只供办公室使用

T1 _____

T2 _____

T3 _____

T4 _____

团队控制号

1906204

问题的选择

C

仅供办公室使用 F1

_____

F2 _____

F3 _____

F4 _____

2019

MCM/ICM 汇总表

阿片类药物危机与策略分析

总结

美国正在经历一场前所未有的阿片类药物危机。阿片类药物作为药物被广泛应用于各种治疗。快节奏的生活和激烈的社会竞争给人们带来了很大的压力。患有精神或身体疾病的人很可能会接受很多药物的治疗，他们很可能会对这些药物上瘾，尤其是阿片类药物。因此，近年来被证实的毒品鉴定案例也越来越多。

我们模型的目标是根据所提供的数据，找到已报道的合成阿片类药物和海洛因事件的传播和特征，并对现状做可能的解释和对未来病例分布的预测。具体而言，该模型的灵感来自于“推荐系统”。该模型的第一步与“推荐系统”具有相似的目的，即发现不同区域和药物之间的相似性和相关性。不考虑这个复杂问题背后的大量因素，直接处理数据是粗糙和不准确的。为了找出图形位置、婚姻状况、受教育程度、年龄分布等因素是如何促成阿片类危机的，一个合适的方法是，首先根据社会结构找到相似的区域，然后比较它们的药物传播和阿片类药物鉴定案例分布之间的关系，然后我们扩展我们的模型来服务于不同的目的，比如追踪药物来源，预测药物传播。

对于第一部分，我们基于相似性(如上所述)构建了一个加权有向图，并使用“Walk Around”策略来模拟药物传播过程，以追踪药物起始源。仍然基于相似度，我们使用 SVR 回归拟合数据随时间的分布，预测未来两年合成阿片类药物的鉴定和海洛因案件的分布情况。然后我们用 SVM 判别器来预测一个县是否会出现阿片类危机的风险。处于阿片类药物危机的县将面临药物滥用的持续增加。

对于第二部分，我们首先通过 Kmeans 算法对所有数据进行二值化，然后使用关联规则学习算法寻找导致阿片类药物和药物成瘾的因素。然后我们引入时间因子，通过相关分析方法进一步简化因子，找到滥用阿片类药物的人。经过这些步骤我们就可以找到所有的主要因素了。但是由于这些因子的数量较多，我们还需要使用 PCA 算法对输出因子进行减少，使预测模型更加简单。我们的模型发现，人口分布的差异对阿片类药物的滥用有巨大的影响。

对于第三部分，我们从第二部分中提取了三个主要特征，重新整合数据，并将其调用到我们之前的模型中，使我们的模型进行多维回归。我们设计了一些针对不同群体的策略，并使用我们的模型来验证我们策略的有效性。我们的模型发现，应该特别关注没有丈夫的女性户主以及 65 岁或以上的户主，提高整体教育水平也可以降低阿片类药物成瘾率。

关键词:推荐系统，回归，PCA，关联规则学习

阿片类危机分析与对策

团队# 1906204

2019 年 1 月 29 日

总 结

美国正在经历一场前所未有的阿片类药物危机。阿片类药物作为药物被广泛应用于各种治疗。快节奏的生活和激烈的社会竞争给人们带来了很大的压力。患有精神或身体疾病的人很可能会接受很多药物的治疗，他们很可能会对这些药物上瘾，尤其是阿片类药物。因此，近年来被证实的毒品鉴定案例也越来越多。

我们的模型的目标是发现报告的合成阿片类药物和海洛因事件的传播和特征，并根据所提供的数据对现状做出可能的解释和对未来病例分布的预测。具体而言，该模型的灵感来自于“推荐系统”。该模型的第一步与“推荐系统”具有相似的目的，即发现不同区域和药物之间的相似性和相关性。不考虑这个复杂问题背后的大量因素，直接处理数据是粗糙和不准确的。为了找出图形位置、婚姻状况、受教育程度、年龄分布等因素是如何促成阿片类危机的，一个合适的方法是，首先根据社会结构找到相似的区域，然后比较它们的药物传播和阿片类药物鉴定案例分布之间的关系，然后我们扩展我们的模型来服务于不同的目的，比如追踪药物来源，预测药物传播。

第 1 部分基于相似性构建加权有向图(如上所述)，采用“走动”策略模拟药物传播过程，追踪药物起始来源。在相似性的基础上，利用 SVR 回归拟合数据随时间的分布，预测未来 2 年的合成阿片类药物鉴定和海洛因案件分布情况。然后，我们使用 SVM 判别器来预测一个县是否会面临阿片类药物危机的风险。处于阿片类药物危机的县将面临药物滥用的持续增加。

对于第二部分，我们首先通过 Kmeans 算法对所有数据进行二值化，然后使用关联规则学习算法寻找导致阿片类药物和药物成瘾的因素。然后我们引入时间因子，通过相关分析方法进一步简化因子，找到滥用阿片类药物的人。经过这些步骤我们就可以找到所有的主要因素了。但是由于这些因子的数量较多，我们还需要使用 PCA 算法对输出因子进行减少，使预测模型更加简单。我们的模型发现，人口分布的差异对阿片类药物的滥用有巨大的影响。

对于第三部分，我们从第二部分中提取了三个主要特征，重新整合数据，并将其调用到我们之前的模型中，使我们的模型进行多维回归。我们设计了一些针对不同群体的策略，并使用我们的模型来验证我们策略的有效性。我们的模型发现，应该特别关注没有丈夫的女性户主以及 65 岁或以上的户主，提高整体教育水平也可以降低阿片类药物成瘾率。

关键词:推荐系统，回归，PCA，关联规则学习

内 容

1 介绍

1.1 背景

一场毒品危机正在美国爆发，关于合成和非合成阿片类药物的使用，无论是用于治疗和控制疼痛，还是用于娱乐目的。疾病控制中心等官方机构正在努力控制传染病的传播，减轻阿片类药物的负面影响，尤其是对受过良好教育的高学历人群和老年人。

近日，DEA/国家法医实验室信息系统(National Forensic Laboratory Information System)发布了一份涉及“药物鉴定结果”的数据量很大的年度报告，引起了广泛关注。为了更深入地了解这一问题，研究主要集中在美国五个州的单个县:俄亥俄州、肯塔基州、西弗吉尼亚州、弗吉尼亚州和田纳西州。相关数据涵盖了这五个州每个县 2010-2017 年合成阿片类药物和海洛因的药物鉴定计数。其他数据包含美国人口普查局(U.S. Census Bureau)的摘录，代表了 2010-2016 年每一年为这五个州的县收集的一组共同的社会经济因素。

2 问题分析

2.1 数据预处理

检查“MCM_NFLIS_DATA. data”中的所有数据。Xlsx”，很容易算出来，这五个州有 69 种不同的阿片类药物，462 个不同的县。更具体地说，一些县的很多种毒品几乎没有或很少有“毒品报告”，这意味着某一个县的某一种毒品的样本是远远不够的，这可能会误导我们分析毒品的传播情况和预测。为了使传播特征和趋势更明显地观察到，我们使用 K-means 聚类对数据进行预处理，将其划分为一些更大的区域，称为“区域”，区域内的药物报告是当前区域内所有居民的药物报告的总和。

2.1.1 用 K-means 法划分区域

为了适当地实施区域化，我们应该让“Zone”既不太大也不太小。通过多次尝试，我们选择了将参数 k 设置为 100 的最佳情况。从技术上讲，我们只是使用了基本的 K-means 算法，所有 100 个集群都有相同的半径，每个集群的县数量大致在 2 到 8 之间。聚类前，将“FIPS_Combined”变换为对应的经纬度，然后应用 K-means 算法进行聚类。最后，我们整合了同一区域内每种药物的所有药物鉴定报告。下图是预处理数据的格式，

图 1:Data by Zone

YYYY	区	substanceName	DrugReport	TotalDrugReportZone
2010	0	丙氧芬	1	327
2010	0	羟考酮	4	327
2010	0	氢可酮	9	327
...	...	...	...	...

将数据转化为这种形式后，我们可以从更高的层面分析药物传播过程，然后聚焦几个特殊区域，方便地深入研究。下图展示了这些区域的可视化。

图 2:ZoneCounty Map

区	纬度	经度	FIPS_Combined
0	37.6189	-75.8400	51001、51620、51115、51119、51131
1	41.5442	-84.0702	39001、39047、39051、39095、39173、39171
1	39.9813	-77.2332	42001、42041、42055、42133
...	...	...	...

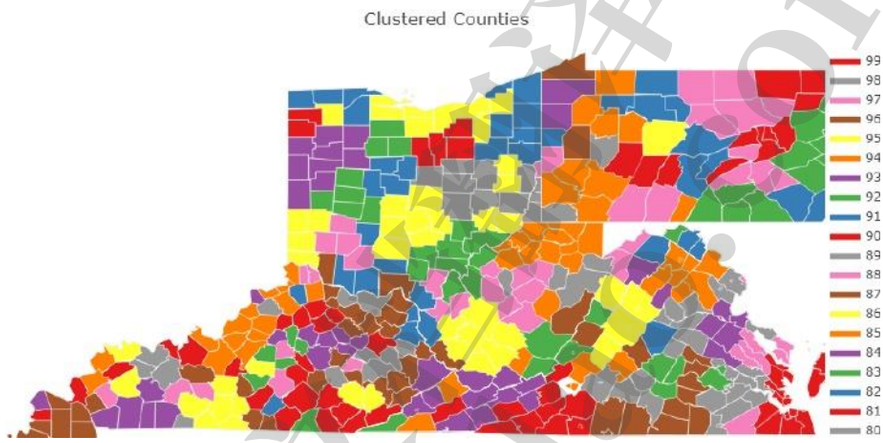


图 3:聚集的区域

(注意:每个区域应该有一种颜色，但我们没有 100 种颜色，我们只用 20 种颜色来区分)

2.2 基本假设

- 阿片类药物成瘾分布趋势随时间变化。在做预测时，我们会考虑时间因素。最新的趋势会有更高的影响因子。
- 阿片类药物滥用的蔓延取决于图形位置。我们假设毒品的传播主要是从一个地方到相邻的地方，不会跨越很大的距离。
- 在处理数据缺失时，我们将其视为平均值。

2.3 概述

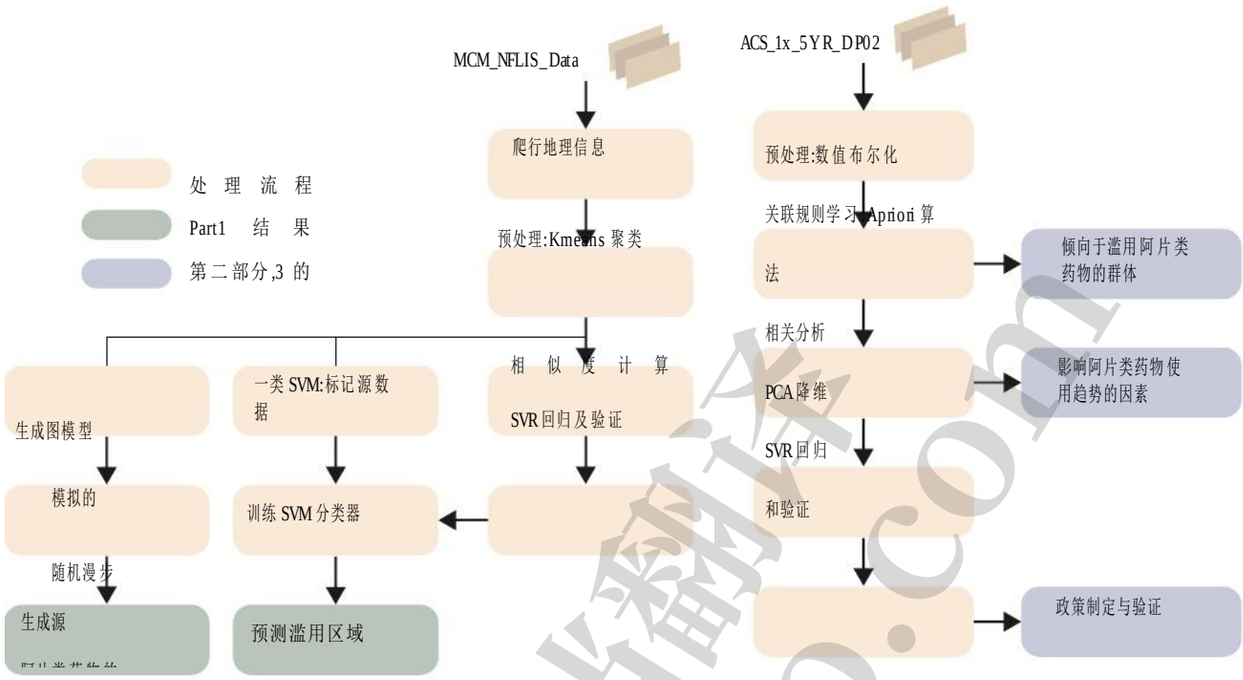


图 4:模型概述

上图介绍了我们模型的主要工作流程。PART1 工作流的左边部分分为数据预处理、模型训练、模型验证。右边部分是 PART2、3 的工作流，主要分为数据预处理、相关性分析、关联规则学习、模型验证、规则制定。

3 模型

3.1 基本型号说明

为了了解报告的合成阿片类和海洛因事件在五个州及其县之间的传播和特征，确定来源并预测未来可能的药物使用情况，我们决定使用推荐系统。

3.1.1 推荐系统

想象一下，你刚刚看完一部电影，对这部电影印象深刻，甚至给了 5 颗星的评价。电影网站接受你的评价，它会根据你最近的行为，比如评价行为，自动给你推荐一些你可能喜欢的电影。推荐系统的核心部分就是找到和你有相同偏好的相似用户，这些相似用户可能会看到一些你没看过但他们觉得不错的其他电影。这样，系统就会给你推荐这些电影。

