1．实验目的

（1）掌握Pandas Dataframe的使用方法。

（2）掌握数据探索基本方法。

（3）掌握数据可视化基本方法。

（4）能够处理缺失值、对数据进行恰当的转码。

（5）能够选择合适的数值规范化方法

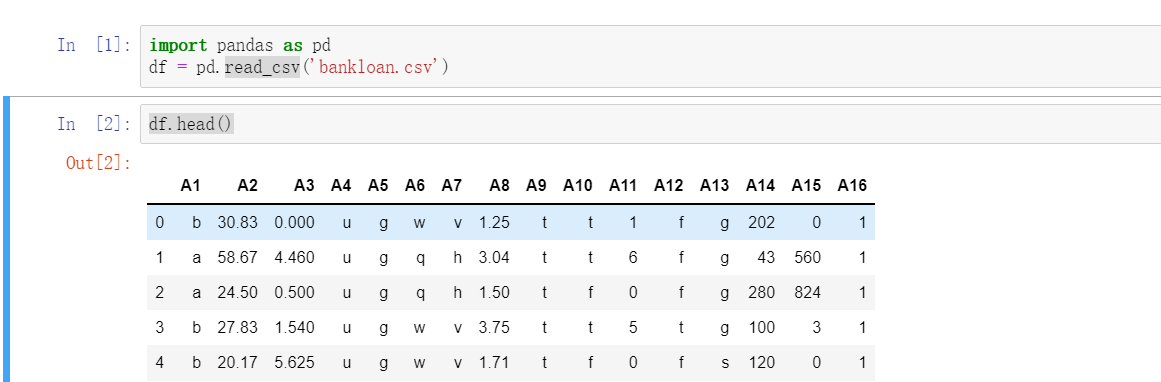
2．实验数据

数据文件bankloan.csv。银行贷款数据，一共包含690个样本，16个属性。基于隐私保护的考虑数据已经脱敏，属性名称用A1，A2，…，A16来表示，其中A16是类标号属性，有两个取值：“+”和“﹣”，“+”表示信贷申请被通过，“﹣”表示信贷申请被否决。需要注意的是，一些属性的值为“NA”，表示值缺失。

