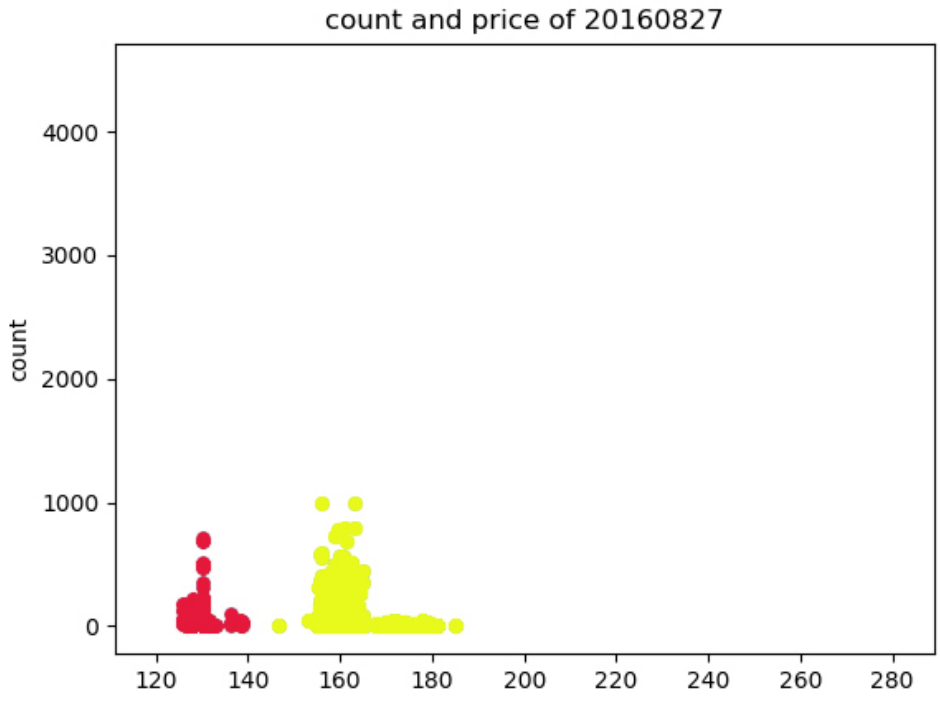

图9 第24天的交易数量和交易价格的关系聚类

选择的参数$(\epsilon, Minpts) = (60,5)$，清洗掉的一些点保留在了文件中，以便后续异常投资者判别。可以看出划分除了左下角（红色标记）的聚类一，和左下角偏右（黄色标记）的聚类二。通过人工筛选数据可以发现，这些购买都是来自于相似的投资者编号。

可以得出以下几个结论：

1. 有少量的大笔交易，大笔交易通常来自于固定的几个投资者。
2. 月初交易量较少，月末交易密度很大。
3. 人们的投资偏向于价格在120~200之间，销售数量在50~800之间。

7. 异常投资者判别模型

首先，通过之前的聚类分析，可以得到以下的异常投资列表：

交易日期	投资者编号	买/卖	成交价格	成交数量	成交金额	手续费	成交时间
20160810	5666616385	买	51.01	70515	3596970.15	0	8/10/16 15:00
20160810	5666032635	卖	51.01	70515	3596970.15	359.7	8/10/16 15:00
20160815	5666398577	买	74.68	49200	3674256	0	8/15/16 15:00
20160815	5666231378	卖	74.68	49200	3674256	367.43	8/15/16 15:00
20160825	5008580719	买	156.64	3044	476812.16	0	8/25/16 10:31