我们的模型反映了“推荐系统”的主要思想，即寻找相似区域。这些相似的区域会在很大程度上分享报道的合成阿片类药物和海洛因事件的相似传播和特征。这些相似的区域有助于我们放大传播和特征，并使预测更加准确。

3.1.2 数学公式

由于有 100 个区域和 69 种不同的阿片类药物，我们创建了一个 100×69 矩阵。每一行代表一个区域编号，每一列代表一个阿片类药物编号。那么这个矩阵中的 cell(i, j)将代表区域 i 中编号为 j 的药物的一些信息，这些信息会因为不同的目的而有所不同。

3.1.3 相似度设计

我们如何判断两个区域是否相似，与模型的性能密切相关。在我们的模型中，我们仔细考虑了三个主要因素。(1)随时间的趋势(2)地理因素(3)药物鉴定规模。

- 因子(1):基于上面的数学公式，我们在每个单元格(i, j)中存储一个列表，其中包含第 i区第 j种药物的 6 年数据(2010-2016)。在计算任意一对区域之间的相似度时，我们找到每种药物的相关性，然后进行加权求和。相关性的定义如下:

$$Corr(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$

  使用 python 中“Numpy”包中的 Numpy .corrcoef()方法可以快速方便地实现计算。

- 因子(2):在实践中，地理因素在药物传播中起着至关重要的作用。人们倾向于从一个区域迁移到相邻的区域。这一过程可能引发药物扩散，导致一个区域内的药物识别情况更容易与相邻区域相似。根据经度和纬度，我们可以很容易地计算出任何区域对之间的距离。根据我们的测试，将距离控制在 200 公里以内可以使模型表现良好。

- 因素(3):药物识别的大小是另一个需要考虑的重要因素。与药品鉴定案例规模较大的区域相比，药品鉴定案例数量较少的区域具有稳定的趋势，增长或下降可能会发生突变。在计算相似度时，我们将所有 6 年的数据相加，并调用时间衰减因子。

$$Total = \sum_{i=2010}^{2016} data_i \nu^{i-1}$$

  当 ν = 0.7 时，数据越晚，数据就应该越重要。

对于每个区域，我们使用因子(2)找到“封闭区域”，并根据因子(1)和(3)给每个封闭区域打分。通过排序，挑选出得分最高的 10 个区域。

3.2 在图中“游走”寻找原点

在这一部分，我们主要是想找一些阿片类药物的来源。想象一下，我们正在浏览网页，我们会打开一个网址，浏览页面，然后根据页面上的 URL 选择喜欢的话题。我们会沿着页面的链接继续打开新的网页。阿片类药物的传播也有类似的行为。一种阿片类药物从一个城市流行开来，慢慢影响周围的城市。但由于城市特点不同，特定阿片类药物在周边城市的传播速度会有很大差异。**这里我们假设特征相似的城市在阿片类药物传播时，会有相似的模式。**因此，阿片类药物向邻近城市传播的概率会有所不同。这个概率是用上面几节中城市之间的相似性来描述的。

我们模型的第一步是构建阿片类药物的传播图模型。假设我们正在寻找海洛因的来源。我们模型的第一步是构建阿片类药物的传播映射模型。

我们选取了任意年份的海洛因数据。图中的每个节点代表一个聚类点，每个节点的权值代表该地点吸食这种毒品的人数。当节点的数量 A (W_A)高于另一个节点 B (W_B)，生成一条从 a 到 B 的有向边(a - >B)。边的权值为 A 和 B 的相似度($sim(A,B) = p_{A->B}$)。为海洛因生成的图表如下所示。

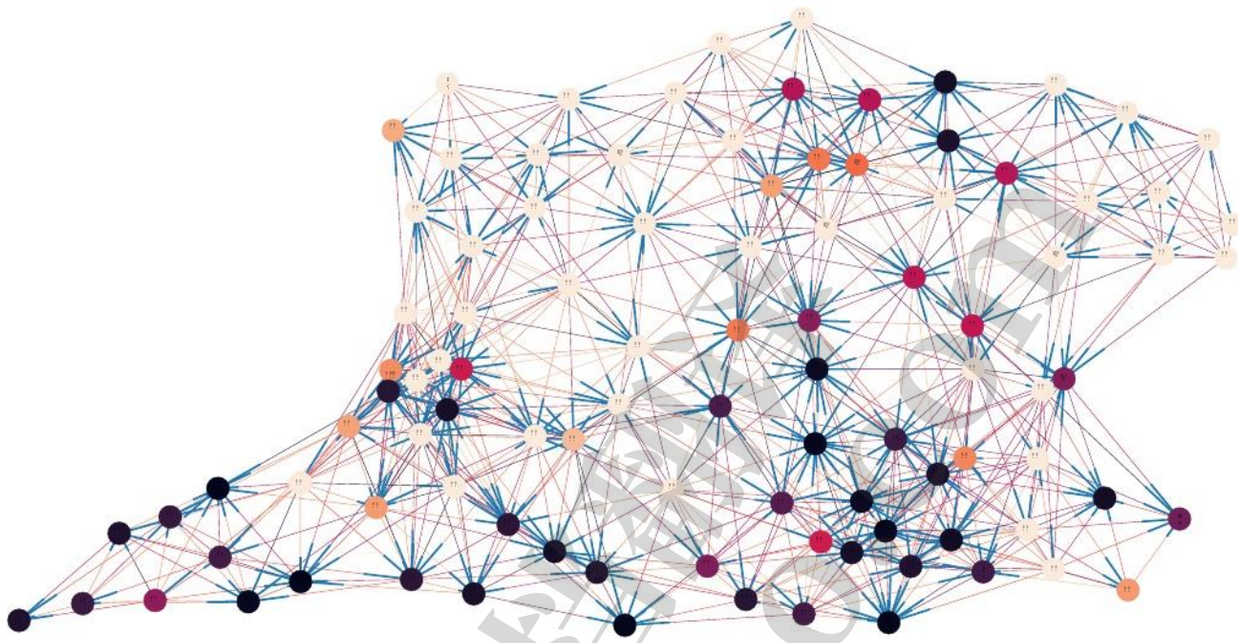


图 5:药物传播的图形模型。节点的浅色代表人数较多，边缘的颜色代表转移概率的大小。

第二步是根据上面的图模型找到可能成为药物来源的节点。我们称之为候选集。CandidateSet = {node1, node2, node3，…}。在图论中，图中一个顶点的度(或价)是附属于该顶点的边的数量，循环计算两次。在这里，我们将定义改为从顶点散射的边数。对于每个我们计算其度(Deg(v))的节点，我们选择最大的 5 个节点并将它们存储在候选集中。

第三步是使用随机游走的模拟方法来模拟药物的传播行为。分别模拟候选集中的节点，每个节点模拟 100 次，记录其他节点的访问次数(n_i)，按下式计算每个节点的得分。得分最高的就是我们想要找到的源。

$$Score_i = \sum_{j=0\ and\ j\neq i}^{99} \text{nj* Wi* pi ->j i}\in \text{候选集}$$

Source = index(max(Score) Score ∈ {Score1, Score2, Score3，…})

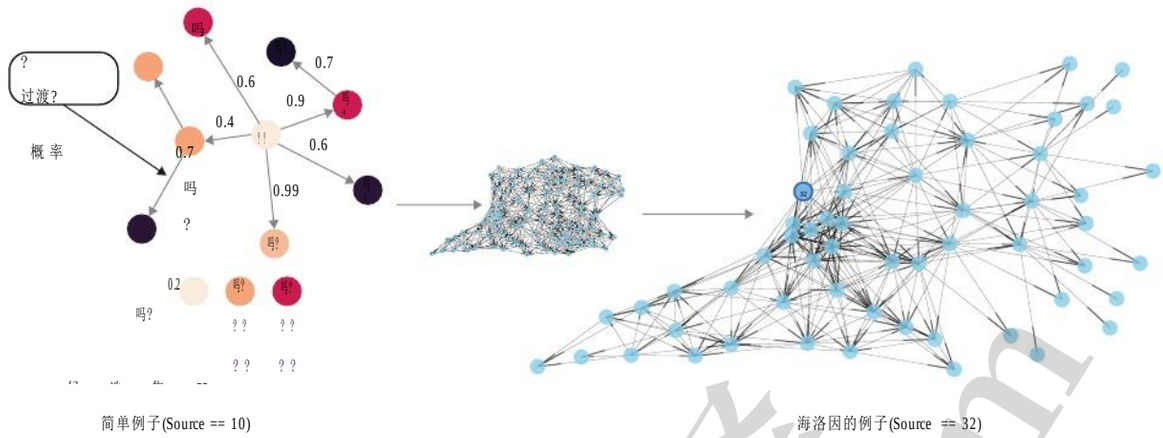


图 6:左图为随机行走仿真算法的实现。右边的例子显示，模拟之后，海洛因的来源在第 32 区。

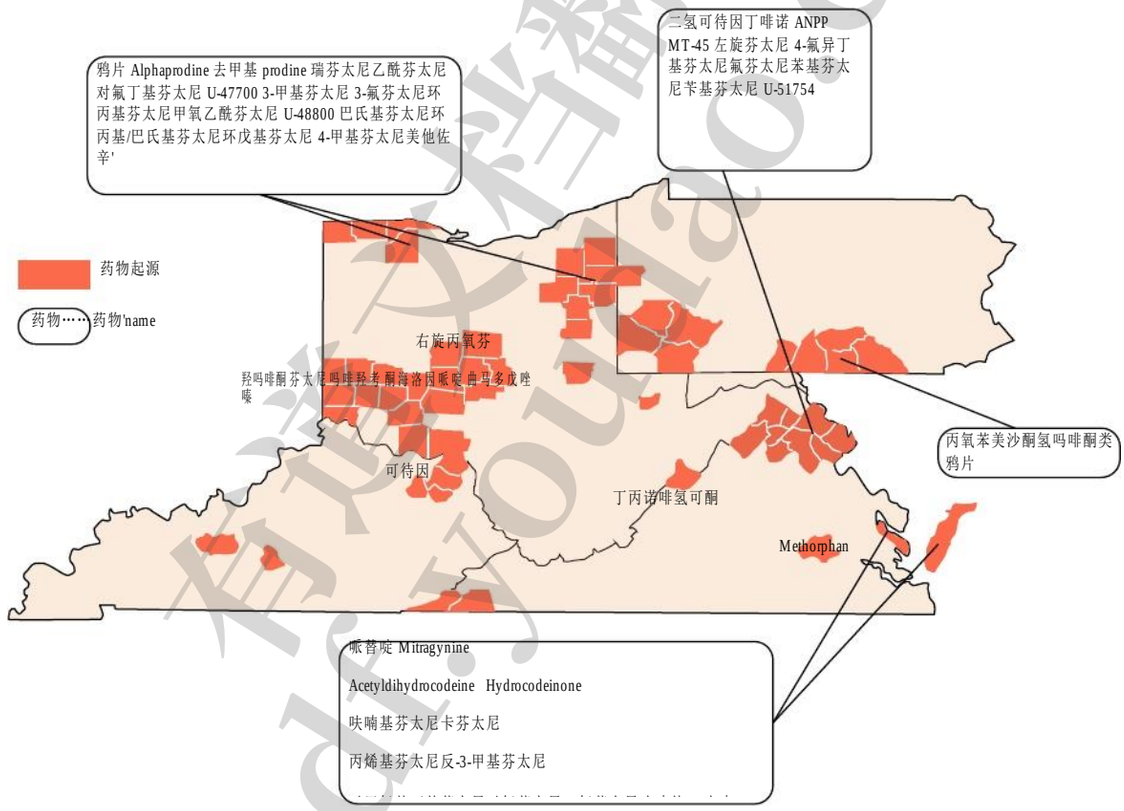


图 7:所有阿片类药物的来源

结果分析，上图显示了问题中出现的所有阿片类药物的来源。由此我们可以分析出许多有趣的现象。**交通因素:我们可以看到很多阿片类药物的来源，在海洋附近，或者在五大湖附近。**由于靠近海洋，更容易从国外进口非法毒品或阿片类药物，使得这些地区成为大量阿片类药物的来源。**法律因素:我们可以看到，俄亥俄州是很多毒品的来源地。**出现这种情况的原因，可能与当地法律法规还存在漏洞，政府的监管力度不强有关。

3.3 SVM 判 别器在评价药物识别阈值水平中的应用

在一些县，阿片类药物识别计数数量明显高于该州其他地区。这表明该县存在更严重的药物滥用问题。大量的毒品事件也反映了人口的密集，这对毒品的传播有很大的影响。

在大多数情况下，一个异常值数据要么揭示了一个错误，要么应该突出。这里我们假设一些县已经面临阿片类药物危机。通过观察，发现毒品报告较少的县与报告较多的县之间存在一定的差距。所以我们用 SVM 来区分这些县。毒品报告较多的县面临阿片类药物危机的风险

3.3.1 SVM判 别 器

SVM 判别器用于区分有陷入阿片类药物危机风险的县和其他县，给定阿片类药物事件的预测数量。对于二维值，SVM 鉴别器训练一条超平面线来区分两个不同的类。超平面由 ω 给出 ^{T} X + γ = 0，其中，ω_{T}的法向量是

超平面，x是平面上的点。一个有效的超平面满足:

$$\omega^{t}\boldsymbol{x}_i+\gamma>0 \quad \text{for } y_i=1$$
$$\boldsymbol{\omega}^{T}\boldsymbol{x}_i+\gamma<0 \quad \text{for } y_i=-1$$

支持向量是最接近超平面的点，它们是两个不同集合的边。

3.3.2 一 类 SVM

SVM 判别器是一种有监督学习模型，但由于训练数据没有标注，SVM 必须采用无监督学习模型。因此，引入了单类 svm。一类 SVM 方法的目标是在特征映射空间尽可能地将数据点与源点分离。首先使用 Gaus-

