**数据预处理的几个方面**

**【Python数据处理】pandas.DataFrame格式数据转为列表List或数组array**

**假设wordsdf是pandas.DataFrame格式数据**

import numpy as np

array\_data = np.array(wordsdf)#df数据转为np.ndarray()

list\_data=array\_data.tolist()#将np.ndarray()转为列表

dict\_data = dict(list\_data)#将列表转为字典

**数据预处理的主要内容包括：**

1. 数据清洗；
2. 数据集成；
3. 数据转换；
4. 数据规约；

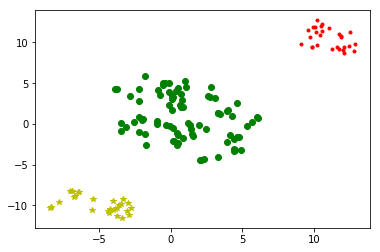
**1、数据归一化处理**

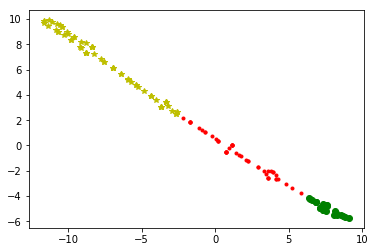
[**MinMaxScaler：归一到 [ 0，1 ]**](https://blog.csdn.net/weixin_40683253/article/details/81508321#MinMaxScaler%EF%BC%9A%E5%BD%92%E4%B8%80%E5%88%B0%20%5B%200%EF%BC%8C1%20%5D%C2%A0)

[**MaxAbsScaler：归一到 [ -1，1 ]**](https://blog.csdn.net/weixin_40683253/article/details/81508321#MaxAbsScaler%EF%BC%9A%E5%BD%92%E4%B8%80%E5%88%B0%20%5B%20-1%EF%BC%8C1%20%5D%C2%A0)

做聚类分析的时候发现，聚类效果往往特别**受其中一列数据的影响**，使得原本应该散布在二维平面图上的点，变成聚集在一条线上的点，其**聚类效果肯定不理想**。

**左图：为所有数据都归一化之后的聚类分析散点图；**

**右图：为其中一列是合同金额，并且没有归一化数据的散点图；**



归一化方法有两种形式，一种是把数变为**（0，1）**之间的小数，一种是把有量纲表达式变为**无量纲表达式（-1，1）**，成为纯量。**后者常见于微波之中**，也就是**电路分析、信号系统、电磁波传输**等，研究物理的人会比较熟悉。而像我们这些普通的数据分析师的日常工作中，不太会遇见需要归一化为无量纲表达式的情况，因此只讨论归一化到 [0，1] 的情况。

归一化一般是把数据映射到 [ 0，1] ，但也有归一到[ -1，1] 的情况，两种情况在Python中分别可以通过**MinMaxScaler 或 MaxAbsScaler方法**来实现。

**MinMaxScaler：归一到 [ 0，1 ]**

X_scaled = \frac{ (X - X.min(axis=0)) }{ (X.max(axis=0) - X.min(axis=0))} \cdot (max - min)+min

从原理中我们注意到有一个**axis=0**，这表示MinMaxScaler方法默认是**对每一列做这样的归一化操作**，这也比较符合实际应用。

**MaxAbsScaler：归一到 [ -1，1 ]，原理与MinMaxScaler相似。**

**2、数据标准化处理- scale()：[去均值，方差规模化](https://blog.csdn.net/weixin_40683253/article/details/81508321" \l "1-%E6%A0%87%E5%87%86%E5%8C%96%E5%8E%BB%E5%9D%87%E5%80%BC%E6%96%B9%E5%B7%AE%E8%A7%84%E6%A8%A1%E5%8C%96" \t "_self)**

数据分析的过程中，比如线性规划这一类的分析，如果有些特征的数值远远高于或低于其他数值，通常称之为独立点、异常值或噪点，那么对于受噪点影响较大的模型就无法正确地去学习其他特征。