交易日期	投资者编号	买/卖	成交价格	成交数量	成交金额	手续费	成交时间
20160825	5008767098	卖	156.64	3044	476812.16	953.62	8/25/16 10:31
20160827	5189381680	买	133	3500	465500	0	8/27/16 9:30
20160827	5189381680	卖	133	3500	465500	46.55	8/27/16 9:30
20160827	5189381680	买	126.5	3300	417450	0	8/27/16 9:30
20160827	5189381680	卖	126.5	3300	417450	41.75	8/27/16 9:30
20160827	5189381680	买	119.25	3100	369675	0	8/27/16 9:30
20160827	5189381680	卖	119.25	3100	369675	36.97	8/27/16 9:30
20160827	5189381680	买	130	3000	390000	0	8/27/16 9:37
20160827	5189381680	买	130	4490	583700	0	8/27/16 9:39
20160827	5189381680	买	130	2490	323700	0	8/27/16 9:39
20160827	5189381680	买	130	2990	388700	0	8/27/16 9:40
20160830	5189381680	买	129.5	3200	414400	0	8/30/16 9:30
20160830	5189381680	买	124	3100	384400	0	8/30/16 9:30
20160830	5189381680	买	126	2750	346500	0	8/30/16 9:30
20160830	5189381680	买	125	2650	331250	0	8/30/16 9:30
20160830	5189381680	买	116.5	4850	565025	0	8/30/16 9:33
20160830	5189381680	买	115.8	5750	665850	0	8/30/16 9:34
20160830	5189381680	买	116.2	5350	621670	0	8/30/16 9:37
20160830	5189381680	买	116.3	3600	418680	0	8/30/16 9:38
20160830	5189381680	买	117.1	2850	333735	0	8/30/16 9:40
20160831	5189381680	买	118.5	7500	888750	0	8/31/16 9:30
20160831	5189381680	买	110	7350	808500	0	8/31/16 9:30
20160831	5189381680	买	108.65	7330	796404.5	0	8/31/16 9:30
20160831	5189381680	买	121	6980	844580	0	8/31/16 9:30
20160831	5189381680	买	118	6800	802400	0	8/31/16 9:30
20160831	5189381680	卖	108.65	5750	624737.5	62.47	8/31/16 9:30
20160831	5189381680	卖	110	4201	462110	46.21	8/31/16 9:30
20160831	5189381680	卖	118	3200	377600	37.76	8/31/16 9:30
20160831	5189381680	卖	110	3100	341000	34.1	8/31/16 9:30
20160831	5189381680	卖	118	2701	318718	31.87	8/31/16 9:30
20160831	5189381680	卖	121	2650	320650	32.07	8/31/16 9:30

表8 通过对每天的商品进行密度聚类得出的异常交易商品

然后，根据以下几个原则寻找一场投资者：

a. 同一时间以相同价格买入卖出的投资者；

由于进行搜索的时候发现，满足这一情况的投资数目有11214条，所以这里只打印出有异常交易的交易者编号。（且只打印异常交易次数超过十次的投资者）

表9 异常交易者编号以及异常行为次数a

序号	编号	异常交易次数
1	598830705605	16
2	500858071996	1171
3	500959305713	87
4	566661638521	75
5	518938168077	1808
6	588626595810	3818
7	555502312363	14
8	518909953722	187
9	555583599195	15
10	169060600057	73
11	500832653169	3144
12	588850713783	11
13	588681311961	680