将数据点投射到特征上的 Sian 内核 空间，那我们要解 $_{Pn}$ 遵循二次规划

$$\min_{w,\zeta_i,\rho}\frac{1}{2}\|w\|^2+\frac{1}{\nu n}\sum_{i=1}^{n}\zeta_i-\rho$$

将投影数据点与原点分离:

$$\zeta \geq 0$$

$\nu \in (0,1)$确定异常值的上限。最后训练特征地图空间中的一条线，将原点与数据分离。然后将这条线和数据变换回未投影的空间，线外是离群值。

3.3.3 阈 值 识 别

本文提出的 SVM 鉴别器结合了一类 SVM 和线性 SVM。首先使用 one-class svm 对 2010—2017 年的阿片类药物报告进行离群值判定。这些异常值表明存在阿片类药物危机危险的县。然后使用一类 SVM 的标记数据训练线性 SVM 阈值作为阿片类药物滥用预警阈值。由于药物滥用问题不仅表现在阿片类药物的使用上，也表现在其他药物的使用上，一个阿片类药物事件相对于药物报告总量较多的县，或者一个没有阿片类药物事件的药物报告总量较多的县，都应该被认为面临危机。因此，我们以某县的毒品总报告为 y 轴，阿片类药物总使用为 x 轴来包括上述情况。稍后，我们将使用这个训练过的模型作为过滤器，预测的毒品报告数作为输入，危险水平作为输出。下图是训练过程，高亮线为超平面:

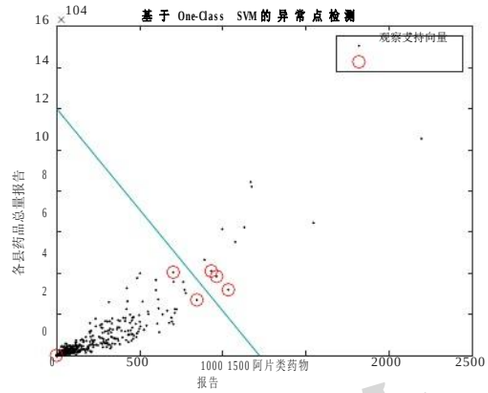


图 8:SVM 鉴别器

3.4 使用 SVR 回归进行预测

为了预测特定药物何时何地达到药物识别阈值，将"推荐系统"与支持向量回归(support Vector Regression，SVR)相结合。

3.4.1 算法细节伪代码

算法一结合 SVR 回归的推荐系统

1: # 100×69×6 信息矩阵存储了每个区域每种药物 6 年的药物报告。
2:特征←空字典
3:为各区子做
4: simi_zones←z $_{,i}$ 的所有相似区域
5: for Each Drug d $_{,j}$ 做
6: feature←空集#存储回归结果
7:获取 SVR 回归作为自己的功能，推送到列表功能
8: for Each Training Sample s_k 做
9:将其样本的 SVR 回归作为一个特征，推送到列表特征
10: end for
11: # weight←空集
15: end for
16:归一化权重
17:存储特征[z $_{,i}$] [d_j]←[特征、重量]
18:端为
19:结束

解释

- 我们首先创建一个 100×69×6 InfoMatrix 来存储每个区域中每种药物 6 年的药物报告。然后对于每个区域，我们找到 10 个最相似的区域，每个区域都会提供一个样本。我们使用 SVR 来拟合每个样本，作为特定区域内特定药物的一个特征。在我们获得所有特征后(总共 11 个，1 个来自自身，10 个来自样本)，我们通过乘以一个超参数-权重系数来修改每个特征，以使模型更加合理和稳定，能够预测未来。

- 基于机器学习的思想，我们将数据分为 3 个集，训练集(数据 2010-2015)，hold -out 集(数据 2016)和测试集(数据 2017)。我们从训练集中提取特征，并从保留集中计算超参数。最后，使用测试集来验证我们模型的性能。

- 另一个问题是我们如何设置超参数。直观地说，一个预测值和真实值之间越接近，这个特征就越重要，大致满足反比。在尝试了很多次之后，我们将权重的格式设置为 $\frac{1}{0.01+(\text{difference})^2}$ 哪个的权重几乎最小

  平方误差时，从测试集的日期进行测试。

- 在我们找到超参数的最佳格式后，我们将数据分为 2 部分，训练集(2010-2016)和保留集(2017)，然后再次训练模型，这将用于后续的预测。

3.5 SVR 的优点

以下原因解释了我们选择 SVR 而不是其他方法进行回归，

- 适合多维回归
- 计算速度快(python 有现成的包)
- 没有过拟合

3.5.1 确定 SVR 的最佳参数

SRV 是一个成熟的模型，有许多参数用于不同的目的。最常见的模式是“rbf”和“poly”。对于“poly”模式，我们可以将度数设置为任何我们想要的整数。所以我们测试这个基本设置来找到最好的模型。下面的测试结果显示了每对参数设置的最小二乘误差(2017 年预测日期与 2017 年真实数据之间的差值)，

图 9:Poly Tests(聚检验)

伦敦政治经济学院 学位 内核	1	2	3.	4
“聚”	47116.0	39458.0	28908.0	48322(慢)

图 10:RBF 测试

伦敦证交所 C γ	0.1	0.5	1	2
1 e0	50332.0	50363.0	50496.0	50692.0

1 e1	47804.0	48434.0	49189.0	49968.0
1 e2	40161.0	41977.0	44438.0	46868.0
1 e3	30788.0	32755.0	37597.0	42033.0

经过测试，我们发现将 kernel 设置为“poly”，将 degree 设置为 3 是最好的选择。(注意:当内核模式为“poly”时，我们仍然需要考虑参数 C，但在我们的测试过程中，C 对结果几乎没有影响，所以我们只使用默认值。)

3.6 未来评估县域阿片类药物问题

3.6.1 模型预测流程

上面介绍的 SVR 预测器使用 2010-2017 年 NFLIS 数据，按地区预测阿片类药物使用总数。由于给定的数据仅涵盖 7 年，基于这些数据对未来的预测缺乏保真度，我们仅预测 2018 年和 2019 年的阿片类药物识别计数。结果如下图所示，颜色越深表示阿片类药物问题越多。

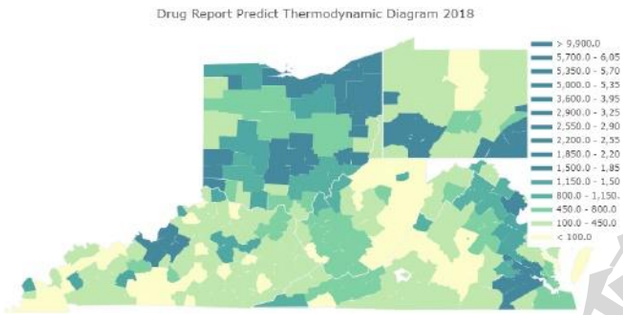


图 11:2018 年阿片类药物预测报告

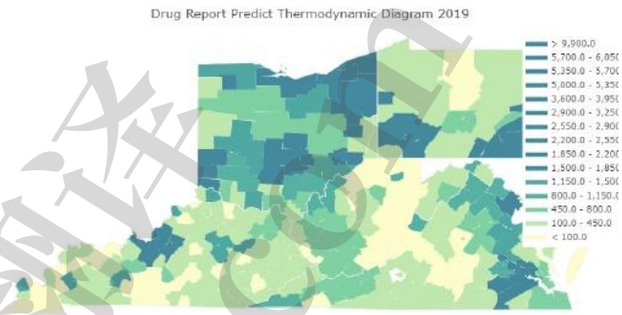


图 12:2019 年阿片类药物预测报告

然后我们根据 2010-2017 年的数据训练 SVM 鉴别器，并将预测结果设置为输入。输出如下图所示，突出显示的区域是阿片类危机风险的区域。

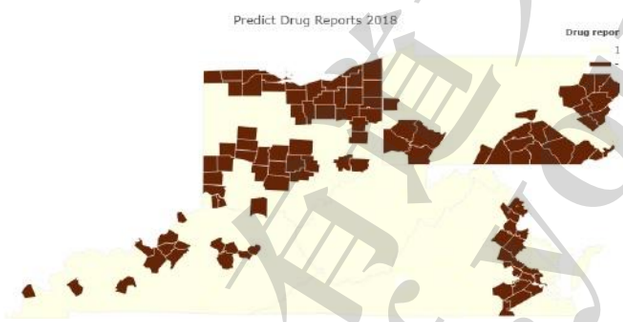


图 13:2018 年风险预测区域

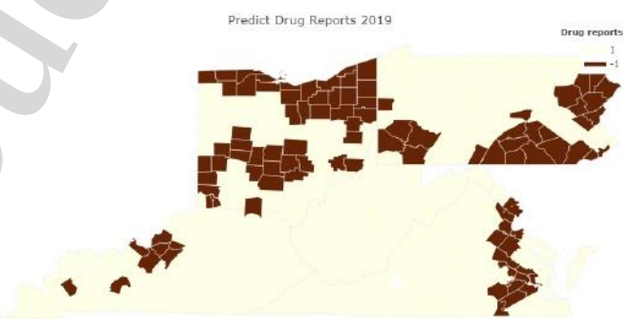


图 14:2019 年风险的预测区域

宾西法尼亚	亚当斯坎伯兰富兰克林约克
俄亥俄州	阿什塔比拉湖
宾西法尼亚	切斯特兰开斯特
俄亥俄州	凯霍加 • 格奥加亨利 • 诺布尔峰会

图 15:2017 年处于危机中的县

2018 年的结果在附录中。阿片类事件总体呈高速增长趋势。在某些地区，这一数字可能会增加数千起。然而，我们也注意到 2018-2019 年部分地区的预测有所下降，这些地区如下图所示:

肯塔基州	克拉克·巴拉德·费耶特·麦迪逊 梅尼菲·鲍威尔·特林布尔

图 16:2018-2019 年可能处于危机状态的县

操作分析

在我们的模型得出的结果中，我们可以看到某些地区的阿片类事件有所下降，但总体上增加了阿片类事件。在已经面临阿片类药物危机的地区，阿片类药物滥用问题越来越严重。存在阿片类药物问题风险的县与阿片类药物开始传播的县并不匹配。然而，阿片类药物开始扩散的县周边的县，阿片类药物事件的数量往往会增加。据我们估计，远离其他毒品问题地区的地区数量在下降。因此，我们可以得出这样的结论，阿片类药物问题似乎是区域性的。

除此之外，我们还注意到环湖海域的阿片类药物问题更为严重。这可能是由于这些县人口密度较高造成的。人群越大，阿片类事件发生越多，人口密度越大，毒品传播也越容易。

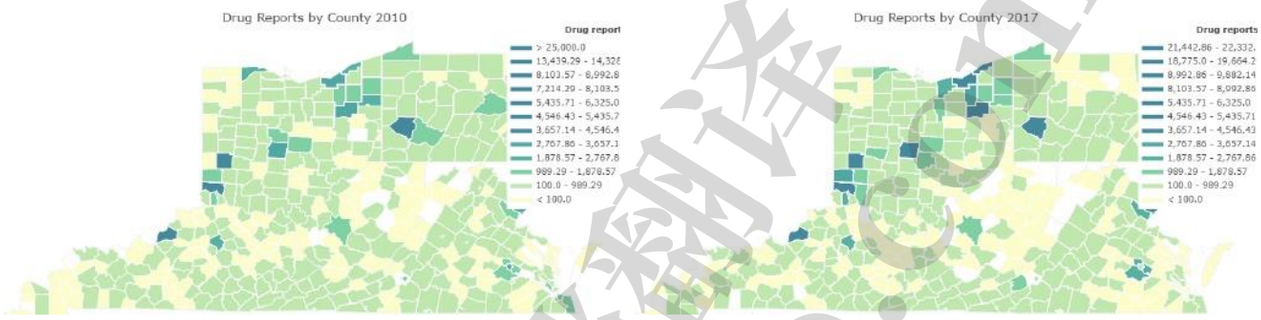


图 17:2010-2017 年毒品事件报告变异情况

从历史数据中，我们可以看到类似的趋势，在已经面临毒品问题的地区，毒品问题越来越严重。这种情况也发生在人口较多的地区。

3.7 选择因素:使用美国人口普查的社会经济数据

在本文中，我们使用美国人口普查社会经济数据集来选择与阿片类药物高度相关的因素。由于数据量大，很难手动选择特征。所以这里我们使用机器学习算法来自动选择因子。我们通过三个步骤来输出最终提取的特征。在第一步中，我们使用关联规则学习算法来选择与阿片类药物相关的人口统计学特征。在第二部分中，我们利用第一步中获得的特征，通过相关分析找出与阿片类药物增长正相关的因素。由于第二部分的数据维数仍然很大，所以在第三步中我们使用 PCA 算法对第二步获得的数据进行维数降低。我们使用第三步中输出的多维向量作为最终的特征。

3.7.1 关联规则学习:Apriori

关联规则学习是一种基于规则的机器学习方法，用于发现大型数据库中变量之间有趣的关系。它的目的是使用一些兴趣度的度量来识别在数据库中发现的强规则。关联规则学习首先应用于在超市销售点系统记录的大规模交易数据中发现产品之间的规律。例如，在超市的销售数据中发现的规则{面包，茶->牛奶}将表明，如果一位顾客同时购买面包和茶，那么他们很可能也会购买牛奶。

寻找阿片类药物和人口统计数据之间的关系也可以使用关联规则学习来解决。在这里，我们将每个城市想象成一个“购物篮”。每一个人口都可以被视为一种商品。我们还把阿片类商品的数量信息加入到“购物篮”中，我们的问题就转化为，当顾客买哪一类商品的时候，他就有大概率去买‘阿片类商品’，也就是说，寻找一个城市的哪种人口特征痴迷，这个城市的

关于阿片类药物的犯罪将急剧上升。

第一步数据预处理。由于我们的关联规则学习算法需要布尔形式的输入，并且我们的数据是连续形式，这里我们首先将数据库中的数据转换为布尔形式。由于每个人口有四种类型的数据(HCxx_VC01, HCxx_VC02，HCxx_VC03，HCxx_VC04)，这里我们使用百分比形式的数据(HCxx_VC03)来消除不同城市人口的影响。然后使用 Kmeans 算法将每个人口统计学特征分为两类，最终生成一个新的布尔数据表。

ID 研究了	**HC03_VC04**	**HC03_VC06**	**HC03_VC07**	**HC03_VC08**	**HC03_VC09**	**…**	**药物**
21001	0	1	0	0	0	…	1
21003	0	0	0	0	1	…	0
21005	0	1	0	0	0	…	0
21007	0	0	0	0	1	…	1
…	…	…	…	…	…	…	…

表 1:数据预处理实例，数值布尔化

第二步是建立公式化描述。令 I = {I $_{.1}$,我 $_2$，……,我 $_n$}是 n 个二进制属性的集合，称为项。这里 I = {HC03_V C04, HC03_V C06, HC03_V C07, HC03_V C10，…药物}。令 D = {t_1t_2，……t_m}是一组称为数据库的事务。这里 D ={21001, 21003, 21005，…} D 中的每笔交易都有唯一的交易 ID，并包含 i中项目的子集。规则被定义为形式的蕴涵:

X⇒ Y，其中 X，Y⊆ I .