Standardization标准化：将特征数据的分布调整成标准正太分布，也叫高斯分布，过程为两步：去均值的中心化（均值变为0）；方差的规模化（方差变为1）。

在sklearn.preprocessing中有一个scale方法，可以实现数据标准化，该方法默认按照列进行标准化。

**3、数据清洗**

数据挖掘过程中，采集的原始数据里存在着各种不利于分析与建模工作的因素，比如**数据不完整、数据矛盾、异常值**等。这些因素不仅影响建模的执行过程，更有甚者在不知不觉间**给出错误的建模结果**，这就使得数据清洗显得尤为重要。但是数据清洗并不是数据预处理的全部内容，它只是第一步而已，接下来还有数据集成、数据转换和数据规约等一系列处理。在实际应用中，数据预处理的工作量占整个建模过程的60%，可以说，预处理做得好，模型基本就出来了。

**（1）**[**缺失值处理**](https://blog.csdn.net/weixin_40683253/article/details/81665896#%E7%BC%BA%E5%A4%B1%E5%80%BC%E5%A4%84%E7%90%86)

[删除缺失值](https://blog.csdn.net/weixin_40683253/article/details/81665896#%E5%88%A0%E9%99%A4%E7%BC%BA%E5%A4%B1%E5%80%BC)

[插补缺失值](https://blog.csdn.net/weixin_40683253/article/details/81665896#%E6%8F%92%E8%A1%A5%E7%BC%BA%E5%A4%B1%E5%80%BC)

[不处理缺失值](https://blog.csdn.net/weixin_40683253/article/details/81665896#%E4%B8%8D%E5%A4%84%E7%90%86%E7%BC%BA%E5%A4%B1%E5%80%BC)

**（2）**[**重复值处理**](https://blog.csdn.net/weixin_40683253/article/details/81665896#%E9%87%8D%E5%A4%8D%E5%80%BC%E5%A4%84%E7%90%86)

**（3）**[**异常值处理**](https://blog.csdn.net/weixin_40683253/article/details/81665896#%E5%BC%82%E5%B8%B8%E5%80%BC%E5%A4%84%E7%90%86)

[遍历查找异常值，并根据规则调整大小](https://blog.csdn.net/weixin_40683253/article/details/81665896#%E9%81%8D%E5%8E%86%E6%9F%A5%E6%89%BE%E5%BC%82%E5%B8%B8%E5%80%BC%EF%BC%8C%E5%B9%B6%E6%A0%B9%E6%8D%AE%E8%A7%84%E5%88%99%E8%B0%83%E6%95%B4%E5%A4%A7%E5%B0%8F%C2%A0)

[删除异常值](https://blog.csdn.net/weixin_40683253/article/details/81665896#%E5%88%A0%E9%99%A4%E5%BC%82%E5%B8%B8%E5%80%BC)

[视为缺失值后进行插补](https://blog.csdn.net/weixin_40683253/article/details/81665896#%E8%A7%86%E4%B8%BA%E7%BC%BA%E5%A4%B1%E5%80%BC%E5%90%8E%E8%BF%9B%E8%A1%8C%E6%8F%92%E8%A1%A5)

**数据清洗主要是删除原始数据中的无关数据，重复数据，平滑噪声数据，筛选掉与建模目的无关的数据，处理缺失值与异常值等。**

**（1）**[**缺失值处理**](https://blog.csdn.net/weixin_40683253/article/details/81665896#%E7%BC%BA%E5%A4%B1%E5%80%BC%E5%A4%84%E7%90%86)

