Pandas-2

2020/6/24 10:10

【数据预处理】

1.合并数据

1.1 **横向合并数据**：

pd.concat([df1,df2,df3],**axis=1**,join='inner',ignore_index=False,keys=None, levels=None,names=None,verify_integrity=False) # 当axis=1时，concat作行合并，**join参数为inner（交集）和outer（并集）**,默认为outer ignore_index，keys，levels，names，verify_integrity这几个参数不常用，设为默认即可

1.2 纵向合并数据

# 使用concat合并 pd.concat([df1,df2,df3],**axis=0**,join='inner',ignore_index=False,keys=None, levels=None,names=None,verify_integrity=False) # **使用append合并，前提条件是两张表的列名需完全一致**
df1.append(df2,ignore_index=False,verify_integrity=False)

1.3 主键合并数据 **（表联接）**

使用merge合并： **pd.merge(left,right,how='inner'**,on=None,left_on=None,right_on=None, left_index=False,right_index=False,sort=False, suffixes='_x','_y'),copy=True,indicator=False) # left和right为联接的表1和表2，**how为联接的方式**（inner,outer,right,left),默认inner left_on为表1的主键，right为表2的主键，left_index为是否将表1的index作为主键，默认False。 **sort为是否根据连接键对合并后的数据进行排序**，默认False，suffixes为合并后数据列名相同的后缀，默认('_x','_y')。 # **使用join进行合并，前提是俩张表的主键的名字必须相同** df1.join(df2,on=None,how='inner',lsuffix='',rsuffix='',sort=None) # lsuffix为合并后左侧重叠列名的后缀，rsuffix为合并后右侧重叠列名的后缀

1.4 重叠合并数据：

俩张表的内容几乎一致，但某些特征在一张表上是完整的，但在另外一张表上是缺失的
df1.combine_first(df2)

2. 清洗数据（重复值，缺失值，异常值）的处理：

2.1重复值的检测及处理：

 # 判断某个字段是否有重复值 len(df.col1.unique()) #将返回值与len(df.col1)进行比较 # 记录重复处理： df.drop_duplicates(subset=['col1','col2'],keep='first',inplace=False) # subset为需要去重复的列，keep参数有first（保留第一个），last（保留最后一个）， false（只要有重复都不保留） inplace为是否在源数据上操作，默认False

2.2 缺失值的检测及处理

 # 判断字段是否有缺失 df.isnull().sum() 或 df.notnull().sum() # 缺失值处理--删除： df.dropna(axis=1,how='any',inplace=False) # axis为1是删除列，为0时删除行 how参数为any（只要有缺失值存在就删除），all（全部为缺失值时才删除，默认为any inplace为是否在源数据上操作，默认为False **# 缺失值处理--替换法 # 替换数值型字段时，常用平均数，中位数，替换类别性字段时，常用众数 df.fillna(value=None,method=None,axis=1,inplace=False,limit=None) # method参数为ffill（用上一个非缺失值填充），bfill（用下一个非缺失值来填充）** # 缺失值处理--插值法 # 线性插值： from scipy.interpolate import interplt x=np.array(df['col1']) y=np.array(df['col2']) linearInsValue = interplt(x,y,kind='linear') linearInsValue([6,7]) **# 拉格朗日插值： from scipy.interpolate import lagrange** x=np.array(df['col1']) y=np.array(df['col2']) largeInsValue = lagrange(x,y) largeInsValue([6,7]) # 样条插值： from scipy.interpolate import spline x=np.array(df['col1']) y=np.array(df['col2']) splineInsValue = spline(x,y,xnew=np.array([6,7])) # 线性插值法需要x与y存在线性关系，效果才好，大多数情况下，用拉格朗日法和样条插值法较好

2.3 异常值的检测及处理

使用3σ原则识别异常值，不过该原则只对正态分布或近似正态分布的数据有效 def outRange(ser1): boolInd = (ser1.mean()-3*ser1.std()>ser1)|(ser1.mean()+3*ser1.std()<ser1) index = np.arange(ser1.shape[0])[boolInd] outRange = ser1.iloc[index] return outRange outlier = outRange(df['col1']) len(outlier) # 用箱线图来分析异常值 import matplotlib.pyplot as plt plt.figure(figsize=(10,7)) p = plt.boxplot(df['col1'].values,notch=True) outlier = p['fliers'][0].get_ydata() plt.show() len(outlier)

3. 数据标准化

3.1 离差标准化数据（区间缩放）

自定义离差标准化函数 def MinMaxScale(data): data = (data-data.min())/(data-data.max()) return data df['col1_scale'] = MinMaxScale(df['col1'])

3.2 标准差标准化数据

自定义标准差标准化函数 def StandardScale(data): data = (data-data.mean())/data.std() return data df['col1_scale'] = StandardScale(df['col1'])

3.3 小数定标标准化数据：通过移动数据的小数位数，将数据映射到区间【-1,1】

自定义小数定标标准化数据 def DecimalScale(data): data = data/10**np.ceil(np.log10(data.abs().max())) return data df['col1_scale'] = DecimalScale(df['col1'])

离差标准化缺点为若数据集中某个数据值很大，则离差标准化的值就会趋近于0，并且相互之间差别不大。但方法简单，便于理解。标准差标准化受数据分布的影响较小，小数定标标准化的适用范围更广。

4.数据的转换

4.1 哑变量处理

pd.get_dummies(data,prefix=None,prefix_sep='_',dummmy_na=False,columns=None) # prefix为哑变量后列名的前缀，默认为None dummy_na为是否需要为NaN值添加一列，默认None columns默认为None，表示对所有的object和category类型的字段进行处理

4.2 连续型数据离散化（分箱处理）

等宽法：对数据分布要求较高，若数据分布非常不均匀，这种方法不合适 pd.cut(df['col1'],bins=5,labels='',precision=3) # bins为离散化后的类别数目，labels为离散化后各个类别的名称，默认为空 precision是标签精度 默认为3 # 等频法： def SameRateCut(data,k): w = data.quantile(np.arange(0,1+1.0/k,1.0/k)) data = pd.cut(data,w) return data result = SameRateCut(df['col1'],5).value_counts()