每条规则由两组不同的项组成，也称为项集，X 和 Y，其中 X 称为前项或左手边(LHS) andY 结果或右手边(RHS)。为了从所有可能规则的集合中选择有趣的规则，使用了对各种显著性和兴趣度量的约束。这里我们使用支持度和置信度上的最小阈值方法，我们首先需要定义支持度和置信度的定义。X 相对于 T 的支持度定义为包含项集 X 的数据集中 T 的事务占比。

$$\mathrm{supp}(X) = \frac{|\{t \in T; X \subseteq t\}|}{|T|}$$

置信度是指规则被发现为真的频率。对于一组交易 T，一条规则 X⇒ Y 的置信度值是包含 X 且包含 Y 的交易所占的比例。置信度的定义为:

$$\mathrm{conf}(X \Rightarrow Y) = \frac{\mathrm{supp}(X \cup Y)}{\mathrm{supp}(X)}$$

规则的提升被定义为:

$$\mathrm{lift}(X \Rightarrow Y) = \frac{\mathrm{supp}(X \cup Y)}{\mathrm{supp}(X) \times \mathrm{supp}(Y)}$$

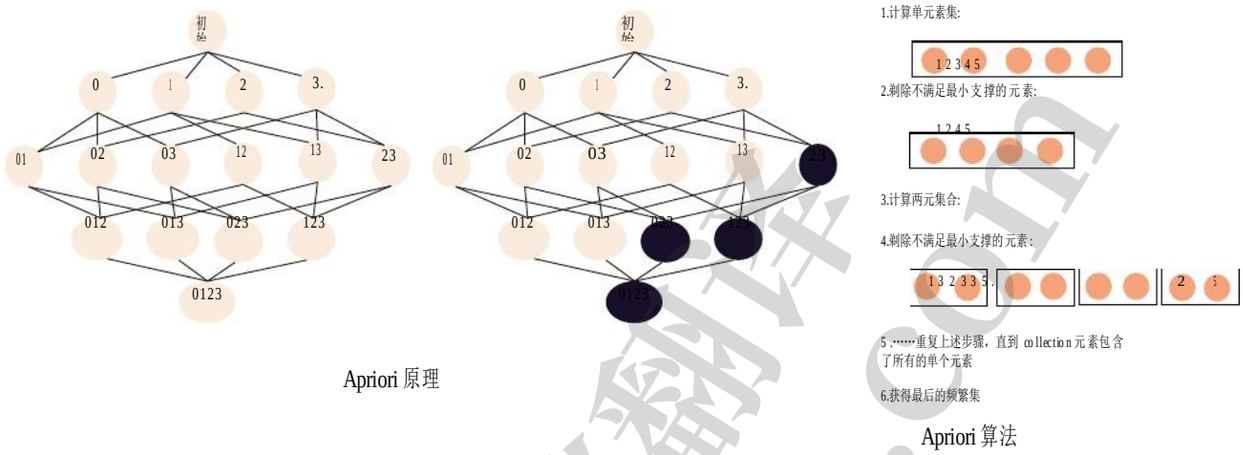
如果升力是> 1，那就可以让我们知道这两次事件相互依赖的程度，并使这些规则对于预测未来数据集中的结果潜在有用。这代表升力越大，关联规则的可信度就越强。

第三步，找到与阿片类药物相关的关联规则。使用 Apriori 算法求解频繁集。然后使用频繁项集构造一个满足用户最小信任的规则。Apriori 算法的具体方法是:首先找出频繁的 1 项集合，记为 L_1;然后用 L1 生成候选集 C2，并判断 C2 中的项来挖掘 L2，这是一个频繁的 2 项集;循环，直到找不到更频繁的 k 项。每挖一层 L_k，你需要扫描整个数据库。该算法利用了 Apriori 原则:

•一个项目集是频繁的，那么它的所有子集也是频繁的

•如果一个项集是一个不频繁项集，那么它的所有超集也是不频繁项集

利用这个原理，你可以避免项集数量的指数级增长，从而在合理的时间内计算出频繁项集。



Apriori 原理　　Apriori 算法

图 18:Aporior 原理和 Aporiori 算法

在图中我们可以看到，已知的灰色部分{2,3}是一个非频繁项集，那么有了上面的知识，我们就可以知道{0,2,3}{1,2,3}{0,1,2,3}都是非频繁项。也就是说，在计算了{2,3}的支持度之后，知道它是不频繁的，就不需要再计算{0,2,3}{1,2,3}{0,1,2,3}支持度了，因为我们知道这些集合不会满足我们的要求。下面我们基于频繁项集生成关联规则。和我们之前生成频繁项集一样，我们可以为每个频繁项集生成许多关联规则。如果可以减少规则的数量来保证问题是可以解决的，那么计算就会好很多。通过观察，我们可以知道，如果一个规则不满足最小可信度要求，那么该规则的所有子集都将不满足最小可信度要求。与之前的 Apriori 算法一样，利用关联规则的上述属性属性，可以减少需要测试的规则数量。

第四部分产生了对阿片类药物传播影响最大的最终因素。由于直接对所有因子运行 Apriori 算法是不现实的，这里我们将表中的数据分成 16 个类别，对每个类别运行 Apriori 算法，最终的结果是。

	1	2	3.	4
家庭	HC03_VC11	HC03_VC09	HC03_VC12	HC03_VC07
的关系	HC03_VC31	HC03_VC30	HC03_VC25	
婚姻	HC03_VC35	HC03_VC42	HC03_VC45	HC03_VC38
爷爷奶奶	HC03_VC67	HC03_VC64	HC03_VC65	
教育	HC03_VC91	HC03_VC89	HC03_VC88	
住宅	HC03_VC120	HC03_VC123	HC03_VC122	
出生的地方	HC03_VC133	HC03_VC128		
语言	HC03_VC172	HC03_VC171	HC03_VC173	HC03_VC170
祖先	HC03_VC198	HC03_VC185	HC03_VC194	HC03_VC182
入职年份	HC03_VC146	HC03_VC150	HC03_VC144	

表 2:可能对阿片类药物传播影响最大的因素

在分析数据后，我们得出结论，以下群体可能更倾向于滥用阿片类药物。

ID	人的类型
HC03_VC11	Female_householder_no_husband_present
HC03_VC09	Male_householder_no_wife_present
HC03_VC35	Males_15_years_and_over
HC03_VC45	分离
HC03_VC67	Responsible_for_grandchildren_5_or_more_years
HC03_VC88	Some_college_no_degree
HC03_VC89	副's_degree
HC03_VC172	Language_other_than_Spanish
HC03_VC170	Speak_English_less_than_“very_well”

表 3:群体可能更倾向于滥用阿片类药物。

3.7.2 相关性分析

鉴于几个可能的影响因素，我们还需要确定这些因素是正相关还是负相关，所以我们还需要计算几个过滤后的因素与 2010-2017 年阿片类事件之间的相关性。这里我们使用皮尔逊相关系数，由:

$$\rho_{X,Y} = \frac{\mathrm{E}[XY] - \mathrm{E}[X]\mathrm{E}[Y]}{\sqrt{\mathrm{E}[X^2] - [\mathrm{E}[X]]^2}\sqrt{\mathrm{E}[Y^2] - [\mathrm{E}[Y]]^2}}$$

在考虑县时，如果图是平滑的，我们对观察者做自相关。它表明相关性是否有太多的变化。这里我们假设 0.8 为强相关，计算|corr| >的个数 0.8 来证明正相关或负相关。我们选择三个因素作为影响阿片类药物使用的最重要因素:

有一个或多个 65 岁及以上老人的家庭
研究生或专业学历
家庭住户(家庭)-女性户主，无丈夫在场，家庭

3.7.3 PCA 特征提取

选择了三个特征，但是对于上面介绍的模型来说，对于输入来说还是太大了。因此，我们使用 PCA 将三个特征投影到一条直线上。主成分分析(PCA)首先计算数据矩阵的协方差矩阵，然后得到协方差矩阵的特征值特征向量，选择特征值最大的 k 个特征对应的特征向量组成矩阵。这样就可以将数据矩阵转换为新的空间，实现降维。我们使用 SVD 实现 PCA，将结果化简为 n*1 矩阵并应用于我们的模型

4 策略对抗阿片类药物危机

由上述模型可知，阿片类药物危机的三个关键因素是:(1)无夫女性比例;(2)高学历人群比例;(3) 65 岁及以上人口 1 人及以上的家庭比例。

4.1 因素简要说明

- 因素(1):离婚或伴侣死亡可引发女性药物使用或其他精神健康障碍。女性户主如果得不到同伴的支持，将会承受更大的生活负担和社会压力。此外，女性与男性不同，有一个

阿片类药物成瘾的可能性较高与生理因素有关，如性激素等。

- 因素(2):受教育程度高的人对阿片类药物成瘾的可能性较小。虽然他们也可能面临很大的工作和社会压力，但他们可以调整自己来适应，教育可以让人们意识到阿片类药物的危险，帮助他们预防阿片类药物成瘾。
- 因素(3):老年人更容易出现身体和精神疾病。药物治疗成为缓解症状和控制疼痛的重要方式。同样，长期治疗也可能引发与老年人相同的成瘾，特别是那些失去孩子或缺乏子女照顾的老年人。

4.2 可能的策略

阿片类药物一直被认为是不太有害的药物，廉价和容易获得，为了减少阿片类药物的使用，联邦政府可以做几件事。

4.2.1 老年人

老年人是影响阿片类药物滥用问题的关键因素。阿片类药物被广泛用于减轻疼痛的医疗用途，这类药物尤其适合老年人使用。此外，医生在为患者提供阿片类止痛药方面不受限制，一些老年人经常使用阿片类止痛药并上瘾。政府可以限制医生提供阿片类药物，或者鼓励制药厂生产廉价、有效的止痛药来取代阿片类药物。政府可以组织社区养老院，帮助这些老人找到同伴。共同的兴趣爱好有利于他们的心理和身体健康。

4.2.2 关于教育水平

我们的模型显示，较高的教育水平可能会减少阿片类药物的使用。受教育程度高的人倾向于避免使用毒品，考虑到损害其社会地位的风险。政府可以为公民提供更多上大学的机会，比如提供奖学金或开办社区大学。在社区开展关于药物滥用的教育也很重要。一些关于阿片类知识的宣传标语可以贴在社区的广告牌上，甚至贴在阿片类药物的包装上。

4.2.3 针对无夫女户主

在医学研究中，女性有时会被忽视。有些药物可能对女性有不同的影响，例如，更容易上瘾。除此之外，没有丈夫的女性家庭主妇大多面临更大的压力。她们更容易使用药物来释放压力。政府可以监督制药厂在研发过程中更多地考虑女性。政府还可以为没有丈夫的女性家庭主妇提供工作机会和福利。社区工作人员应该多去看望那些女户主，不仅要关心她们的身体健康，还要关心她们的心理健康。

4.2.4 考虑其他因素

- 不同州的法律不同，从药物传播图来看，大多数阿片类药物是从俄亥俄州和弗吉尼亚州传播的，这两个州的法律对药物更宽容。应该制定更严格的法律来限制阿片类药物的生产和贸易。
- 从药物传播图来看，阿片类药物在交通发达的地区更容易传播，例如靠近海岸的地区。运输中的库存可以以更严格的方式进行检查。

4.3 策略验证

以上策略都是为了降低阿片类药物对这些人群的影响。在我们的模型中，我们认为特定群体受到阿片类药物影响的因素是固定的。验证我们策略的等效方法是降低相应群体的百分比。在现实中，我们实际上不能“降低”其百分比，但这里的“降低”与降低特定群体的影响因子具有相同的目的。

4.3.1 等效验证的验证

假设一共有 k 种组。让{α1,α2,……,αk}表示各群体占总人口的比例。让{β1,β2,……,βk}表示第 i 组阿片类药物成瘾者占该组总人口的比例。设 S 为总人口数。那么原始的总成瘾人数占，

$$OriginTotal = S \times (\alpha 1\beta 1 + \alpha 2\beta 2 + \cdots + \alpha k\beta k) = S\ \alpha i\sum_{i=1}^{k}$$

现在我们用策略来降低 β_i，然而，β_i 在我们的模型中不是特定的，因为我们无法根据现有的信息计算出来。为了验证我们的策略，一个合理的方法是降低 α_i 达到同样的目的。假设我们的策略降低了 β_j 对 β_j^0。现在我们试着算出新的 α_j^0 基于我们的等价思想。策略实施后的总成瘾人数为，

$$\begin{aligned} NewTotal &= S \times (\alpha_1\beta_1 + \alpha_2\beta_2 + ... + \alpha_j\beta_j' + ... + \alpha_k\beta_k) \\ &= S\sum_{i=1,i\neq j}^{k}\alpha_i\beta_j + S\alpha_j\beta_j' \end{aligned} \tag{1}$$