b. 每k笔交易中是否出现同价位大量买入卖出的投资者；

这里取单笔交易超过500的输出。

表10 异常交易者编号以及异常行为次数b

1	500959305713	1
2	588626595810	20
3	518938168077	93
4	518909953722	8

c. 关键时间(开盘、闭盘等)成交情况，对于大量买入、卖出者予以谨慎；

由于开盘、闭盘之时，如果没有得到明确的信息，是无法判断期货会涨还是会跌的，所以在开盘和闭盘时期大量买入属于异常。

通过筛选得到以下的数据（买入、卖出量超过1000），由于数量过多，省略了63条信息。

交易日期	投资者编号	买/卖	成交价格	成交数量	成交金额	手续费	成交时间
20160810	5666616385	买	51.01	70515	3596970.15	0	8/10/16 15:00
20160810	5666032635	卖	51.01	70515	3596970.15	359.7	8/10/16 15:00
20160815	5666398577	买	74.68	49200	3674256	0	8/15/16 15:00
20160815	5666231378	卖	74.68	49200	3674256	367.43	8/15/16 15:00
20160831	5189381680	买	118.5	7500	888750	0	8/31/16 9:30
20160831	5189381680	买	110	7350	808500	0	8/31/16 9:30
20160831	5189381680	买	108.65	7330	796404.5	0	8/31/16 9:30
20160831	5189381680	买	121	7230	844580	0	8/31/16 9:30
20160831	5189381680	买	118	6800	802400	0	8/31/16 9:30
20160830	5189381680	买	115.8	5750	665850	0	8/30/16 9:34
20160831	5189381680	卖	108.65	5750	624737.5	62.47	8/31/16 9:30
20160830	5189381680	买	116.2	5350	621670	0	8/30/16 9:37
20160830	5189381680	买	116.5	4850	565025	0	8/30/16 9:33
20160827	5189381680	买	130	4490	583700	0	8/27/16 9:39
20160831	5189381680	卖	110	4201	462110	46.21	8/31/16 9:30
20160830	5189381680	买	116.3	3600	418680	0	8/30/16 9:38
20160827	5189381680	买	133	3500	465500	0	8/27/16 9:30
20160827	5189381680	卖	133	3500	465500	46.55	8/27/16 9:30
20160827	5189381680	买	126.5	3300	417450	0	8/27/16 9:30
20160827	5189381680	卖	126.5	3300	417450	41.75	8/27/16 9:30
20160830	5189381680	买	129.5	3200	414400	0	8/30/16 9:30
20160831	5189381680	卖	118	3200	377600	37.76	8/31/16 9:30
20160827	5189381680	买	119.25	3100	369675	0	8/27/16 9:30
20160827	5189381680	卖	119.25	3100	369675	36.97	8/27/16 9:30
20160830	5189381680	买	124	3100	384400	0	8/30/16 9:30
20160831	5189381680	卖	110	3100	341000	34.1	8/31/16 9:30
20160827	5189381680	买	130	3000	390000	0	8/27/16 9:37
20160827	5189381680	买	130	2990	388700	0	8/27/16 9:40
20160830	5189381680	买	117.1	2850	333735	0	8/30/16 9:40

交易日期	投资者编号	买/卖	成交价格	成交数量	成交金额	手续费	成交时间
20160830	5189381680	买	126	2750	346500	0	8/30/16 9:30
20160831	5189381680	卖	118	2701	318718	31.87	8/31/16 9:30

表11 异常交易者编号以及异常行为次数c（省略了63条信息）

于是，总共可以筛选出的异常交易者如上，异常交易较多的编号如下表所示

表12 异常交易较多者编号

8. 模型优缺点分析

8.1 优点分析

模型的优点有以下几个

1、快捷性，除了DBSCAN算法，其他的算法复杂度并不高，在实验的过程中，多数的数据在一分钟之内就可以通过计算得到。

2、全面性，从多个方面比较了分析指标，以及异常查询指标，使得可以用多种方法刻画交易特征以及查找异常交易。

8.2 缺点分析

1、深度缺失，挖掘的数据仅存于表面，没有挖掘更深层次的问题，比如：如何更好盈利，何样的异常操作会扰乱市场秩序等。

2、维数缺失，本文对于数据的处理多位于2维，仅主成分分析属于五维分析，维度越小，能表现的特征越少，需要寻找到更多联系，运用多维分析。

8.3 改进方向

本文有两个改进方向：

一、让信息变得更加具体、可以通过多加数据或利用已有的数据分析平仓、持仓等数据，使得信息更加完善，更加全面。

二、利用更高维度或更高的数量进行分析，可以利用更高性能的设备来处理数据，由于设备有限，仅仅能做出每一条的成交价格和成交数量的图，如果条件允许，可以对每一条信息进行聚类分析，图片如下：