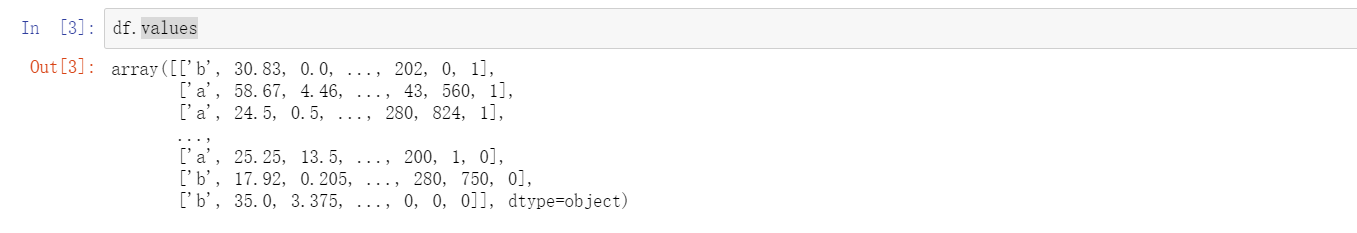
3．实验步骤和实验结果

（1）读取数据，以矩阵存储数据

首先先读取bankloan.csv文件，并使用read\_csv读取数据转换成dataframe的形式，并且再用head()函数来查看数据的格式

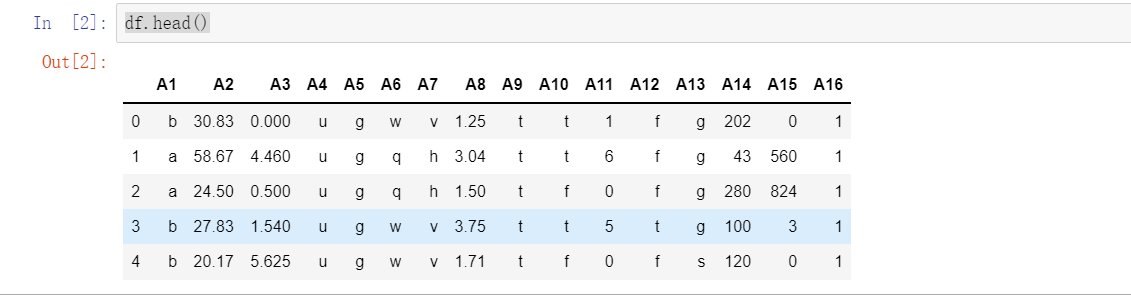


然后用values函数将dataframe的数据格式转化为矩阵用于保存数据

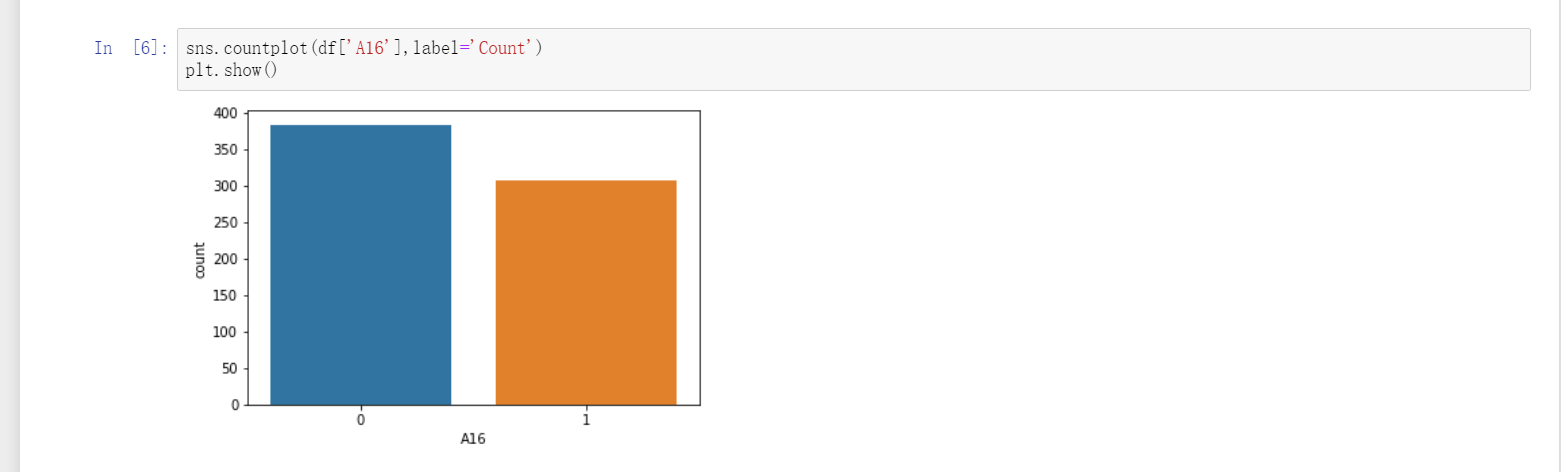


（2）探索数据，并适当可视化

在探索数据的过程中，我们基于上面的head()，先了解的数据的样式

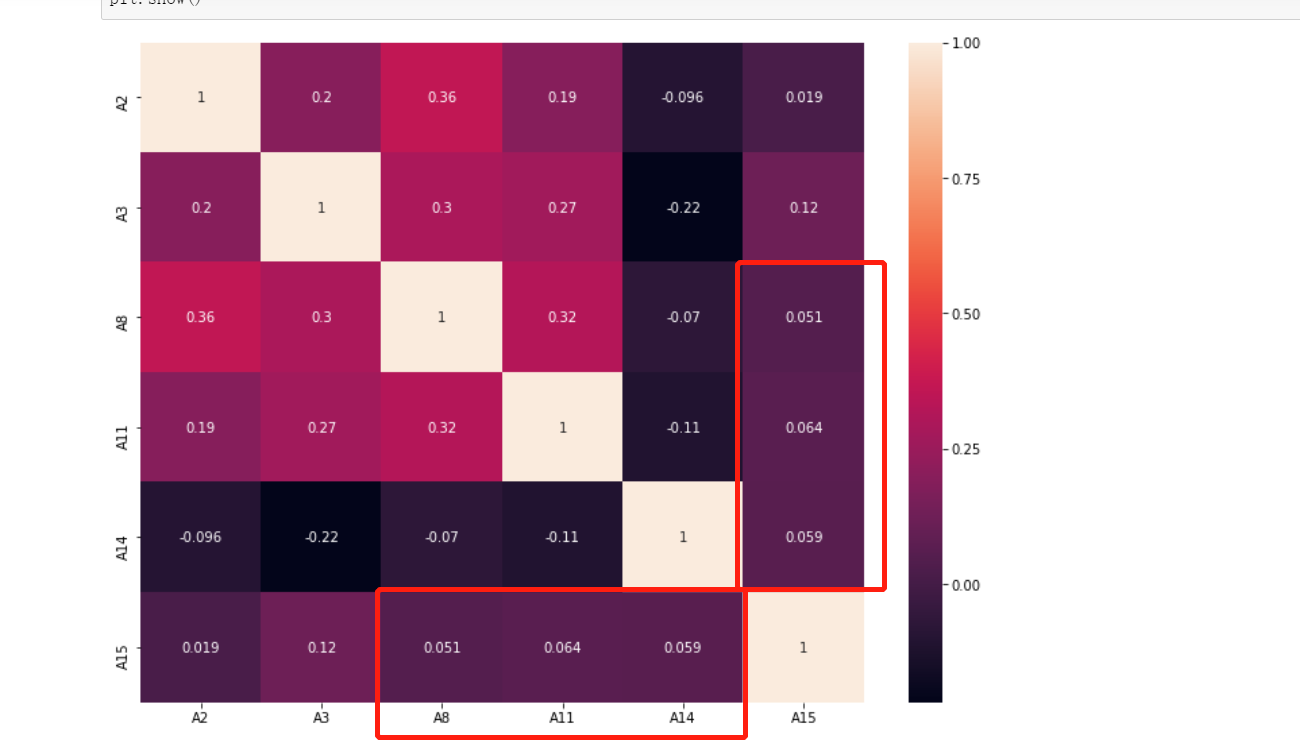


然后对数据进行一个探讨的过程，首先根据上面要求，我们可以知道A16就是我们最后的结果，前面所有的值都是为A16做铺垫的，所以我们首先先检查A16的值，对其进行可视化处理



在这里我们调用了是seaborn可视化库，有图可知，0意味着信贷申请被否决，1意味着信贷申请被通过，因此我们可以做一个简单的判断，信贷申请被否决明显多于信贷申请被通过，看来信贷申请还是想对较为困难的，通过的条件较为苛刻

然后我们再对其他的数据做一个热力图进行判断，看看哪个数值在申请的过程占据重要成分



由图可知，看来A8,A11,A14这三个参数，在从多参数中占据重要比重，信贷是否能申请通过，和这三个参数有着莫大的关系，所以想要信贷通过关于A8,A11,A14要做好相关的资料，使得贷款得以顺利进行