然后我们固定 β_j，集合 α_j 为 α_j^0 并保持其他参数不变。结果，总上瘾人数减少了 $S(\alpha_j - \alpha_j^0)$，用 S’表示。新的总成瘾人数可计算为:

$$\begin{aligned} NewTotal &= S' \times (\alpha_1\beta_1 + \alpha_2\beta_2 + ... + \alpha_j'\beta_j + ... + \alpha_k\beta_k) \\ &= S'\sum_{i=1,i\neq j}^{k}\alpha_i\beta_j + S'\alpha_j'\beta_j \end{aligned} \tag{2}$$

根据上面的式(1)和式(2)，为简单起见，我们假设 $S \approx S^0$ 并且为了使假设合理，我们的模型无法验证何时($\alpha = \alpha_j - \alpha_j^0$ 很大。暂时，我们使假设满足，那么

$$S\sum_{i=1,i\neq j}^{k}\alpha_i\beta_j + S\alpha_j\beta_j' = S'\sum_{i=1,i\neq j}^{k}\alpha_i\beta_j + S'\alpha_j'\beta_j$$

于是得到，αi0 = αj ββ0jj。在实践中，如果一种策略可以将一个因子 β 降低 20%，那么该策略就被认为是非常成功的(所以假设 β 将降低 k 个百分点是合理和实际的，其中 k 小于 20)。由于主要因素的 α 分别为 12%、8%和 27%。我们可以改变 α_i 相应的，最多变动 2.4%，1.6%和 4 个$ %。因此，我们大致可以说 $S \approx S^0$ 因为变化是可以接受的小。

4.3.2 验证结果

验证结果如下表所示，

图 19:策略 A 测试区域 88

因素 水平	女户主	高等学历人士	年龄≥65 岁的户主	总递减率
起源水平	12%	8.0%	27%	0%
调整水平	12%	8.0%	26%	0.34%
调整水平	12%	8.0%	25%	1.33%
调整水平	12%	8.0%	24%	2.57%

图 20:策略 B 对区域 88 进行测试

因素 水平	女户主	高等学历人士	年龄≥65 岁的户主	药品报告总数
起源水平	12%	8.0%	27%	0%
调整水平	11.2%	8.0%	27%	0.43%
调整水平	10.4%	8.0%	27%	1.17%
调整水平	9.6%	8.0%	27%	2.11%

图 21:策略 C 对 88 区进行测试

因素 水平	女户主	高等学历人士	年龄≥65 岁的户主	药品报告总数
起源水平	12%	8.0%	27%	0%
调整水平	12%	8.5%	27%	0.15%
调整水平	12%	9.0%	27%	0.34%
调整水平	12%	9.5%	27%	0.71%

注:88 区由 CAMPBELL、BUTLER、CALDWELL、KENTON 组成。4 个县均位于肯塔基州。

4.3.3 发现

上述结果表明，应特别关注无夫女性户主和 65 岁及以上高龄户主。这两个特殊群体是本次阿片类药物危机的主要组成部分，针对这两个群体的策略应予以认真考虑。教育在这场阿片类危机中也起到了至关重要的作用，提高人们的整体教育水平可以帮助美国度过这场阿片类危机。

5 模型分析

虽然我们认为我们的模型比大多数现有模型要好，但它仍然有一些需要注意的弱点。

- 在考虑图形位置影响时，我们根据生活经验将影响半径设置为 200km，不够具体。然而，我们还没有找到一种有效的方法来计算出哪个半径最合适。
- 当考虑到时间的影响时，我们的尝试是让最新的趋势比以前的有更大的影响。然而，确定时间衰减因子是我们在这么短的时间内没有解决的另一个难题。
- 根据我们的相关分析，这次阿片类危机背后有很多影响因素。然而，我们的回归模型对于高维数据来说不够准确。在实践中，我们考虑了三个主要的影响因素来训练我们的模型同时进行预测。如果可能的话，我们应该同时调用更多的因素进行分析。

参考文献

[1] https://www.drugabuse.gov/publications/drugfacts/substance-use-in-women

[2] Bernhard SchA”ulkopf, Williamson, R. C.， Smola, A. J.， shaw - taylor, J.， & Platt, J. C.。(1999)。新颖性检测的支持向量法。神经信息处理系统的进展 12，[NIPS 会议，美国科罗拉多州丹佛，1999 年 11 月 29 日- 12 月 4 日]。麻省理工学院出版社。

[3]勒夫。Suresh1, T. R.(2013)。使用 apriori 算法挖掘频繁项集。国际计算机趋势与技术杂志，4(4)。

[4]使用 Scikit-Learn 和 tensorflow 进行动手机器学习

[5] Fouss, F.， Pirotte, A.， render, J. M.， & Saerens, M.。(2007)。图节点间相似度的随机游走计算，并应用于协同推荐。IEEE 知识与数据工程汇刊，19(3)，355-369。

[6] Recht, B.;(2009)。一种更简单的矩阵补全方法。Journal of Machine Learning Research。

[7] Castroneto, M.， Jeong, Y. S.， Jeong, M. K.， & Han, L. D.。(2009)。典型和非典型交通条件下短期交通流预测的在线 svr。专家系统与应用，国际期刊，36(3)，6164-6173。

[8] Lin, m.y.， Lee, P. Y.， & Hsueh, S. C.。(2012)。MapReduce 上基于 apriori 的频繁项集挖掘算法。泛在信息管理与通信国际会议。ACM。

6 备忘录

6.1 阿片类药物滥用的起源

从阿片类药物滥用的起源我们可以分析很多有趣的现象。大多数阿片类药物的来源集中在俄亥俄州和弗吉尼亚州。造成这种分布特征的因素可能有很多。我们主要分析交通特征和法律特征。

交通因素:在海洋附近，或者五大湖附近，我们可以看到很多阿片类药物的来源。由于靠近海洋，更容易从国外进口非法毒品或阿片类药物，使得这些地区成为大量阿片类药物的来源。

法律因素:我们可以看到，俄亥俄州是许多毒品的来源。出现这种情况的原因，可能与当地法律法规还存在漏洞，政府的监管力度不强有关。

6.2 阿片类药物滥用分布预测

根据我们的预测，阿片类药物滥用在部分地区有所下降，但总体呈上升趋势。在已经面临阿片类药物危机的地区和相邻地区，阿片类药物滥用问题越来越严重。俄亥俄州的大部分地区、肯塔基州的东海岸以及弗吉尼亚和宾夕法尼亚州的西海岸都面临着阿片类药物危机的威胁。

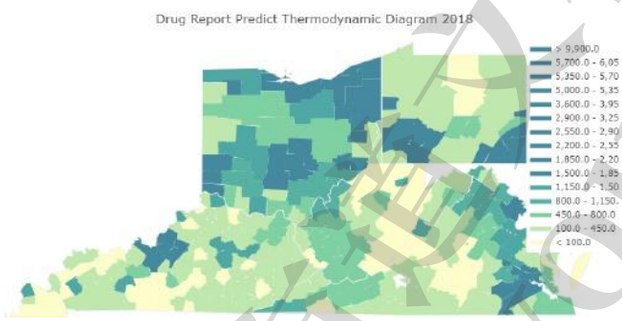


图 22:2018 年阿片类药物预测报告

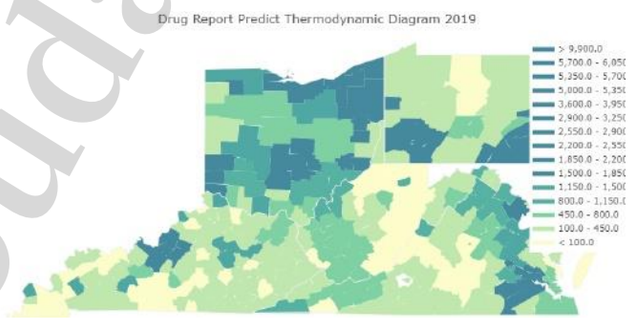


图 23:2019 年阿片类药物预测报告

6.3 相关因素

从我们的观察来看，老年人、无夫女性家庭与阿片类药物滥用的增加有关。教育在阿片类药物危机中也起着至关重要的作用。

ID	人的类型
HC03_VC11	Female_householder_no_husband_present
HC03_VC09	Male_householder_no_wife_present
HC03_VC35	Males_15_years_and_over
HC03_VC45	分离
HC03_VC67	Responsible_for_grandchildren_5_or_more_years
HC03_VC88	Some_college_no_degree
HC03_VC89	副's_degree
HC03_VC172	Language_other_than_Spanish

HC03_VC170	Speak_English_less_than_“very_well”

表 4:群体可能更倾向于滥用阿片类药物。

6.4 战略

6.4.1 针对老年人

政府可以限制医生提供阿片类药物，或鼓励制药厂生产廉价、有效的止痛药来取代阿片类药物，使老年人不太可能获得阿片类药物并使用替代药物来治疗疾病。政府可以组织社区养老院，帮助这些老人找到同伴。共同的兴趣爱好有利于他们的心理和身体健康。

6.4.2 关于教育水平

政府可以为公民提供更多上大学的机会，如提供奖学金或开办社区大学。在社区开展关于药物滥用的教育也很重要。一些关于阿片类知识的宣传标语可以贴在社区的广告牌上，甚至贴在阿片类药物的包装上。

6.4.3 针对无夫女户主

政府可以监督制药厂在研发过程中更多地考虑女性。政府还可以为没有丈夫的女性家庭主妇提供工作机会和福利。社区工作人员应该多去看望那些女户主，不仅要关心她们的身体健康，还要关心她们的心理健康。

6.4.4 考虑其他因素

- 各州的法律各不相同，从毒品传播图来看，大多数阿片类药物是从俄亥俄州和弗吉尼亚州传播的，这两个州的法律对毒品更宽容。应该制定更严格的法律来限制阿片类药物的生产和贸易。