**删除缺失值**

**删除缺失值具体的情况是一下几种：**

data.dropna() #直接删除含有缺失值的行

data.dropna(axis = 1) #直接删除含有缺失值的列

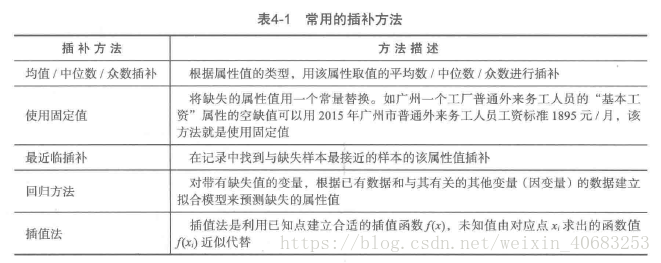
data.dropna(how = 'all') #只删除全是缺失值的行

data.dropna(thresh = 3) #保留至少有3个非空值的行

data.dropna(subset = [u'血型 ']) #判断特定的列，若该列含有缺失值则删除缺失值所在的行

**插补缺失值**

**常见的插补方法：**

****

**简单的缺失值插补方法：**

data.fillna(data.mean()) #均值插补

data.fillna(data.median()) #中位数插补

data.fillna(data.mode()) #众数插补

data.fillna(data.max()) #最大值插补

data.fillna(data.min()) #最小值插补

data.fillna(0) #固定值插补--用0填充

data.fillna(5000) #固定值插补--用已知的行业基本工资填充

data.fillna（method='ffill'）#最近邻插补--用缺失值的前一个值填充

data.fillna（method='pad'） #最近邻插补--用缺失值的前一个值填充

**通过拟合函数来插补的方法：**

主要说一说拉格朗日插值法吧，除了**拉格朗日插值法**，还有**牛顿插值法**、**Hermite插值法**、**分段插值法**和**样条插值法**。

**（2）**[**重复值处理**](https://blog.csdn.net/weixin_40683253/article/details/81665896#%E7%BC%BA%E5%A4%B1%E5%80%BC%E5%A4%84%E7%90%86)

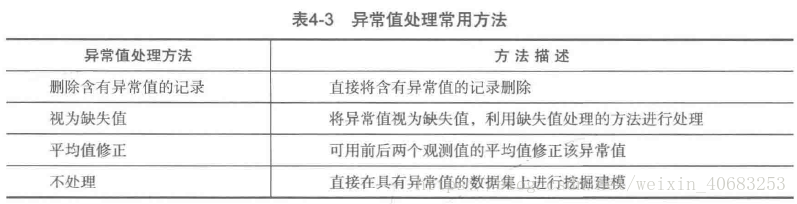
在Pandas中，.duplicated()表示找出重复的行，默认是判断全部列，返回布尔类型的结果。对于完全没有重复的行，返回 False，对于有重复的行，第一次出现的那一行返回 False，其余的返回 True。

与.duplicated()对应的，.drop\_duplicates()表示去重，即删除布尔类型为 True的所有行，默认是判断全部列。

**（3）**[**异常值处理**](https://blog.csdn.net/weixin_40683253/article/details/81665896#%E7%BC%BA%E5%A4%B1%E5%80%BC%E5%A4%84%E7%90%86)

数据清洗过程中的异常值的处理，是选择剔除还是用其他值代替，需要视情况而定。有些异常值可能包含某些信息，需认真思考后采取处理方法。

**常见的异常值处理办法：**

****

**4、数据降维方法**

**为什么要降维**

建模初期，我们往往只有几个指标，这个时候不太涉及到降维，但是一个月后你就发现，模型的指标越来越多，从原有的五六个指标一步一步变成 100 个指标。100 个很多吗？不多！但是以后呢？两个月过去可能会变成 500 个，三个月过去就会超过 1000 个，以后还会更多！

这么多指标里面，肯定有冗余，怎么找出他们？总不能一个一个分析吧......所以需要更高效快捷的方法来处理高维数据，比如降维（减少指标数量的同时，避免丢失太多信息、改进模型性能的方法）。

**降维的好处：**

1. 减少数据存储所需的空间，节约成本。
2. 减少数据处理与建模的时间，提高效率。
3. 提高算法的性能（一些算法在高维度数据上容易表现不佳）。
4. 解决多重共线性问题。
5. 有助于数据可视化（绘制二维三维数据图表比较简单）

**降维方法**

**（1）缺失值比例（高：删！）**

学校做课题时拿到的数据，缺失值是相对较少的，但是在工作中，缺失值的比例简直让人瞠目结舌，高于 80% 都司空见惯了。当缺失值比例过高时，代表它包含的可用信息太少了，一般会设置一个阈值，缺失值比例高于阈值的特征便删掉。

**（2）方差（低：删！）**

若某一个特征中的数据基本一致，代表这一特征对因变量 y 的解释能力是比较低的，因此我们认为这种特征包含的可用信息也很少。特征中的数据基本一致的直接表现就是方差比较小。