（3）恰当处理缺失值

关于缺少值，我们首先来看看总体的缺失值，看看哪些列，存在一定的缺少值

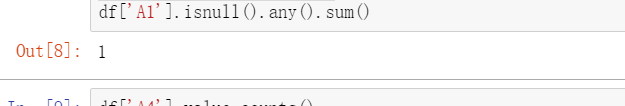
首先我们先用isnull().any()来查看每一列的缺少值总体划分



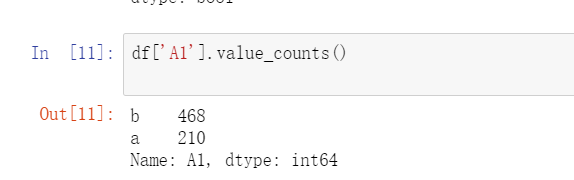
True意味着存在缺少值

知道存在的缺失值之后，我们可以看看，单独一列，缺少值的统计

用isnull().any().sum()来统计其中一列的缺少值

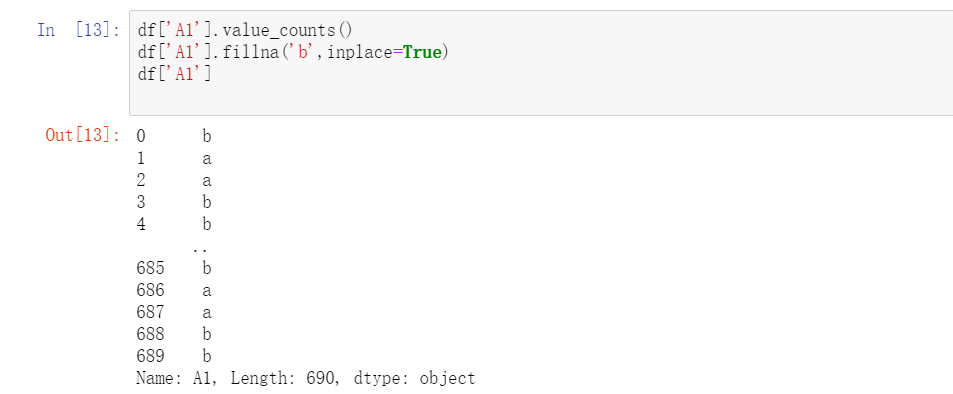


整体来看，缺失值整体缺少较少，大多在1-2之间针对这种情况，我们可以采取用众数来代替缺少值，这样可以既不影响整体的情况下，也能很好的处理缺失值，我们可以用value\_counts()函数来得知每一列的众数是什么，然后再采取众数来替代缺失值



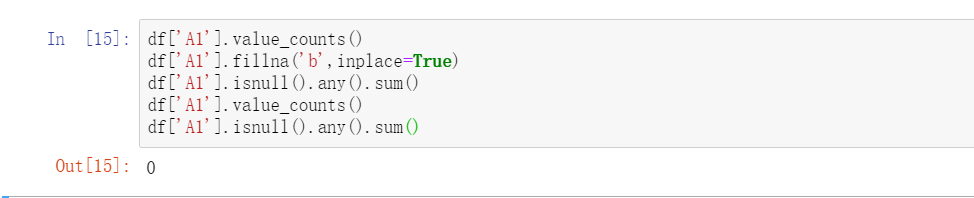
有图可知，在A1列中，b是最多的，所以我们可以采用b来代替缺少值，替换的函数是fillna