参考文献

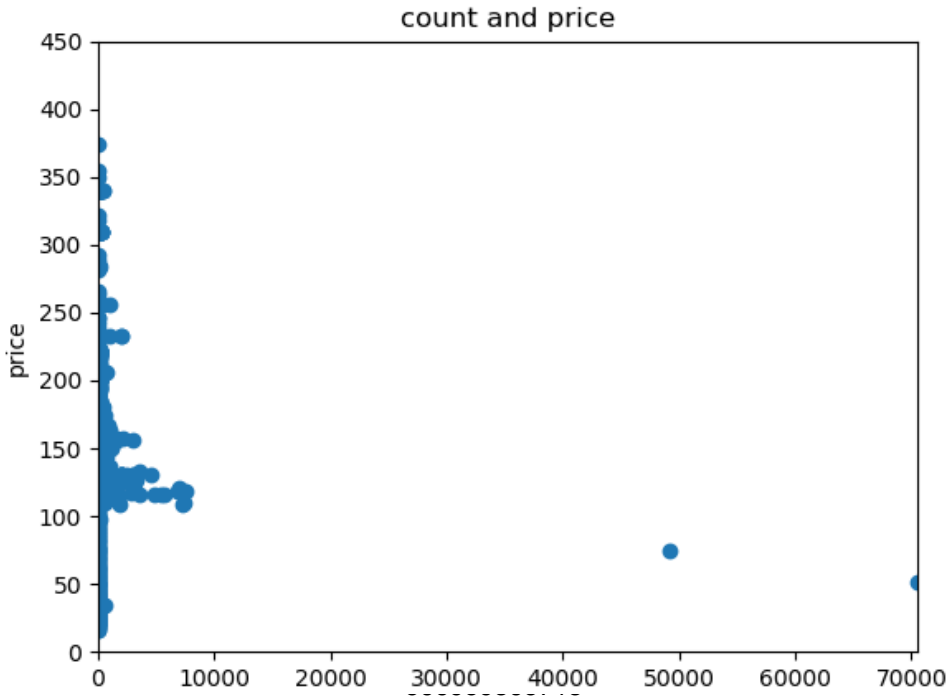

图12 所有交易的交易价格和交易数量的散点图

518938168077
588626595810
555502312363
518909953722
555583599195
169060600057
500832653169
588850713783
588681311961

[1]. Automated futures trading exchange, SW Wagner - US Patent 4,903,201, 1990 - Google Patents.
[2]. Futures trading, information and spot price volatility: evidence for the FTSE-100 stock index futures contract using GARCH, A Antoniou, P Holmes - Journal of Banking & Finance, 1995 - Elsevier
[3]. Subjective overachievement: Individual differences in self-doubt and concern with performance, Kathryn C. Oleson Kirsten M. Poehlmann John H. Yost Molly E. Lynch Robert M. Arkin, Journal of Personality, 2000 - Wiley Online Library
[4]. Density-based clustering, M Ester - Encyclopedia of Database Systems, 2009 - Springer
[5]. 机器学习, 周志华, 北京; 清华大学出版社, 2016.