在实际应用中，不同特征的数据范围是不一样的，有 [ 0, 1 ] 的，也有 [ 1, +∞) 的，因此通过这种方法降维之前一定要先对数据进行归一化处理（归一化与标准化不同）。

**（3）相关性矩阵（高：保留其一！）**

如果两个特征的相关度较高，说明这两个特征携带的信息是高度相似的，过多的相似特征会降低某些算法的性能，比如逻辑回归（Logistic Regression）和线性回归（Linear Regreesion）。因此，一般会设置一个阈值，相关度高于阈值的特征只保留其中一个。

在下例中，因为特征数量较多，不便于展示，所以我们只选取 'x91', 'x92', 'x95', 'x96',' x97' 这五个特征进行相关度计算。

**输出**

**x91 x92 x95 x96 x97**

**x91 1.000000 0.897225 -0.086567 -0.102104 -0.099413**

**x92 0.897225 1.000000 -0.080699 -0.093550 -0.106207**

**x95 -0.086567 -0.080699 1.000000 0.844717 0.724978**

**x96 -0.102104 -0.093550 0.844717 1.000000 0.841306**

**x97 -0.099413 -0.106207 0.724978 0.841306 1.000000**

选取的五个特征中，'x91' 与 'x92' 的相关度达0.89，高相关，保留其一即可，其他同理。

0~0.3 弱相关

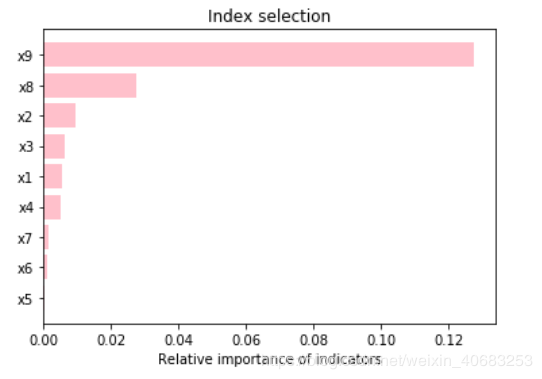
0.3~0.6 中等程度相关

0.6~1 强相关

**（4）随机森林（低：删！）**

随机森林是一种广泛使用的特征选择算法，RandomForestRegressor 模块中的 feature\_importances\_ 函数会自动计算各个特征的重要性，因此大大降低了编程的难度。之后再根据特征间的相对重要性选择指标即可。

需要注意的是，随机森林只接受数字，不接受空值、逻辑值、文字等其他类型，因此需要**先填充空值，转化特征值**等。



**（5）反向特征消除**

反向特征消除是指先选择全部的特征进行建模，再根据模型评价指标，**逐步删除特征**。

**步骤：**

a.获取数据集中的全部 100 个特征和因变量（ y\_增长率）。

b.使用特征与因变量训练模型（逻辑回归 or 线性回归 or ......）。

c.计算模型评价指标（召回率、精确率、F1值、准确率）。

d.删除第 i 个特征后，用剩下的 99 个特征训练模型，并计算模型评价指标。

e.比较两次的评价指标，若评价指标提高，则删除第 i 个特征。

f.循环 4 ~ 5 过程，直到模型评价指标不再明显增加。

**（6） 前向特征选择**

前向特征选择是相对反向特征消除而言的，前向特征选择并不是一开始就选择所有的特征，而是先选择最优的单个特征，再根据模型评价指标，逐步增加特征，直到召回率、精确率等评价指标不再提高为止。