具体写法如下



这样我们就替换好了

然后再去检查的时候



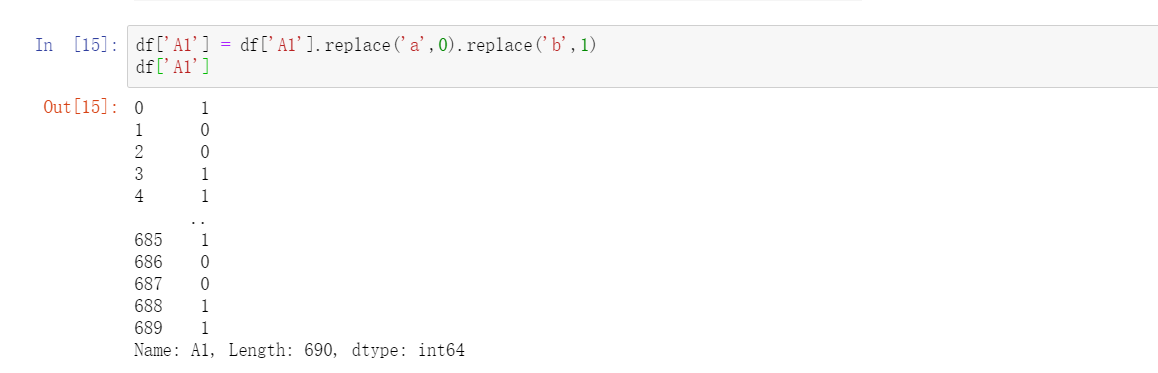
缺少值就已经全部不存在了

（4）对非数值属性，编程替换为数值

对于非数值的属性，我们可以做一个简单的替换

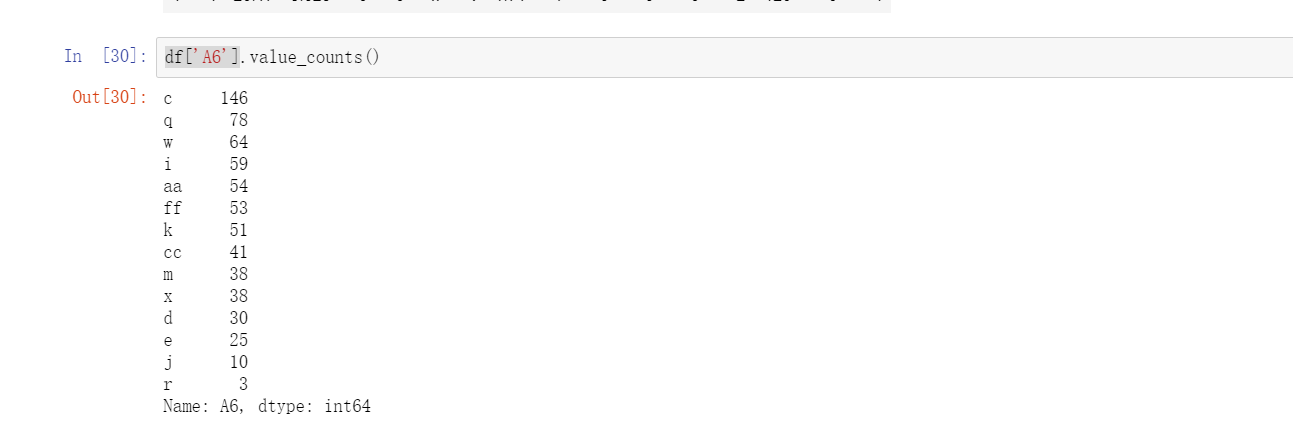
例如在A1列，一共有a,b两个类型，针对这两个类型，我们可以直接替换成数字就好了，a替换为0，b替换为1

具体做法如下



而对于一些像A6列有多种不同的非数值属性，我们就要写一个对应的函数去进行处理，因为类型太多，一个个手动去替换，有点不合理，也较为废时间，所以我们可以写一个新的函数去处理这些多种不同的非数值类型