- 从药物传播图来看，阿片类药物在交通发达的地区更容易传播，例如靠近海岸的地区。运输中的库存可以以更严格的方式进行检查。

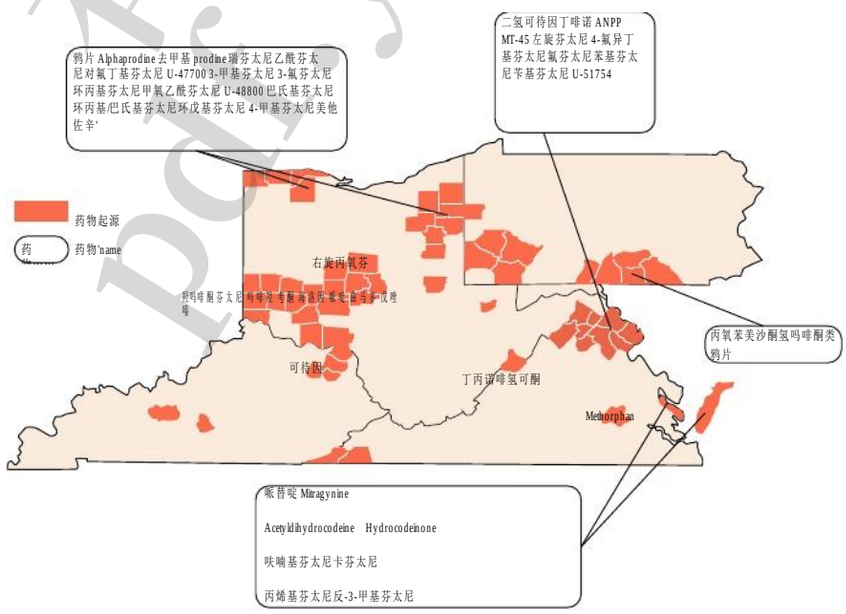


图 24:所有阿片类药物的来源

附录

以下是我们在模型中使用的程序。**数据预处理，Kmeans 算法**

```
进口 xlrd
进口熊猫作为 pd
将 numpy 导入为 np
从数学导入弧度，cos, sin, asin, SQRT 导入
XLWT
进口经营者
进口复制
从 functools 导入 reduce

#初始化
def init(表 ):
    Nrows = table.nrows
    Ncols = table.ncols
    #使用 FIPS_combined 作为键记录 info
    county = {}
    对于范围(1,nrows)中的 I:
        FIPS_Combined= table.row_values(i)[6]
        如果 FIPS_Combined 不在 county.keys():
        lat = table.row_values(i)[12] lon =
        table.row_values(i)[13]
        county[FIPS_Combined]= (lon, lat)
    返回县

Def haversine(pos1, pos2):
    ”
    “ ”
    计算两点之间的大圆距离
    在地球上(以十进制度数指定)
    ”
    “ ”
    #将十进制转换为圆弧制
    Lon1, lat1 = pos1 .
    Lon2, lat2 = pos2
    Lon1, lat1, lon2, lat2 = map(弧度， [Lon1, lat1, lon2, lat2])

    # haversine 公式
    Dlon = lon2 - lon1
    dat = lat2 - lat1
    A = sin(dat /2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2 c
    = 2 * asin(根号(A))
    R = 6371 #地球半径， 单位千米
    返回 c * r * 1000

Def renew_cluster(centers, pos, cities):
    Label = dict().fromkeys([i for i in range(len(centers))], [])
    for city in cities:
        Dist = []
        For center in centers:
            dist.append(半正矢(pos(城市),中心))
        Index = dist. Index (min(dist))
        Cluster = copy.deepcopy(label[index])
        cluster.append(市)
        Label [index] =
        cluster
```

```
    New_centers  = []


    对于 range(len(centers))中 的
    I:

        位置= list(map(lambda x: pos[x],   label[i]))

        new_centers.append (np。Mean (positions,
    axis=0))返回 new_centers, label  . append


#将位置转换为 k 个簇 def
k_means(k, pos):

    Cities  = list(pos.keys())
    if k > len(Cities):
```

```
        print(“ K too large” )
        返回 False

    其他：

        # initial centers

        Old_centers = [list(pos[cities[i]]) for i in range(k)]

        New_centers = []

        # itartion 直到收敛

        而真正的：

            New_centers, cluster = renew_cluster(old_centers, pos, cities) #
            print(New_centers)

            打破；

            If (np.array(new_centers) == np.array(old_centers)).all():

                打破

            其他：

                Old_centers = new_centers


        Zone = {}

        对于集群中的键：

            对于集群中的城市[key]:
                zone[city] = key

        返回 new_centers, duster, zone


Def 输出(centers, cluster, zone, data):

    # new_file = xlwt.;Workbook(encoding = ' utf-8 ') #
    new_table = file.add_sheet(' data ')


    Table = []

    对于 range(1,data.nrows)中的
    I:

        YYYY = data.row_values(i)[1]

        FIPS_Combined = data.row_values(i)[6]

        物质名= data.row_values(i)[7]

        药物报告端口=
        data.row_values(i)[8]

        总药物报告端口= data.row_values(i)[9]

        label = zone[FIPS_Combined]

        entry =[YYYY, label, 物质名, 药物报告端口, 总药物报告端口]


    # def f(x, y):

    #如果(np.array (x [3]) = = np.array (y[3]))所有 (): #
    返回 x [3] + [3] [x + y [3]] + [4] [x + y[4]] #减少
    (λx, y: f表)

    返回表



如果__name__ = ' __main__ ':

    Data = xlrd.open_workbook(' excel_output_add.xls ').sheets()[0]
    county = init(Data)

    Centers, cluster, zone = k_means(100, county)

    Table = output(centers, cluster, zone, data)

    #将数据写入 excel

    excel = xlwt.Workbook()

    Sheet1 = excel。add_sheet(“基于区域的日期”)

    Sheet2 = excel。add_sheet(“ZoneCounty Correspondence
    Table”)sheet3 = excel。add_sheet(“县”)


    sheet1。写(0,0, "YYYY")

    sheet1。write(0,1,
     “Zone”)
```

```
sheet1。 写入(0,2， "物质名称")

sheet1。 write(0,3，
"DrugReportZone")
```

sheet1。写(0,4， "TotalDrugReportZone")
为行，条目在枚举(表):

```
    对于 col in range(len(entry)):

        sheet1。 写入(row+1, col, entry[col])
```

sheet2。写(0,0， "Zone") sheet2。
write(0,1， "Latitude") sheet2。
写(0,2， “经度”)

```
对于行，键在 enumerate(list(cluster.keys())):
    sheet2。 写入(row+ 1,0, row)
    sheet2。 写入(row+ 1,1, centers[row][1])
    sheet2。 写入(row+ 1,2, centers[row][0])
    对于 col in range(len(cluster[key])):
        sheet2。 写入(row+1, col+3, cluster[key][col])

sheet3。 写(0,0， “县”)
sheet3。 write(0,1,
"FIS_Combined")
sheet3。 写(0,2， “纬度”)
sheet3。 写(0,3， “经度”)
Written = []
Row_num =
0.
对于 range(1, data.nrows)中 的
row:
    Entry = data.row_values(row)
    如果条目[6]没有写入:
        Row_num += 1
        written.append[6](条目)
        Info = [entry[6], entry[2]+ ' + ' +entry[3], entry[12],
        entry[13]] for col in range(4):
            sheet3。 Write (row_num, col,
            info[col])

excel.save( “/用户/ lsd /桌面/代码/ k - means /
kmeans.xls” )
```

寻找阿片类药物的来源

```
进口 xlrd
进口熊猫作为 pd
将 numpy 导入为 np
从 math import *
进口 xlwt
进口经营者
进口复制
从 functools 导入减少导入
matplotlib。 Pyplot 作为 PLT 导
入 matplotlib
导入 networkx 作为 nx
从 itertools 导入组合从 igraph import
*
将 seaborn 导入为 SNS
进口 pylab
进口随机
sns.set_style( “darkgrid” )

def adjMatrix(medical_name, time, distance_threshold):
    G=nx.DiGraph()
    H=nx.DiGraph()
    M=nx.DiGraph()
    Excel_path = ' kmeans.xls '
    Klocation = pd.;read_excel(excel_path, sheet_name= ' ZoneCounty 对应表')kdata = pd。
    read_excel(excel_path, sheet_name= '基于 Zone 的日期')
    Distance = pd.read_excel(' Distance . excel ')xls, sheet_name = “距离”).
    values
    Simi = pd.read_excel('相似度。 Xls ', sheet_name = ' sim ').values

    Edges = []
    medical_data = kdata[(kdata[' substanceName '] == medical_name) & (kdata[' YYYY '] == time)]
    combins = [list(c) for c in combination (set(medical_data[' Zone '].values.tolist()), 2)]
```

```
Weights = []
对于 range(len(combins))中 的
I:
    X = combins[i][0]
    Y = combins[i][1]
    如果 distance[x, y] < distance_threshold:
        #print(medical_data[medical_data['Zone'] == x])
        #print(medical_data[medical_data['Zone'] == y])
        if (sum(medical_data[medical_data['Zone'] == x]['DrugReportZone'].values) >=
            sum(medical_data[medical_data['Zone'] == y]['DrugReportZone'].values):
            edges.append
            (combins[我])
            G。Add_edges_from ([combins[i]], weight = round(simi[combins[i][0],
            combins[i][1]], 1)) weights.append(round(simi[combins[i][0], combins[i][1]], 1))
        其他：
```

```
                combins[我].reverse
                ()

                edges.append
                (combins[我])

                #打印(combins[我].reverse
                ())

                G。Add_edges_from ([combins[i]]， weight = round(simi[combins[i][0]，
                combins[i][1]]， 1))权重。追加(圆(思米[combins[我][0],combins[我][1]],1))
```

打印(边缘)

```
度=[[x,0] for x in range(100)] for I
in range(len(edges)):
    Degree [edges[i][0]][1] += 1

学位。sort(key=lambda x: x[1]， reverse = True)
ra = random.randint(0,3)
Origin = degree[ra][0] .
```

打印(程度)

```
#画图
# G.add_edges_from(边缘)
Ind = []
Posi = []
```

对于范围(100)中的 I:

```
    ind.append(
    我)
    posi.append (klocation(“经度”)。iloc[我],klocation(“纬度”).iloc[我]))
Position = dict(zip(ind, posi))

Vals = []
```

对于范围(100)中的 I:

```
    瓦尔斯。追加(int(sum(medical_data[medical_data[' Zone '] == i]['DrugReportZone'].values))
values_dic = dict(zip(ind, vals))
Values = [values_dic. value]get(node, 1.0) for G.nodes()中
的 node]
```

对于 range(len(values))中的 I:

```
    Values [i] = Values [i] -
    10
    If (values[i] > 210):
        Values [i] = 210
edge_labels = dict ((((u, v), d['减肥'])
                for u,v,d in G.edges(data=True)])
plt.figure ()
nx。draw_networkx_edge_labels(G,pos = position,edge_labels=edge_labels)
# nx。draw_networkx_edges(G, pos = position, width = 2, alpha = 0.5, arrows = True, arrowstyle= ' -> ', e
nx.;draw(G, pos = position, node_color = "skyblue", node_size = 2000, edge_cmp = plt.cm.Blues,
with_labels =
True)
FIG = matplotlib.pyplot.gcf()
fig.set_size_inches(40
岁,20)
plt.savefig(”。/FIG/"+ str(medical_name) + '_' + str(time) + "_" + "0.pdf") #另存为 png 格式
#flow_value = nx.;shortest_path(G, 41,32)
#pos=nx.spring_layout(G) #所有节点的位置
# nx。draw_networkx_nodes(G,pos = position,cmap=plt.get_cmap(' jet '), node_size=500,
node_color=values #nx。draw_networkx_edge_labels(G,pos = position,edge_labels=edge_labels)
plt.figure ()
# nx。draw_networkx_nodes(G,pos = position,cmap=plt.get_cmap(' jet '), node_size=1000, node_shape="o"
#nx.;draw_networkx_edges(G, pos = position, width = 2, alpha = 0.5, arrows = True, arrowstyle= ' -> ', e
#nx.;draw_networkx_edge_labels(G,pos = position,edge_labels=edge_labels)
nx。draw(G, pos =位置， node_color =值， node_size = 2000, edge_cmp =
 plt.cm。蓝色， edge_color=weights, with_labels=True)
# plt.axis(“了”)
```

```
# plt。 传奇(loc =右上角)

FIG = matplotlib.pyplot.gcf()

fig.set_size_inches(40
岁,20)

plt.savefig(”。/FIG/"+ str(medical_name) + '_' + str(time) + "_" + "1.pdf") #另存为 png 格式

#plt.show() #显示

# pylab.savefig(“weighted_graph。 Png”, dpi = 500) #保存为
Png

#graphStyle= {' vertex_size ': 20}

# g = Graph(edges = edges, directed = True)

# g.write_svg(“的例子。 svg", width=3000, height =3000,
**graphStyle)


plt.figure ()

plt.axis(“了”)

New_edges, edges_colors = get_new_edges(edges, origin)

对于 range(len(new_edges))中 的
I:

    H。 add_edges_from ([new_edges[我]],体重=圆(思米[new_edges[我][0],new_edges[我][1]]))
```

```
    # real_color = []
    # elarge = [(u, v) for (u, v, d) in G.edges(data=True)]
    # for I in range(len(elarge)):
    # for k in range(len(edges_colors)):
    # #打印(edges_colors[k][0], elarge[i])
    # if ((edges_colors[k][0][0] == elarge[i][0]) and (edges_colors[k][0][1] == elarge[i][1])): #
real_color.append(edges_colors[k][1])
    # elif(k == (len(edges_colors) - 1)):
    #         real_color.append(“mediumslateblue”)
    #打印(len(elarge), len(real_color))
    nx。draw_networkx_nodes(H,pos = position,cmap=plt.get_cmap('jet'), node_size=2000, node_shape="o", a
node_color = 'skyblue')
    nx。draw_networkx_edges(H, pos = position, width = 1, alpha = 0.5, arrowstyle='->')
    plt。Scatter(position[origin][0], position[origin][1], color='royalblue', marker='o',
edgecolors='g'plt。Text(position[origin][0], position[origin][1], str(origin), fontsize=20,
style='斜', ha='c plt.xlabel('lng')
    plt.ylabel(lat)
    # nx。绘制(H, pos = position, node_color = 'skyblue', node_size = 1000, edge_cmp = plt.cm.Reds)
    FIG = matplotlib.pyplot.gcf()
    fig.set_size_inches(40
岁,20)
    pylab.savefig(”。+ str(medical_name /图”)+’_’ + str(时间)+ "_" + "2. pdf”)#保存为
png


Def get_new_edges(edges, origin):
    New_edges = []
    Color_mat = []
    对于 range(len(edges))中 的
    I:
        If (edges[i][0] == origin):
            new_edges.append(边缘[我])
            边缘 color_mat.append([[我],“鲑鱼”))
    S_edges = new_edges
    对于 range(len(new_edges))中 的
    I:
        Tem = new_edges[i][1]
        对于 k在 range(len(edges)):
            If (edges[k][0] == tem):
                s_edges.append(边缘[k])
                Color_mat.append([edges[k], 'navajowhite'])
t_edges = s_edges
    打印(s_edges)
    对于 range(len(s_edges))中 的
    I:
        Tem1 = s_edges[i][1]
        对于 k在 range(len(edges)):
            If (edges[k][0] == tem1):
                t_edges.append(边缘[k])
                color_mat。Append([edges[k], 'mediumslateblue
']) return t_edges, color_mat



Def get_all_origin(time, distance_threshold):

    Excel_path = 'kmeans.xls'
    Klocation = pd.;read_excel(excel_path, sheet_name='ZoneCounty 对应表')kdata = pd。
read_excel(excel_path, sheet_name='基于 Zone 的日期')
    Distance = pd.read_excel('Distance .excel')xls, sheet_name =“距离”).
values
    Simi = pd.read_excel('相似度。Xls', sheet_name ='sim').values


    Medical = []
    def opoid_extract(x):
```

```
        If not (x in medical):
            medical.append(x)

kdata['物质名'].apply(opoid_extract)


All_origins = []

对于范围内的 kk (len(medical)):
    medical_name = medical[kk]

    Edges = []

    medical_data = kdata[(kdata['物质名']== medical_name) & (kdata[' YYYY '] ==时间)]combinins
    = [list(c) for c in combination (set(medical_data[' Zone '].values.tolist())， 2)] weights = []

    对于 range(len(combins))中 的
    I:
```

```
            X = combins[i][0].
            Y = combins[i][1]
            如果 distance[x, y] < distance_threshold:
                #print(medical_data[medical_data[' Zone '] == x])
                #print(medical_data[medical_data[' Zone '] == y])
                if (sum(medical_data[medical_data[' Zone '] == x]['DrugReportZone'].values) >=
                    sum(medical_data[medical_data[' Zone '] == y]['DrugReportZone'].values):
                    edges.append
                    (combins[我])
                    # G。 Add_edges_from ([combins[i]], weight = round(simi[combins[i][0],
                    combins[i][1]], 1) #weights.append(round(simi[combins[i][0], combins[i][1]], 1))
                其他：
                    combins[我].reverse
                    ()
                    edges.append
                    (combins[我])
                    #打印(combins[我].reverse
                    ())
                    # G。 Add_edges_from ([combins[i]]], weight = round(simi[combins[i][0],
                    combins[i][1]], 1 #权重。追加(圆(思米[combins[我][0],combins[我][1]],1))
        Degree = [[x,0] for x in range(100)]
        for I in range(len(edges)):
            Degree [edges[i][0]][1] +=
            1
        学位。sort(key=lambda x: x[1], reverse = True)
        ra = random.randint(0,3)
        Origin = degree[ra][0].
        all_origins。追加([医疗[kk], 原点])打
        印([医疗[kk], 原点])
    打印(all_origins)

如果__name__ == '__main__':
    Data = xlrd.open_workbook(' kmeans.xls ')
    Drug2num, num2drug, dataslice, center = init(data)
    训练矩阵= np.zeros((100,69,7), dtype = np.int)
    #打印(len (num2drug))
    适用于范围(7)中的年
    份：
        tmp =稀疏矩阵(dataslice[year], drug2num, num2drug)
            对于 range(len(num2drug))中的
            j:
                训练矩阵[i][j][year] += tmp[i][j]
    held_outMatrix =稀疏矩阵(dataslice[7], drug2num, num2drug) #
    print(训练矩阵)

    #打印(相似度(2, 训练矩阵, 中心, 300000))
    # get_all_origin(2010、
    200000)
```