**步骤：**

a.将每个特征与因变量 y 训练模型，得到 100 个模型 与 100 个模型评价指标。

b.选择模型评价指标最高的特征作为已入选特征。

c.添加一个特征到已入选特征中继续训练模型。

d.根据模型评价指标，选择第二个入选的特征，加入到已入选特征中。

e.循环 3 ~ 4 过程，直到模型评价指标不再明显增加。

**注意：**

前向特征选择和反向特征消除需要循环建模，耗时较久，计算成本很高，所以只适用于特征较少的数据，超过 1000 个特征的数据就慎重了。

**（7） 因子分析（Factor Analysis）**

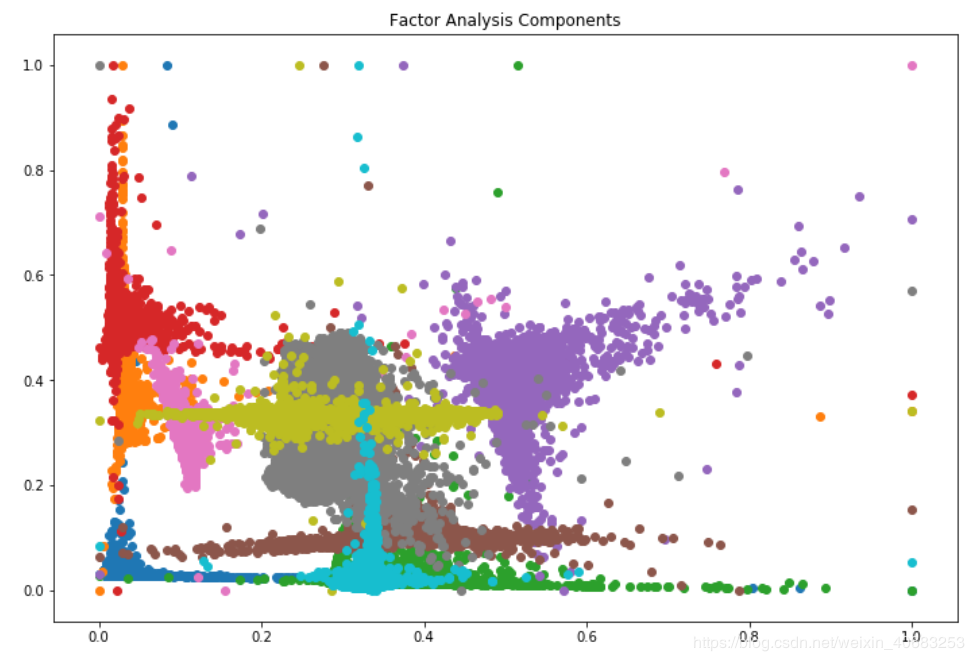
与前几种方法不同的是，**因子分析并不是通过删除特征来降低冗余数据的影响，而是通过挖掘潜在因子，在不删除特征的前提下降低影响。**一般情况下，潜在因子的个数是远少于特征个数的，但是其携带的信息并不比原特征的少。

和主成分分析相似，首先从原理上说，主成分分析是试图寻找原有自变量的一个线性组合，取出对线性关系影响较大的原始数据，作为主要成分。

**因子分析**，是假设所有的自变量可以**通过若干个因子（中间量）被观察到**。什么意思呢，举个例子，比如一个学生的考试成绩，语文80，数学95，英语79，物理97，化学94 ，那么我们认为这个学生理性思维较强，语言组织能力较弱。其中理性思维和语言组织能力就是因子。通过这两个因子，我们能够观察到他的偏理科的成绩较高，偏文科的成绩较低。这就是因子分析，通过这点，大家就可以感受到，因子分析和主成分分析是明显不一样的。

因子分析又存在两个方向，一个是**探索性因子分析**（exploratory factor analysis）。另一个是**验证性因子分析**（confirmatory factor analysis）。探索性因子分析是不确定一堆自变量背后有几个因子，我们通过这种方法试图寻找到这几个因子。而验证性因子分析是已经假设自变量背后有几个因子，试图通过这种方法去验证一下这种假设是否正确。验证性因子分析又和结构方程模型有很大关系。后面我们会专门的介绍，今天先介绍探索性因子分析。