如下：



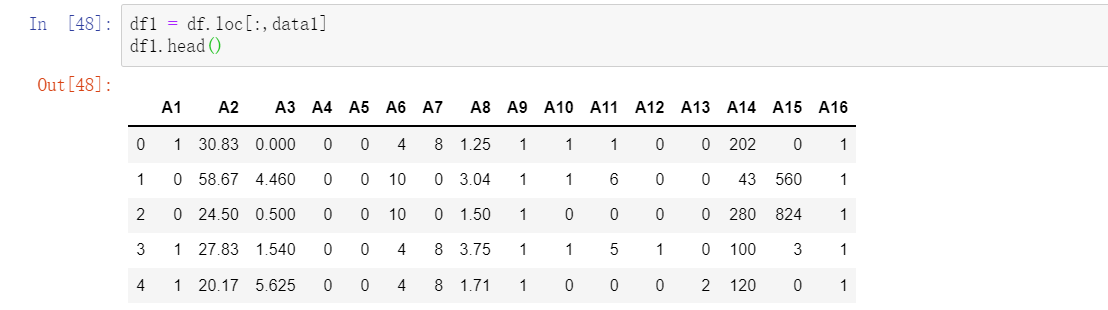
这个A6单单就有14种了，我们一个个去替换有点浪费时间

所以我们直接写一个新的函数



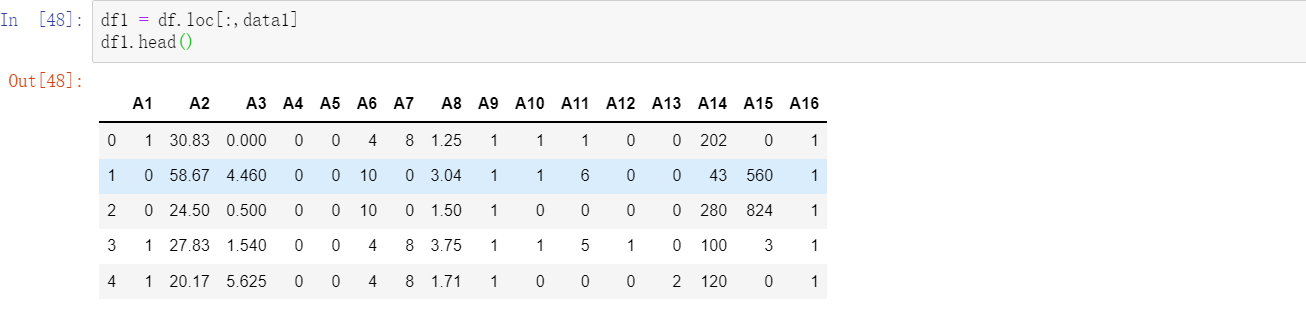
去把这些非数值类型转换为数值类型

最后处理的结果如下



在这里我们可以看到，所有非数值的类型都被我们转换为数字了，也方面我们后续对其进行处理

（5）对数值做必要的归一化

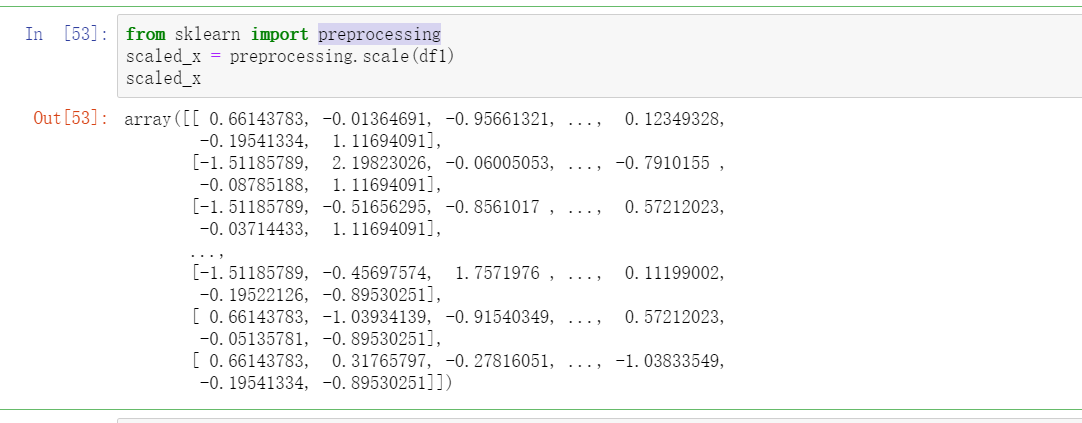


在上面这里我们可以看的很清楚，虽然说都被处理成数字了，但是每个数字背后代表的含义都不同，我们要怎么对其进行归一化处理呢，针对这种，最好的方法就是用Z-score规范化的方法去做了

具体公式如下

新数值 = （原数值-均值）/标准差

这里我们直接调用sklearn中的preprocessing去进行处理



4．分析与讨论

（1）填充缺失值可以采取哪些策略？

针对缺失值，我们可以采取三种策略，一种是用中位数来填补缺失值，一种是用众数来填补缺失值，还有一种就是直接把缺失值所在的那一行全部删除，不同的策略对应不同的环境，比如上面的在面对缺少值的时候，众数把缺失值填补好一点，如果出现最大和最小差距较大的时候，可以用中位数把缺失值给填补，如果数据集量很大，几万条以上，那么我们就可以完全把缺失值那一行给去掉，这样实际上并不会影响整体的分析情况

（2）何时需要对数据做规范化？本实验数据，适合哪种规范化方法？

当我们需要对数据进行探索和相关性分析的时候，我们就需要提前把数据进行规范化，其目的就是保证数据的整洁性，使得后面处理的结果正常，有时候数据转换比算法选择更重要，数据错了，算法再正确也是错的，在本实验数据中，因为每个数据的差异较大，所代表的含义也不同，最适合Z-scroe规范化处理，使得所有的数据都用相同的标准来进行比较，从而得知正确的分析结果

5．附录

# 非数值列处理函数

def handel\_non\_numerical\_data(df,name): #----------------name是需要处理的列名称(str)，暂不考虑列表

nrows = len(df[name]) #----------------数据集的行数

old\_col = df.columns.tolist() #----------------初始的列名集合

name\_index = old\_col.index(name) #---------要处理的列的在数据集中的索引值

name\_data = df[name].values.tolist()#-----------将要处理列复制成一个列表

# df.drop([name],axis =1,inplace =True)

unique\_kinds = set(name\_data)

convert\_dict = {}; x = 0 #构造对应种类数值转化字典

for i in unique\_kinds:

convert\_dict[i] = x

x += 1

def convert(val):

return convert\_dict[val]

name\_data = list(map(convert,name\_data))#利用map函数直接迭代转化

new\_col = df.columns.tolist()

new\_col.insert(name\_index,name)

df.reindex(columns = new\_col) #----------------重构数据的列

df[name] = name\_data