数据预测

```
进口 xlrd
进口熊猫作为 pd
将 numpy 导入为 np
从 math import *
进口 xlwt
进口经营者
进口复制
从 functools 导入 reduce
从 sklearn 导入 SVM
从 sklearn.decomposition 导入 PCA
```

```
Def haversine(pos1, pos2):
    ”
    “ ”
    计算两点之间的大圆距离
    在地球上(以十进制度数指定)
    ”
    “ ”
    #将十进制转换为圆弧制
    Lon1, lat1 = pos1 .
    Lon2, lat2 = pos2
    Lon1, lat1, lon2, lat2 = map(弧度, [Lon1, lat1, lon2, lat2])


    # haversine 公式
    dlon = lon2 - lon1
    dat = lat2 - lat1
```

```
    A = sin(dat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2 c
    = 2 * asin(根号(A))

    R = 6371 #地球半径， 单位千米

    返回 c*r*1000


def init(数据):
    Sheet1 = data.sheets()[0]                    #基于 Zone 的日期
    sheet2 = data.sheets()[1]                    # ZoneCounty 对应表#县位置
    sheet3 = data.sheets()[2]



    #映射物质名到药品号 drug2num = {}

    Num2drug = []


    Rank = 0

    dataslice = [[], [], [], [], [], [], [], []]
    sheet1.nrows 的行范围(1):

        Entry = sheet1.row_values(row)

        YYYY=条目[0]

        物质名称=条目[2]

        如果物质名不在 drug2num.keys()
            中 :drug2num[物质名]= rank
            num2drug.append(物质名)rank += 1


        如果 YYYY 年== 2010 年， 则按年将数据
        分成 8 部分:

            dataslice[0].append(入口)

        如果 YYYY== 2011:

            数据集[1].append(条目)如果
        YYYY== 2012:

            数据集[2].append(条目)如果
        YYYY== 2013:

            数据集[3].append(条目)如果
        YYYY== 2014:

            数据集[4].append(条目)如果
        YYYY== 2015:

            数据集[5].append(条目)如果
        YYYY== 2016:

            数据集[6].append(条目)如果
        YYYY== 2017:

            dataslice[7].append(入口)


    # dict 存储每个区域的位置中心= {}

    对于 range(1, sheet2.nrows)中的 row:
        entry = sheet2.row_values(row)
        zone = int(entry[0])

        如果 zone 不在 center.keys():
            lat = entry[1]

            Lon =入口[2]

            中心[zone] = (lon, lat)

    返回 drug2num, num2drug, dataslice, center


def init_feature(特征):

    Sheet = feature.sheet_by_name(u'extract_feature')
    factor = [[], [], []]

    对于 range(1, sheet.nrows)中的
    row:

        Entry = sheet.row_values(row)

        如果条目[1]== 2010:

            因素[0].append(条目[3:])
```

```
    如果条目[1]== 2011:
        因子[1].append(条目[3:])如
    果条目[1]== 2012:
        Factor [2].append(entry[3:])
返回因子
```

```
def 稀疏矩阵(data, drug2num, num2drug):
    "'
    矩阵有 100 个区域，每个区域形成一排
    每一行表示该区域内每种药物的药物报告的 num，我们将其全部设为 0
    "'
    # len(num2drug) = 69
    稀疏矩阵= np。0((100、69))
    对于 row in range(len(data)):
        Zone = int(data[row][1])
        drugName= data[row][2]
        drugNum= drug2num[drugName]
        drugReport= data[row][3]
        稀疏矩阵[区域][drugNum]+=drugReport

    返回稀疏矩阵

"'
    找到任何区域之间的相似性 alpha 是距离系数我们只关心距离在 alpha 以内的对
"'
Def 相似性(zone, data, center, alpha):
    邻接区= []
    对于范围(100)中的 I:
        如果 I!= zone 且 haversine(center[I], center[zone]) < alpha:
            Sum = np.sum(np.sum(np.array(data[i])))+np.sum(np.sum(np.array(data[zone])))
            corr = 0
            对于 range(69)中的 j:
                Tendency_i = data[i][j]
                Tendency_zone = data[zone][j]
                Sum_i = np.sum(np.array(tendency_i))
                Sum_zone = np.sum(np.array(tendency_zone))
                如果 sum_zone == 0 或 sum_i == 0:
                    继续
                Por = sum_i + sum_zone
                Coef = np.;Corrcoef(tendency_i, tendency_zone)[0,1]
                如果 isnan(coef)或 coef == 0:
                    通过
                其他：
                    corr += coef*1.0*por/Sum
            邻接区。追加((我 corr))
    如果 len(adjacentZones) > 10:
        邻接区。Sort (key=lambda x: x[1])
        返回列表(map(lambda x: x[0], 邻接区[-10:]))
        返回列表(map(lambda x: x[0], 邻接区))

def 正常化(数组):
    Sum = np.sum(np.array(array))
    返回 list(map(lambda x: x*1.0/Sum, array))
```

“ ‘

目的是预测未来几年的药物传播趋势

注意:

(1)对于每个区域和它的每一种药物，如果药物报告太小，我们不能预测它的趋势，我们用来自它相似区域的趋势来预测它

(2)如果当前区域的特定药物有足够的样本，那么我们将其趋势与其相似区域的趋势结合起来进行预测

“ ‘

```
数据， 中心， held_outMatrix:
    Features = {}
    轴= [[0], [1], [2], [3], [4],
    [5], [6]]
    对于范围(100)中的 I:
        Simi_zone =相似度(i, data, center, 200000)
        N = len(simi_zone)
        对于 range(69)中的
        j:
            #找到每个样本的趋势模型
            Feature = []
            CLF = svm。SVR(gamma= 'auto', degree = 3, kernel= '
            poly')适合(轴、数据[我][j])
            feature.append (clf)
            对于范围(n)中的 k:
                CLF = svm。SVR(gamma= 'auto', degree = 3, kernel= '
                poly')适合(轴、数据(simi_zone [k]] [j])
                feature.append (clf)
            #使用 held_out 数据查找每个模型的权重 weight = []
            对于特征中的 f:
                Pre = f.predict([[7]])
                real =held_outMatrix[j]
                weight.append (1.0 / (0.01 + np.square (pre-
                real)))
            Weight =归一化(权重)特征[(i, j)]
            =[特征， 权重]
    返回功能


def find_SVR_coef_Mul(数据， 中心， held_outMatrix， 因
    子):features = {}
    对于 range(100)中的
    I:
        轴= []
        Simi_zone = similarity(i, data, center, 200000) n
        = len(Simi_zone)
        Feature = []
        对于 range(3)中的
        j:
            # pca = pca (n_components = 1)
            # newaxis = pca.fit_transform(因素
            [j][我])
            # axis.append ([newaxis])
            axis.append(因素[j][我])
        CLF = svm。SVR(gamma= 'auto', kernel= 'poly
        ')
        #打印(轴)
        #打印([np。Sum (list(map(lambda x: x[0], data[i]))), np.;Sum (list(map(lambda x: x[1],
        data[i])))]) clf。(轴,[np。Sum (list(map(lambda x: x[0], data[i]))),
            np。Sum (list(map(lambda x: x[1], data[i]))), np.;Sum (list(map(lambda x: x[2],
            data[i])))])
        feature.append (clf)
        对于范围(n)中的 k:
            轴= []
            对于范围(3)中的 t:
                axis.append(因素[t] [simi_zone [k]])
            CLF = svm.;SVR(gamma='auto', kernel='poly')
            clf。(轴,[np。sum(列表(地图(λ x: x[0],数据(simi_zone [k]))),
                np。sum(列表(地图(λ x: x[1],数据(simi_zone [k]))), np。Sum (list(map(lambda x: x[2], da
            feature.append(clf))
        Weight = []
```

```
            对于特征中的f:
                Pre = f.predict([factor[2][i]])
                real =held_outMatrix[i]
                weight.append (1.0 / (1 + abs (pre-
                real)))
            Weight = normalization(Weight)
            特征[i] =[特征， 权重]
        返回功能


Def predict(features, year): t
    = year - 2010
    Predict = np。范围(100)中 I
    的零([100,69]):
        对于范围(69)中的 j:
```

```
            F, w = features[(i, j)]

            Pre = round(np.;sum(列表(地图(λ x, y: x.predict([[t]]) * y, f, w))),如果前>
            0 0):

                Predict [i][j] =
                pre .

            其他：

                预测[i][j] = 0

    回归预测



def predict_Mul(特性， new_factor):

    Predict = np.;0([1] 100 年)

    对于范围(100)中的 I:

        New_f = new_factor[i]

        F, w = feature [i]

        Pre = round(np.;sum(列表(地图(λ x, y: x.predict ([new_f]) * y, f, w))),如果前>
        0 0):

            Predict [i] =
            pre .

        其他：

            Predict [i] = 0
    返回 Predict



如果__name__ == '__main__':

    data = xlrd.open_workbook(' ../K-means/kmeans.xls ')
    drug2num, num2drug, dataslice, center = init(data)


    训练矩阵= np.zeros((100,69,7)， dtype = np.int)


    适用于范围(7)中的年
    份：

        tmp =稀疏矩阵(dataslice[year]， drug2num, num2drug)

            对于 range(len(num2drug))中的
            j:

                训练矩阵[i][j][year] += tmp[i][j]

    held_outMatrix=稀疏矩阵(dataslice[7]， drug2num, num2drug)



    Extract_feature = xlrd.open_workbook(' Extract_feature .xls ')
    factor = init_feature(Extract_feature)


    #打印 (’ ---------- 真正的 ----------
    -’ ) 真正的= np.zeros([1] 100 年)

    对于 range(len(dataslice[2]))中的
    I:

        Entry = dataslice[2][i]

        Real [int(entry[1])] +=
        entry[3] .

    #打印(真正
    的)

    #打印([列表(地图(λ x: x [2], dataslice[2][我]))我的范围(100)])#打印
    (’ ---------- 前 ----------’ )

    训练矩阵， 中心， 实数， 因子

    pre =predict_Mul(训练， 因子[2])

    #打印(前)



    Factor_new = [[12.],
    5,27]]*100 .

    pre_new =predict_Mul(train, factor_new)


    # for I in range(100):
```

```
#    打印(“真实:{0}——pre:{1}——新:{2}——区:{3}”。Format (real[i],  pre[i],
```

打印(“真实:{0}——pre:{1}——新:{2}——区:{3}”。格式(real[88], pre[88], pre_new[88], 88 # feature = find_SVR_coef(训练矩阵，中心，held_outMatrix)

```
#打印(功能)
#打印 (’ ---------- 2017 年预测 ----------
’ )
# pred2017 = predict(features, 2017)
#打印(pred2017)
#打印(held_outMatrix)
#打印(训练矩阵[0][1])
# sim =相似度(0， 训练矩阵， 中心， 300000)
#打印(sim)
# for s in sim:
```

```
#     print(训练矩阵[s][1],    '————')
#打印('----------  2017 年实际  ----------
----')
# real2017 =稀疏矩阵(数据集[7], drug2num, num2drug)
#打印(real2017)

# Diff = np.array(pred2017) - np.array(real2017)
# D = np;Sum (list(map(lambda x: abs(x),
  diff)))
# 打印
  (d)
# 打印('----------  2018 年预测  ----------
  -')
# Rel2018 = predict(feature, 2018)
# 打印
  (rel2018)
# 打印('----------  2019 年预测  ----------
  -')
# Rel2019 = predict(features, 2019)
# 打印
  (rel2019)
# excel = xlwt.Workbook()
# 表格= excel。add_sheet("基于 Zone 的预测日期")
# 表。写(0,0, "YYYY")
# 表。写(0,1, "Zone")
# 表。写入(0,2, "物质名称")
# 表。write(0,3, "DrugReportZone")

# row = 0
# for zone in range(len(rel2018)):
#     对于范围(69)内的药
      物:
#         Row += 1
#         物质名称= num2drug[药物]
#         表。写(行, 0,2018)
#         表。写入(行, 1,zone)
#         表。写(行, 2, 物质名)
#         表。写(行, 3,rel2018[区][药])
# for zone in range(len(rel2019)):
#     对于范围(69)内的药
      物:
#         Row += 1
#         物质名称= num2drug[药物]
#         表。写(行, 0,2019)
#         表。写入(行, 1,zone)
#         表。write(row, 2, substanceName)
#         表。写(行, 3,rel2018[区][药])



# excel.save("/用户/ lsd /桌面/代码/模型/ predict.xls")
```