附录

以下是Python实现的以上所有分析代码，具体的功能请看注释

```
#!-*- coding:utf8-*-
```

```
import xlrd
import math
import numpy as np
import matplotlib.pyplot as plt
from xlrd import xldate_as_tuple
from datetime import datetime

fname = "/Users/yongxinxu/Downloads/选拔题二/data.xlsx"
DealDay =
[1,2,3,4,5,6,8,9,10,11,12,13,15,16,17,18,19,20,22,23,24,25,26,27,29,30,31]
infoList = []
dailySales = []
dailyDeals = []
PriceAndSalesPerDealOfSomeDay = []
satisX = []
satisY = []

def getExcelData(filename):
    bk = xlrd.open_workbook(filename)
    shxrang = range(bk.nsheets)
    try:
        sh = bk.sheet_by_index(0)
        nrows = sh.nrows
        ncols = sh.ncols
        for i in range(3,nrows):
            row_data = []
            for j in range(ncols):
                ctype = sh.cell(i, j).ctype  # 表格的数据类型
                cell = sh.cell_value(i, j)
                if ctype == 2 and cell % 1 == 0:  # 如果是整形
                    cell = int(cell)
                elif ctype == 3:
                    # 转成datetime对象
                    date = datetime(*xldate_as_tuple(cell, 0))
                    cell = date
                elif ctype == 4:
                    cell = True if cell == 1 else False
                row_data.append(cell)
            infoList.append(row_data)
    except:
        print("Select sheet failed")
        return ""

def calculateDailySalesAndDeals():
    defaultDate = 20160801
    sales = 0
    deals = 0
    for item in infoList:
        date = item[0]
        if(date != defaultDate):
            dailySales.append(sales)
```

```
            dailyDeals.append(deals)
            sales = 0
            deals = 0
            defaultDate = date
        else:
            sales += item[5]
            deals += 1
    dailySales.append(sales)
    dailyDeals.append(deals)
    return ""

def generateLineChartOfDailySales():
    x = []
    for i in range(27):
        x.append(i + 1)
    plt.plot(x, dailySales, marker='*', mec='r', mfc='w')
    plt.xlabel('Day')
    plt.ylabel('Daily Sales')
    plt.title('Line Chart of Daily Sales and Day')
    plt.legend()

    plt.savefig("/Users/yongxinxu/Documents/ImageFromLab/MM2/1.png")

    plt.show()

    return ""

def generateLineChartOfDailyDeals():
    x = []
    for i in range(27):
        x.append(i + 1)

    plt.plot(x, dailyDeals, marker='o', mec='b', mfc='w')
    plt.xlabel('Day')
    plt.ylabel('Daily Deals')
    plt.title('Line Chart of Daily Deals and Day')
    plt.legend()

    plt.savefig("/Users/yongxinxu/Documents/ImageFromLab/MM2/2.png")

    plt.show()

    return ""

def getDataOneDay(numOfDay):
    satisX.clear()
    satisY.clear()
    dayNumber = DealDay[numOfDay-1]
    print(dayNumber)
    datInt = 20160800 + dayNumber
    for item in infoList:
        if(item[0] < datInt):
```

```
            continue
        elif(item[0] == datInt):
            satisX.append(item[4])
            satisY.append(item[5])
        else:
            break
    plt.scatter(satisX, satisY, s=25)
    # plt.xlim(10, 70515)
    # plt.ylim(0, 450)
    plt.axis()
    plt.title("count and price of "+(str(datInt)))
    plt.xlabel("price")
    plt.ylabel("count")

    plt.savefig("/Users/yongxinxu/Documents/ImageFromLab/MM2/Day"+str(datInt)
+".png")

    plt.show()

    return ""

def dist(a, b):
    """
    输入：向量A, 向量B
    输出：两个向量的欧式距离
    """
    return math.sqrt(np.power(a - b, 2).sum())

def writeDataOfDay(day):
    satisX.clear()
    satisY.clear()
    dayNumber = DealDay[day - 1]
    print(dayNumber)
    datInt = 20160800 + dayNumber
    for item in infoList:
        if (item[0] < datInt):
            continue
        elif (item[0] == datInt):
            satisX.append(item[4])
            satisY.append(item[5])
        else:
            break
    filename = 'data1.txt'
    # print(satisX)
    # print(satisY)
    with open(filename, 'w') as f:
        for i in range(len(satisY)):
            print(satisX[i], end=',')
            print(satisY[i])

def filterSuspect():
    lastData = infoList[0]
```

```
    for item in infoList:
        if(item == lastData):
            continue
        if((lastData[8] == item[8]) and (lastData[1]==item[1]) ):
            suslist.append(item)
        lastData = item

def filterSuspect2():
    suslist.clear()
    lastData = infoList[0]
    for item in infoList:
        if(item == lastData):
            continue
        if((lastData[8] == item[8]) and (lastData[1]==item[1]) and
(item[5]>500)):
            suslist.append(item)
        lastData = item
    # print(len(suslist))

def aggSuspect():
    agglist.clear()
    for item in suslist:
        found = 0
        for i in agglist:
            if(item[1] == i[0]):
                i[1] += 1
                found = 1
        if(found == 0):
            agglist.append([item[1],1])
    print(len(agglist))
    for v,m in agglist:
        if(m>10):
            print(v,end = " ")
            print(m)

def aggSuspect2():
    agglist.clear()
    for item in suslist:
        found = 0
        for i in agglist:
            if(item[1] == i[0]):
                i[1] += 1
                found = 1
        if(found == 0):
            agglist.append([item[1],1])
    print(len(agglist))
    for v,m in agglist:
        print(v,end = " ")
        print(m)

if __name__ == '__main__':
    '''读取信息,所有程序必须执行'''
```

```
    getExcelData(fname)
    '''读取信息,所有程序必须执行'''

    '''画出每天销售总量的折线图'''
    # calculateDailySalesAndDeals()
    # generateLineChartOfDailySales()
    '''画出每天销售总量的折线图'''

    '''画出每天交易量的折线图'''
    # calculateDailySalesAndDeals()
    # generateLineChartOfDailyDeals()
    '''画出每天交易量的折线图'''

    '''取得某一天单价和交易数量的信息'''
    # calculateDailySalesAndDeals()
    # getDataOneDay(27)
    '''取得某一天单价和交易数量的信息'''
    # calculateDailySalesAndDeals()
    # writeDataOfDay(11)
    filterSuspect2()
    aggSuspect2()
```