**结果：**



能不能找到10种颜色呢？哈哈，100 维到 10 维，没有删减特征，却成功降维啦！

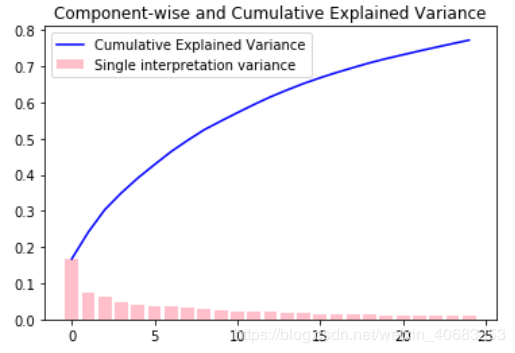
**（8） 主成分分析（PCA）**

**因子分析是假设存在一系列潜在因子**，能反映变量携带的信息，本质是一种**向下挖掘**的方法。

而**主成分分析PCA**则是通过**正交变换**将原始的n维特征变换到一个新的被称做主成分的数据集中，本质是一种**向上提取**的方法。

**主成分分析通过累计的解释方差之和来判断主成分对所有特征的解释程度。**

**输出：**

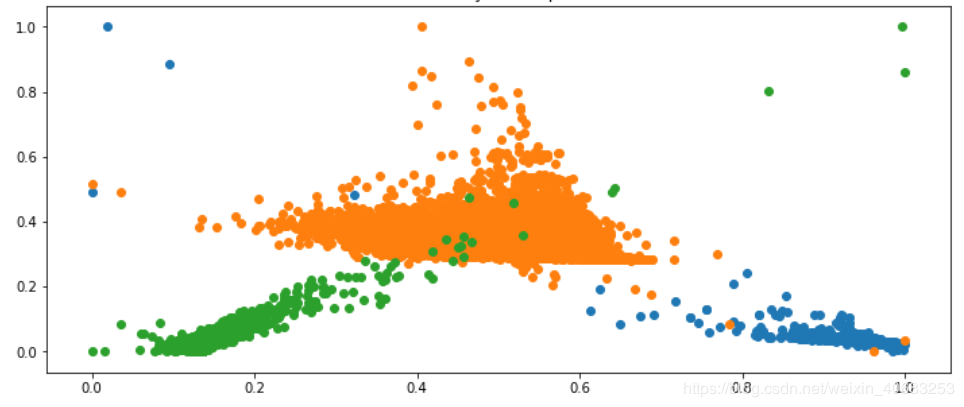
****

图中，粉色柱状图一共25个柱子，代表提取的25个主成分，其高度代表每个主成分对方差的解释程度。

蓝色折线图代表25 个主成分对方差的累计解释程度。我们可以看到，25 个主成分对100 个特征方差的累计解释程度已经达到了80%，因此这25个主成分携带了原始特征中的大部分信息，成功降维！

**（9）独立分量分析（ICA）**

PCA 和 ICA 之间的主要区别在于，PCA寻找不相关的因素，而ICA寻找独立因素。如果两个特征是独立的，就代表两者之间没有任何关系。例如，今天下不下雨和我工资多少无关。

