西安交通大学实验报告

课程 **Python数据处理** 实验名称 **使用pandas进行数据预处理**  共 **26** 页

系 别 **电信学部** 实 验 日 期 **2023**年**4**月**23**日

专业班级 实 验 报 告 日 期 **2023**年**4**月**23**日

姓 名 学号 报 告 退 发( 订正 、 重做 )

同 组 人\_\_\_无\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 教 师 审 批 签 字

*本次实验合计约使用****3****个小时*

# **实验目的**

1. **学习如何使用Pandas和NumPy库从原始数据中提取、处理和标准化数据。**
2. **探究三种常用的数据标准化方法（标准差标准化、离差标准化和小数定标标准化）的实现原理和优缺点。**
3. **了解如何将数据标准化应用于现实世界中的数据集以提高数据的可比性和可分析性。**
4. **通过此实验，能够熟练使用Python编写函数对数据进行批量处理，提高实验效率和数据处理能力。**

# **实验内容**

## 实验题目一：（35分）

**使用Pandas完成以下实验题目，数据集在思源学堂中下载。**

**1）读取A、B和C三个用户用电量数据missing\_data.csv；**

**2）检测缺失值的分布情况；**

**3）对缺失值采用删除法并显示处理后的结果（注意：不对原表上操作）；**

**4）对缺失值采用均值替换法并显示处理后的结果（注意：不对原表上操作）；**

**5）对缺失值采用拉格朗日插值法并显示处理后的结果（注意：不对原表上操作）；**

**6）对缺失值处理后的结果采用等宽离散化并显示分布情况；**

**7）对缺失值处理后的结果采用等频离散化并显示分布情况。**

#### 程序源代码：

**import pandas as pd**

**import numpy as np**

**from scipy.interpolate import lagrange**

**origin\_data = pd.read\_csv('missing\_data.csv', names=['用户A', '用户B', '用户C'])**

**data = origin\_data.copy(deep=True)**

**print('A、B、C三个用户用电量数据：')**

**print(data)**

**print()**

**print('检测缺失值的分布情况')**

**print('isnull函数运行得到的布尔矩阵：')**

**print(data.isnull())**

**print('统计每个用户缺失数据的数目为：')**

**print(data.isnull().sum())**

**print()**

**print('对缺失值采用删除法并显示处理后的结果')**

**data\_del = data.dropna(axis=0, how='any', inplace=False)**

**print('处理后的结果：')**

**print(data\_del)**

**print('去除包含缺失值的行之前，数据的形状为：', data.shape)**

**print('去除包含缺失值的行之后，数据的形状为：', data\_del.shape)**

**print()**

**data\_mean = origin\_data.copy(deep=True)**

**print('对缺失值采用均值替换法并显示处理后的结果')**

**for user in data\_mean.columns:**

**mean = data\_mean[user].mean()**

**print(f'{user}用电量数据的均值：{mean}')**

**data\_mean[user].fillna(value=mean, inplace=True)**

**print('均值替换法处理后的结果：')**

**print(data\_mean)**

**print()**

**print('对缺失值采用拉格朗日插值法并显示处理后的结果')**

**data\_lag = origin\_data.copy(deep=True)**

**for user in data\_lag.columns:**

**data\_lag\_notnull = data\_lag.loc[data\_lag[user].notnull(), user]**

**x\_notnull = data\_lag\_notnull.index**

**y\_notnull = data\_lag\_notnull.values**

**x\_null = data\_lag.loc[data\_lag[user].isnull(), user].index**

**lagrangeValue = lagrange(x\_notnull, y\_notnull)**

**y\_null\_afterlag = lagrangeValue(x\_null)**

**i = 0**

**for index in x\_null:**

**data\_lag.loc[index, user] = y\_null\_afterlag[i]**

**i += 1**

**print('处理后结果：')**

**print(data\_lag)**

**print()**

**k = eval(input("请输入离散化类别数目：\n"))**

**def printTable\_w(data, user):**

**print(f'{user}的数据分布情况：')**

**interval = pd.cut(data[user], k)**

**print(interval.value\_counts())**

**print('数据最小值：', min(data[user]))**

**print('数据最大值：', max(data[user]))**

**print('用电量数据', k, '等分值：', (max(data[user]) - min(data[user])) / k)**

**print()**

**print('对缺失值处理后的结果采用等宽离散化并显示分布情况')**

**for user in data.columns:**

**print()**

**print(f'对{user}采用删除法之后，等宽离散化处理的结果：')**

**printTable\_w(data\_del, user)**

**for user in data.columns:**

**print()**

**print(f'对{user}采用均值替换法之后，等宽离散化处理的结果：')**

**printTable\_w(data\_mean, user)**

**for user in data.columns:**

**print()**

**print(f'对{user}采用拉格朗日插值法之后，等宽离散化处理的结果：')**

**printTable\_w(data\_lag, user)**

**def printTableF(data, user):**

**print(f'{user}的数据分布情况：')**

**interval = pd.qcut(data[user], k)**

**print(interval.value\_counts())**

**print('数据最小值：', min(data[user]))**

**print('数据最大值：', max(data[user]))**

**print()**

**print('对缺失值处理后的结果采用等频离散化并显示分布情况')**

**for user in data.columns:**

**print()**

**print(f'对{user}采用删除法之后，等频离散化处理的结果：')**

**printTableF(data\_del, user)**

**for user in data.columns:**

**print()**

**print(f'对{user}采用均值替换法之后，等频离散化处理的结果：')**

**printTableF(data\_mean, user)**

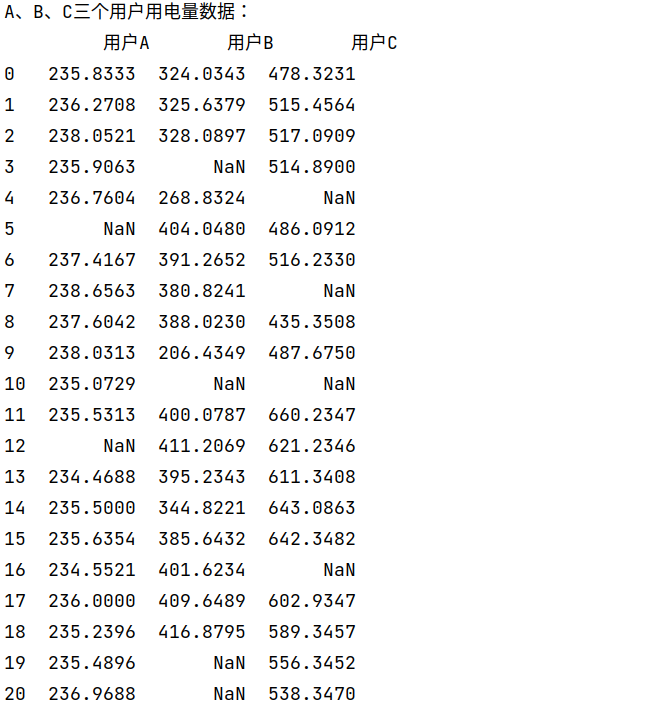
**for user in data.columns:**

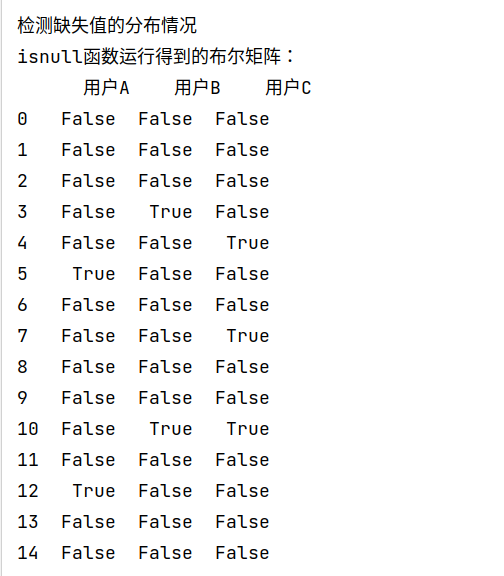
**print()**

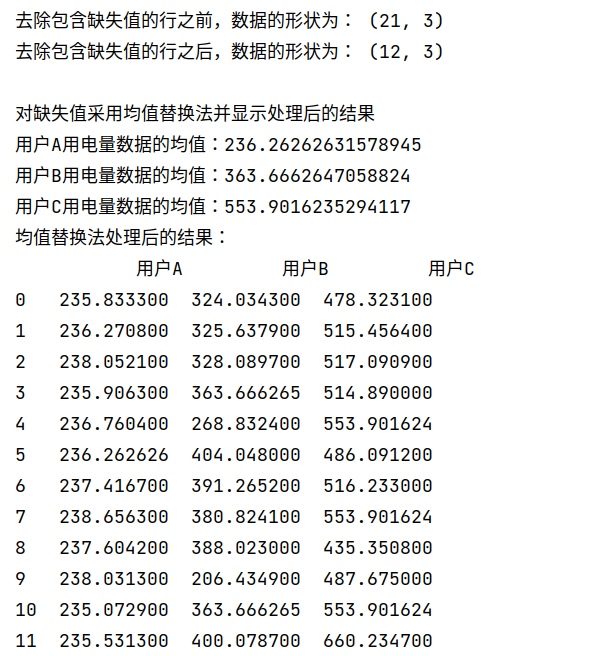
**print(f'对{user}采用拉格朗日插值法之后，等频离散化处理的结果：')**