以下是DBSCAN聚类分析算法的Python程序

```
# -*- coding: utf-8 -*-

import numpy as np
import matplotlib.pyplot as plt
import math
import time

UNCLASSIFIED = False
NOISE = 0

def loadDataSet(fileName, splitChar='\t'):
    """
    输入：文件名
    输出：数据集
    描述：从文件读入数据集
    """
    dataSet = []
    with open(fileName) as fr:
        for line in fr.readlines():
            curline = line.strip().split(splitChar)
            fltline = list(map(float, curline))
            dataSet.append(fltline)
    return dataSet

def dist(a, b):
    """
    输入：向量A, 向量B
    输出：两个向量的欧式距离
```

```
    """
    return math.sqrt(np.power(a - b, 2).sum())

def eps_neighbor(a, b, eps):
    """
    输入：向量A, 向量B
    输出：是否在eps范围内
    """
    return dist(a, b) < eps

def region_query(data, pointId, eps):
    """
    输入：数据集, 查询点id, 半径大小
    输出：在eps范围内的点的id
    """
    nPoints = data.shape[1]
    seeds = []
    for i in range(nPoints):
        if eps_neighbor(data[:, pointId], data[:, i], eps):
            seeds.append(i)
    return seeds

def expand_cluster(data, clusterResult, pointId, clusterId, eps, minPts):
    """
    输入：数据集, 分类结果, 待分类点id, 簇id, 半径大小, 最小点个数
    输出：能否成功分类
    """
    seeds = region_query(data, pointId, eps)
    if len(seeds) < minPts: # 不满足minPts条件的为噪声点
        clusterResult[pointId] = NOISE
        return False
    else:
        clusterResult[pointId] = clusterId # 划分到该簇
        for seedId in seeds:
            clusterResult[seedId] = clusterId

        while len(seeds) > 0: # 持续扩张
            currentPoint = seeds[0]
            queryResults = region_query(data, currentPoint, eps)
            if len(queryResults) >= minPts:
                for i in range(len(queryResults)):
                    resultPoint = queryResults[i]
                    if clusterResult[resultPoint] == UNCLASSIFIED:
                        seeds.append(resultPoint)
                        clusterResult[resultPoint] = clusterId
                    elif clusterResult[resultPoint] == NOISE:
                        clusterResult[resultPoint] = clusterId
            seeds = seeds[1:]
        return True

def dbscan(data, eps, minPts):
```

```
    """
    输入：数据集，半径大小，最小点个数
    输出：分类簇id
    """
    clusterId = 1
    nPoints = data.shape[1]
    clusterResult = [UNCLASSIFIED] * nPoints
    for pointId in range(nPoints):
        point = data[:, pointId]
        if clusterResult[pointId] == UNCLASSIFIED:
            if expand_cluster(data, clusterResult, pointId, clusterId, eps,
minPts):
                clusterId = clusterId + 1
    return clusterResult, clusterId - 1

def plotFeature(data, clusters, clusterNum):
    nPoints = data.shape[1]
    matClusters = np.mat(clusters).transpose()
    fig = plt.figure()
    scatterColors = ['black', 'blue', 'green', 'yellow', 'red', 'purple',
'orange', 'brown']
    ax = fig.add_subplot(111)
    for i in range(clusterNum + 1):
        colorSytle = scatterColors[i % len(scatterColors)]
        subCluster = data[:, np.nonzero(matClusters[:, 0].A == i)]
        ax.scatter(subCluster[0, :].flatten().A[0],
subCluster[1, :].flatten().A[0], c=colorSytle, s=50)

def main():
    dataSet = loadDataSet('/Users/yongxinxu/Desktop/Papers/data/mdata.txt',
splitChar=',')
    dataSet = np.mat(dataSet).transpose()
    # print(dataSet)
    clusters, clusterNum = dbscan(dataSet, 2, 15)
    print("cluster Numbers = ", clusterNum)
    # print(clusters)
    plotFeature(dataSet, clusters, clusterNum)

if __name__ == '__main__':
    start = time.clock()
    main()
    end = time.clock()
    print('finish all in %s' % str(end - start))
    plt.show()
```