数据分类

```
# -*-编码:utf-8 -*-

导入熊猫作为 pd
将 numpy 导入为 np
进口 sklearn.preprocessing


Label = []
Data = []
阿片类= pd.read_csv('阿片类总数。csv, index_col = 0)
county = opioid['FIPS_Combined '].unique()
对于范围(10,17)中的 I:
```

```
column =['地理。 id2']

data_csv = pd.read_csv('ACS_' +str(i)+ '_5YR_DP02/ 'ACS_' +str(i)+ '_5YR_DP02_with_ann ')Csv ',
index_col=0) for n in data_csv.columns.values:

    if 'HC03 ' in n:

        column.append (n)

Data_csv = Data_csv[列]

data.append (data_csv)

Label = np.unique(np. unique)追加(标签，
data_csv.loc['Id'].tolist()))标签=标签[1:len(标签)]


df = pd.DataFrame(columns=label)
for fips in county:
```

```
对行 标签：
    一个= []
    B = []
    为 范围(7)中的I:
        试
        试：
            A.append(int);loc[(阿片类['FIPS_Combined '] == fips)
                &(阿片类['YYYY']== i+2010), 'DrugReports ])
        除 :A.append(0)
        试
        试：
            B.append(float(data[i].loc[(data[i]['GEO]))Id2 '] ==
            str(fips)),
             (数据[我]。 loc['Id']== row)].values[0]
        除 :B.append(0)
        A = sklearn.预处理.normalize(np.array([A]), norm = 'l2 ')
        B = sklearn.预处理.normalize(np.array([B]), norm = 'l2 ')
            corr =
        np.corrcoef(A,B)[0][1] .
    df。 Loc [fips,row] =
            corr .
打印(fips)
对于 label 中的行：
    df。 Loc ['avg ', row] =
        df[row].mean()
df.to_csv(“rough_corr_result。 csv”,编码= 'utf - 8”)
```

关联规则学习

```
进口 xlrd
进口熊猫作为 pd
将 numpy 导入为 np
从 math import *
进口 xlwt
进口经营者
进口复制
从 functools 导入 reduce
进口 matplotlib。 Pyplot 作为PLT
进口 matplotlib
导入 networkx 为 nx
从 itertools 导入组合
从 igraph 导入 *
进口 seaborn 为 SNS
进口 pylab
进口随机
从 sklearn。 集群导入 KMeans
进口 sklearn.preprocessing
从 mlxtend.frequent_patterns 导入 apriori
from mlxtend.frequent_patterns import association_rules
from collections import Counter
sns.set_style(“darkgrid”)

get_drugReport(时间，位置，excel_output_add_new =
 pd.read_excel(“excel_output_add_new。 xls, sheet_name = 'biubiu ')):
    试
    试：
        返回 excel_output_add_new[(excel_output_add_new['FIPS_Combined']==
            location)&(excel_output_add_new['YYYY']== time)]['TotalDrugReportsCounty ']。 值[0]除外：
```

```
        返回 0
#打印(get_drugReport(2010,51001))


def get_new_dataFile_with_drug(时 间 ):
    Problem2_data = pd.read_excel(' Problem2_data .exe ')xlsx ', sheet_name= '
    Sheet1 ') dat = [0]
    for i in range(1, len(problem2_data['地 理 。 id2']):
        loc = problem2_data['地 理 。 id2'].iloc[i]
        k =get_drugReport(time, loc)
        dat。 追 加(get_drugReport(time, loc))
        打 印 (k)
    Problem2_data [' drug '] = dat
    print(dat)
    返 回 “ 成 功 ”


def Kmean_sub(数 据 ):
```

```
estimator =KMeans(n_clusters=2)

Label_pred =
estimator.labels_ .

Centroids = estimator。
Cluster_centers_惯性=
estimator.inertia_

打印(label_pred)

打印(重心)

打印(惯性)


def softmax (x):

"""计算 x 中每组分数的 softmax 值。""" e_x = np.;Exp (x -
np.max(x))

返回 e_x / e_x.sum(axis=0) #唯一的差值



Def to_bool(problem2_data, alpha):

Column_name = problem2_data.columns.values.tolist()

column_name = [x for x in column_name if (x[0] == 'H') and (x[3] == '3') or (x == 'DRUG'))]all_info = []

New_column_name = []

对于 range(1, len(column_name))中的
I:

Item = column_name[i]

Temp = []

试
一
试：

Threshold = problem2_data[item].iloc[1:].quantile(alpha)

对于范围(1,len(problem2_data[item]))中的 k:

If (problem2_data[item][k] >= threshold):

temp.append (1)

其他：

temp.append(0)
new_column_name.append(item)
all_info.append(temp)

除了：

print("Don 't work:", item)

Bool_frame = pd.;DataFrame(data = np.转置(np.array(all_info)),

column = new_column_name, index = problem2_data['地理。 id2'].iloc[1:].tolist())

打印(new_column_name)

bool_frame.to_excel (bool_pro2.xls, sheet_name = “biubiu”)

返回 new_column_name


Def get_corr(d, metadata, type_info):

d = d[d['DRUG']== 1]

D = D [type_info]

打印(d.shape)

#find_rule(d, support, confidence).to_excel(' rules.xls ')

# frequent_itemsets = apriori(d, min_support=0.05, use_colnames=True)

打印(“先天做!”)

Rules = association_rules(frequent_itemsets, metric= “lift” ,
min_threshold=1)

打印(“association_rules做!”)

=规则[(规则['取消']> = 1)&(规则(“信心”)> = 0.8)]。 sort_values("lift", ascending = fa# rules =
rules[规则['结果'].isin(["DRUG"])]

Dro = []

# df (df(“´L° Ua ˇ R ▪isin (lc ' Zyaž ˇ TcŽ ˇ Da˘ Aij)))

规则。 Index = range(len(rules[' lift ']))

对于 range(len(rules[' lift '])中的 I: m =
rules[' resulents ']。 iloc[i]如果不是
(m 中的“DRUG”):
```

```
            dro.append(
            我)

规则=规则。 Drop (index = dro)
```

规则。 Index = range(len(rules[' lift ']))
influence = rules[' antecedents ']。 Values
influence = [list(x) for x in influence]
temp5 = []

```
对于 range(len(influence))中 的
I:

    对于 range(len(influence[i]))中 的
    j:

        temp5.append(影响[我][j])

Thre = pd.value_counts(temp5).quantile(0.7)
```

```
    Freqence_temp5 = pd.value_counts(temp5)
    Freqence_temp5 = Freqence_temp5 [Freqence_temp5 >= thre]
    freIndex= freqence_temp5.index.tolist()
    元数据。index =元数据['地理。id'].values.tolist()
    Final_result = []
    for i in range(len(freIndex)):
        final_result。append([freIndex[i], freqence_temp5[freIndex[i]]), 元数据。loc[freIndex[i],
        'Id']])
    打印(final_result)
    返回 final_result
    #
    rules.to_csv(“rules.csv”)


Def classfy(all_type, metadata, people_type):
    Result = []
    对于 range(len(people_type))中 的
    I:
        result.append ([])
    元数据。index =元数据['地理。id'].values.tolist()
    for i in range(len(all_type)-1):
        Temp =元数据。loc[all_type[i], 'Id']
        对于 range(len(people_type))中 的
        j:
            If (people_type[j] in temp):
                结果[j] .append
                (all_type[我])
                打破
    #输出(结果)
    返回结果


Def get_all_corr(all_type, d, metadata, people_type):
    # slice_number = []
    # for I in range(0, len(all_type)-1, split_number): #
    if ((I +split_number)<= (len(all_type)-1)):
    #         slice_number。追加([我+ split_number])
    其他:
    #         slice_number。追加([我,len (all_type) 1])
    #打印(slice_number)
    CLS = classfy(all_type, metadata, people_type)
    All_info = []
    对于 range(len(cls))中 的
    I:
        #ty = all_type[slice_number[i][0]:slice_number[i][1]]
        ty = cls[i]
        ty.append(“DRUG
        ”)
        all_info。Append (get_corr(d, metadata,
        ty))
        print(“测试用例:”,
        cls[i])
        print("People type:%s\n" %people_type[i])
    打印(all_info)
    打印(’ ----------\ n \
    n” )
    All_info_new = []
    对于 range(len(all_info))中 的
    I:
        对于 range(len(all_info[i]))中 的 j:
            all_info_new.append(all_info[i][j])

    all_info_new。sort(key = lambda x: x[1], reverse = True)
    返回 all_info_new

如果__name__ == '__main__':
```

```
#excel_output_add_new = pd.read_excel(' excel_output_add_new. #excel_output_add_new .#xls,
sheet_name = ' biubiu " )

# get_new_dataFile_with_drug (2010)

'ACS_11_5YR_DP02_with_ann。xlsx ', sheet_name=
'ACS_11_5YR_DP02_with_ann#print(problem2_data['DRUG'].iloc[1:].quantile(0.6))

#predata = problem2_data['DRUG'].values[1:].重塑(-1,1)

#打印(predata)

# Kmean_sub (predata)

All_type = to_bool(problem2_data, 0.5)

Support = 0.06

信心= 0.75

Ms ='——'

D = pd.read_excel(' bool_pro2.xls ', sheet_name= ' biubiu ')

metadata = pd.read_excel(' ACS_11_5YR_DP02_metadata.xlsx ', sheet_name= ' ACS_11_5YR_DP02_metadata ')
#get_corr(d, 元数据)



#all_type = [' HC03_VC04 ', ' HC03_VC06 ', ' HC03_VC07 ', ' HC03_VC08 ', ' HC03_VC09 .                    HC03_VC
```

```
# ‘HC03_VC12’，‘HC03_VC13’，‘HC03_VC14’，‘HC03_VC15’，‘HC03_VC17’，‘HC03_VC18’，‘HC03_VC25’，‘HC03_ #
HC03_VC27”、“HC03_VC28”、“HC03_VC29”、“HC03_VC30”、“HC03_VC31”、“HC03_VC35”、“HC03_VC36”、
“HC03_
# ‘HC03_VC38’，‘HC03_VC39’，‘HC03_VC40’，‘HC03_VC42’，‘HC03_VC43’，‘HC03_VC44’，‘HC03_VC45’，‘HC03_ #
HC03_VC47”、“HC03_VC51”、“HC03_VC61”、“HC03_VC62”、“HC03_VC64”、“HC03_VC65”、“HC03_VC66”、
“HC03_
# ‘HC03_VC69’，‘HC03_VC70’，‘HC03_VC71’，‘HC03_VC75’，‘HC03_VC76’，‘HC03_VC77’，‘HC03_VC78’,HC03_ #的
HC03_VC80”、“HC03_VC84”、“HC03_VC85”、“HC03_VC86”、“HC03_VC87”、“HC03_VC88”、“HC03_VC89”、
“HC03_
# ‘HC03_VC91’，‘HC03_VC93’，‘HC03_VC94’，‘HC03_VC98’，‘HC03_VC99’，‘HC03_VC117’，‘HC03_VC118’，‘HC0 #
HC03_VC120” 、 “HC03_VC121” 、 “HC03_VC122” 、 “HC03_VC123” 、 “HC03_VC124” 、 “HC03_VC128” 、
“HC03_VC129“#”HC03_VC131” 、 “HC03_VC132” 、 “HC03_VC133” 、 “HC03_VC134” 、 “HC03_VC138” 、
“HC03_VC144” 、 “HC03_VC146
# ‘HC03_VC156’，‘HC03_VC166’，‘HC03_VC167’，‘HC03_VC168’，‘HC03_VC170’，‘HC03_VC171’,’ HC03_VC17 #
HC03_VC174 ', ' HC03_VC175 ', ' HC03_VC176 ', ' HC03_VC177 ', ' HC03_VC178 ', ' HC03_VC182 ', ' HC03_VC18 # '
HC03_VC185 ', ' HC03_VC186 ', ' HC03_VC187 ', ' HC03_VC189 ', ' HC03_VC190 ', ' HC03_VC19 # ' HC03_VC193 ', '
HC03_VC194 ', ' HC03_VC195 ', ' HC03_VC196 ', ' HC03_VC197 ', ' HC03_VC203 ', ' HC03_VC202 ', ' HC03_VC205 ', '
HC03_VC206 ', ' HC03_VC209 ', 'DRUG']


people_type =['按类型划分的家庭', '关系', '婚姻 STATUS',
“生育能力”，“祖父母”，“入学率”，“受教育程度”，
“退伍军人 STATUS”，“平民非收容人口的残疾 STATUS”，“居住一年前”，“出生地”，“美国”公民
STATUS”，”入境年份”，”外国出生的世界地区”，”国内使用的语言”，”祖先”]


打印（’ ----------\ n \
n”）
Print (get_all_corr(all_type, d, metadata, people_type)) #
slice_number = []
# split_number = 25 .
# if ((I +split_number)<= (len(all_type)-1)):
#       slice_number。追加([我+ split_number])
其他:
#       slice_number。追加([我,len (all_type) 1])
#打印(slice_number)
# for I in range(len(slice_number)):
# ty = all_type[slice_number[i][0]:slice_number[i][1]] #
ty.append("DRUG")
#打印(泰)
```

SVM分类器及阈值分析

```
report = opioidtotalbyzone.DrugReportZone;县域=
鸦片 totalbyzone. totaldrugreportzone;A =[报告县];
idx = zeros(size(A,1),
1);
我=1:1大小(1)
    如果一个(我,2)< 125000 - 100
    *(我,1)
        Idx (i) = 1;
    其
    他的
        Idx (i) = 2;
    结束
结
束


图；
情节((:1)。 *(idx-1)、 (:,2)。 *(idx-1),'r。 ');
抓住；
情节((:1)。 *(idx-2),(:,2)。 *(idx-2),'b。 ');idx
(idx == 2) = 1;
svm = fitcsvm(A,idx);
```

svInd= svm.IsSupportVector;

数字

情节((:1)(:,2),“ k” 。)

抓住

plot(A(svInd， 1)， A(svInd， 2)， ' ro ' ， ' MarkerSize ' ， 10) %输入新数据这里的评分是输出 1或-1
predict_data = predictopioidtotal.DrugReportZone;

```
County = County(1:length(predict_data));B
= [predict_data County];


score = predict(svm,B);


title({\bf 离群点检测 via One-Class SVM})xlabel('阿片类
药物报告')
ylabel(“各县总药物报告”)
传奇(“观察”、“支持向量”)
推迟
```