数据离散化是指将连续的数据进行分段，使其变为一段段离散化的区间。  
  
朴素贝叶斯算法，是基于离散型的数据展开的。如果要使用该类算法，必须将离散型的数据进行。有效的离散化能减小算法的时间和空间开销，提高系统对样本的分类聚类能力和抗噪声能力。  
  
离散化的特征相对于连续型特征更易理解，更接近知识层面的表达  
  
可以有效的克服数据中隐藏的缺陷，使模型结果更加稳定

K-means聚类算法把连续型数据离散化

算法的主要思想是通过迭代过程把数据集划分为不同的类别，使得评价聚类性能的准则函数达到最优，从而使生成的每个聚类内紧凑，类间独立。这一算法不适合处理离散型属性，但是对于连续型具有较好的聚类效果。

把关注房源人数、平米价、面积等数据用K-means聚类算法把连续型数据离散化

代码：

（关注房源人数）

k = 7

data\_people\_num = data['关注房源人数'].copy()

kmodel = KMeans(n\_clusters = k, n\_jobs = 1)#n\_jobs是并行数，一般等于CPU数

kmodel.fit(data\_people\_num.values.reshape(len(data\_people\_num), 1))

c = pd.DataFrame(kmodel.cluster\_centers\_).sort\_values(by=0)

#rolling\_mean表示移动平均，即用当前值和前2个数值取平均数，

#由于通过移动平均，会使得第一个数变为空值，因此需要使用.iloc[1:]过滤掉空值。

w = c.rolling(2).mean().iloc[1:] #相邻两项求中点，作为边界点

w = [0] + list(w[0]) + [data\_people\_num.max()] #把首末边界点加上

dPNum = pd.cut(data\_people\_num,w,labels=range(k))

def cluster\_plot\_Pnum(d,k):

import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

plt.figure(figsize = (12,4))

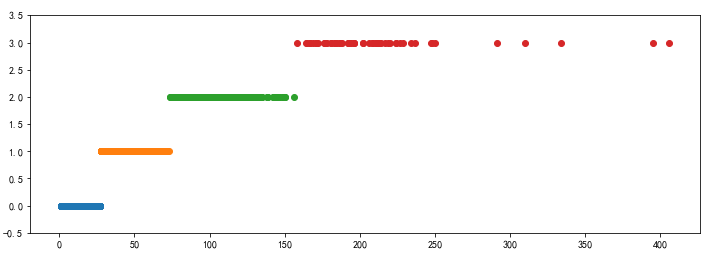
for j in range(0,k):

plt.plot(data\_people\_num[d==j], [j for i in d[d==j]],'o')

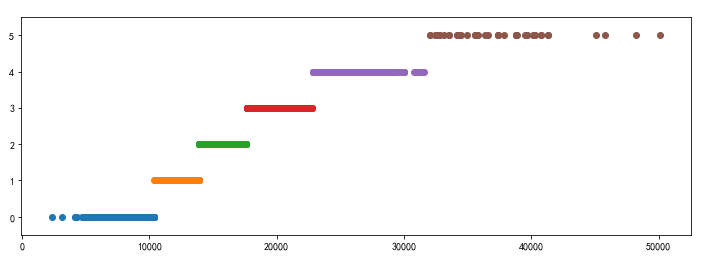
plt.ylim(-0.5, k-0.5)

return plt

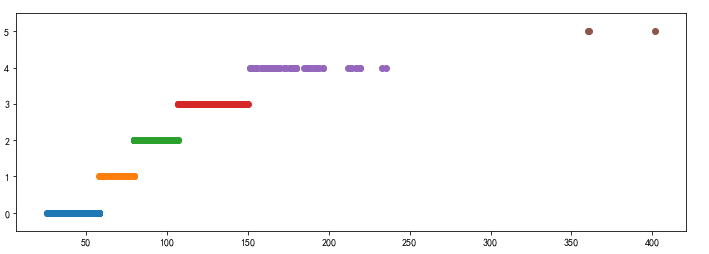
cluster\_plot\_Pnum(dPNum, k).show()



关注房源人数



平米价



面积

从处理后的数据中提取出特征值和目标值

Alldata=[]

traffic\_feature=[]

traffic\_target=[]

x = data.iloc[:, [0,2,3,4,5,9,11,13,16,17,28,29,25]].as\_matrix()

xFrame = pd.DataFrame(x)

for indexs in xFrame.index:

Alldata.append(xFrame.loc[indexs].values[0:-1])

traffic\_feature.append(xFrame.loc[indexs].values[1:13])

traffic\_target.append(xFrame.loc[indexs].values[0])

scaler = StandardScaler() # 标准化转换

scaler.fit(traffic\_feature) # 训练标准化对象

traffic\_feature= scaler.transform(traffic\_feature) # 转换数据集

feature\_train, feature\_test, target\_train, target\_test = train\_test\_split(

traffic\_feature, traffic\_target, test\_size=0.3,random\_state=0)

（所有数据中取30%作为测试数据，70%作为训练数据）

NB=BernoulliNB()

NB.fit(feature\_train,target\_train)

print(NB.score(feature\_test,target\_test)) #预测结果与实际结果准确程度



conf\_mat = confusion\_matrix(target\_test, predict\_results) #混淆矩阵 表明类别是否被混淆 对角线上的元素是被正确识别的数量 第i行第j列表示，本应该属于第i类，实际被划分到第j类

print(conf\_mat)