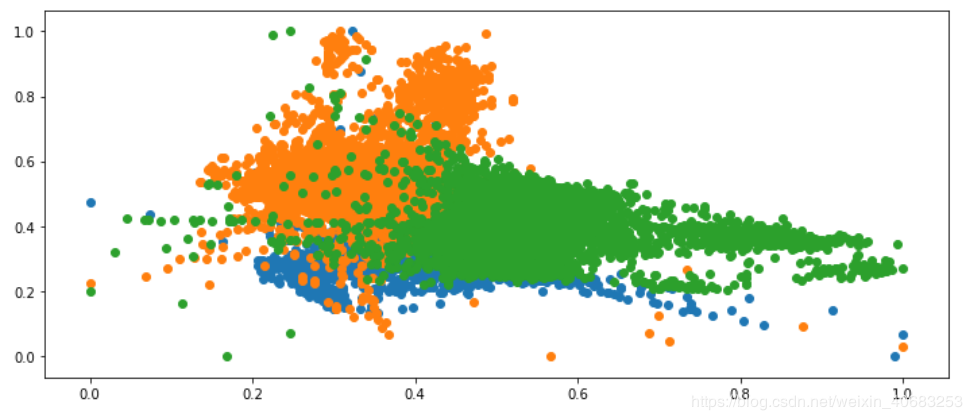


**（10）manifold.Isomap算法**

**isomap算法主要流程：**

1. 构建邻接图G：基于输入空间X中流形G上的的邻近点对i,j之间的欧式距离dx (i,j)，选取每个样本点距离最近的K个点（K-Isomap）或在样本点选定半径为常数ε的圆内所有点为该样本点的近邻点，将这些邻近点用边连接，将流形G构建为一个反映邻近关系的带权流通图G；
2. 计算所有点对之间的最短路径：通过计算邻接图G上任意两点之间的最短路径逼近流形上的测地距离矩阵DG={dG(i,j)}，最短路径的实现以Floyd或者Dijkstra算法为主。
3. 构建k维坐标向量：根据图距离矩阵DG={dG(i,j)}使用经典Mds算法在d维空间Y中构造数据的嵌入坐标表示，选择低维空间Y的任意两个嵌入坐标向量yi与yj使得代价函数最小。

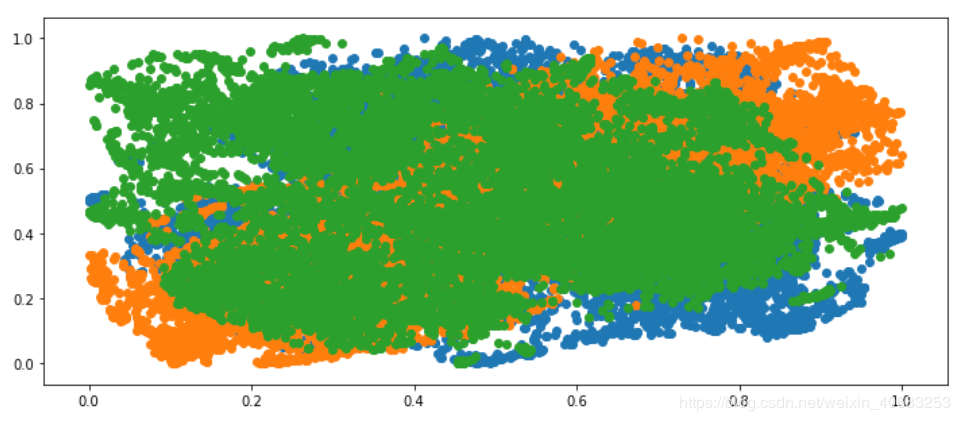
**输出：**

****

**运行耗时，降维效果差，结果不满意！**

**11.manifold.TSNE算法**

**输出：**

****

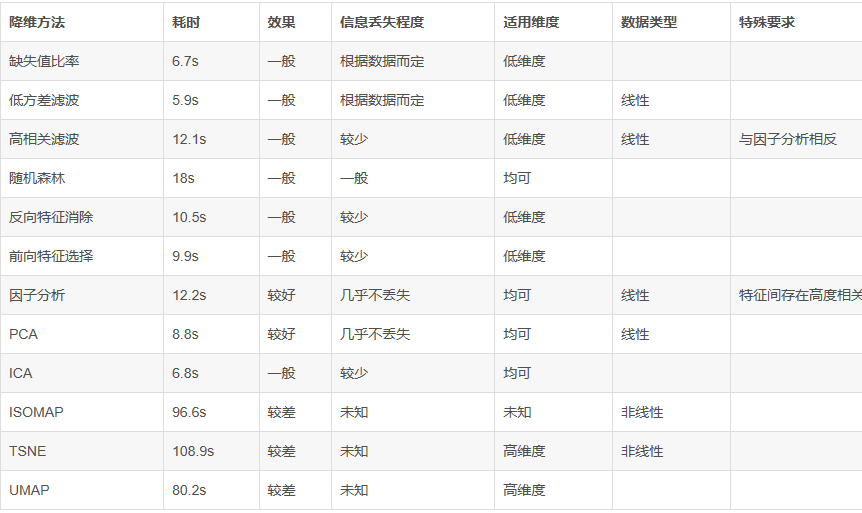
运行耗时，降维效果差，不满意！

**12.UMAP算法**

运行耗时，结果不满意！

## 大总结

这么多中方法，总有各自的优缺点，下面总结一下：其中耗时为100个特征，11528行数据的运行耗时。



文中所说的反向特征消除与前项特征消除比较耗时是针对高维度数据来说了，根据实际情况，在低维度数据中，两种方法并没有很耗时。