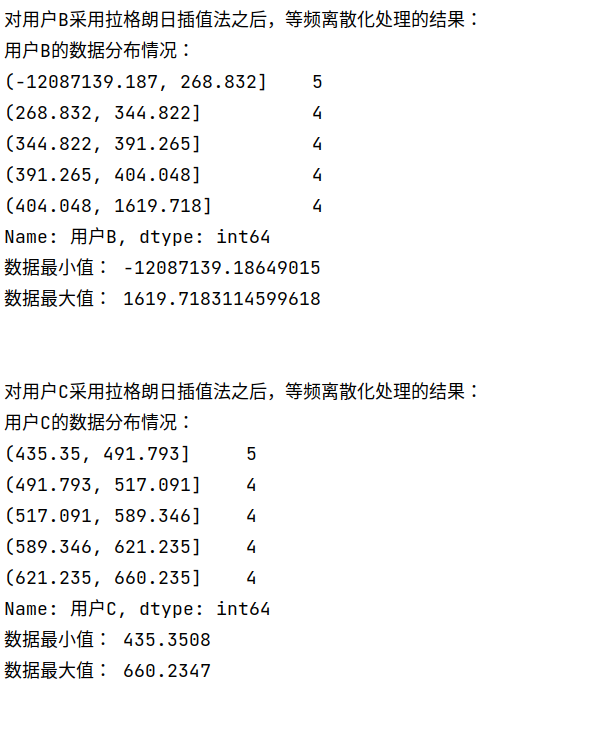
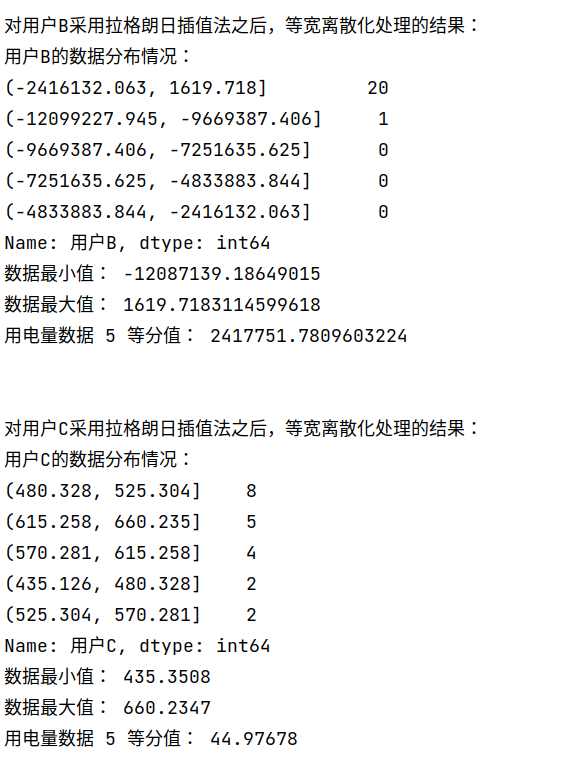
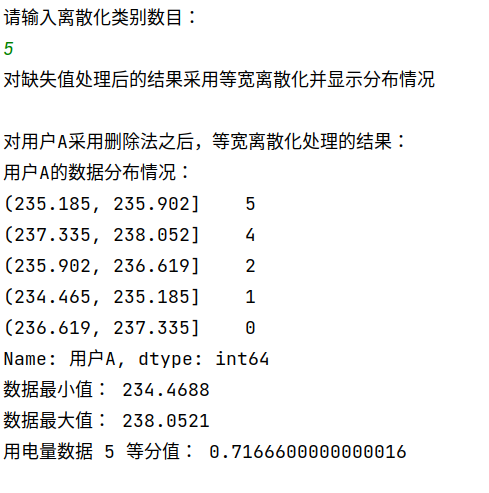
**printTableF(data\_lag, user)**

#### 运行结果：









## 实验题目二：（35分）

**使用Pandas完成以下实验题目，数据集在思源学堂中下载。**

**1）读取线路数据ele\_loss.csv（包含线路编号ID，时间date，用电量ele，线损loss）和线路告警数据alarm.csv（包含线路编号ID，时间date，报警提示alarm）**

**2）将 ele\_loss.csv 和 alarm.csv 横向堆叠内连接构成table1并查看合并后的数据；**

**3）对 table1进行特征去重并显示去重前后的列数；**

**4）将 ele\_loss.csv 和 alarm.csv 纵向堆叠内连接构成table12并查看合并后的数据；**

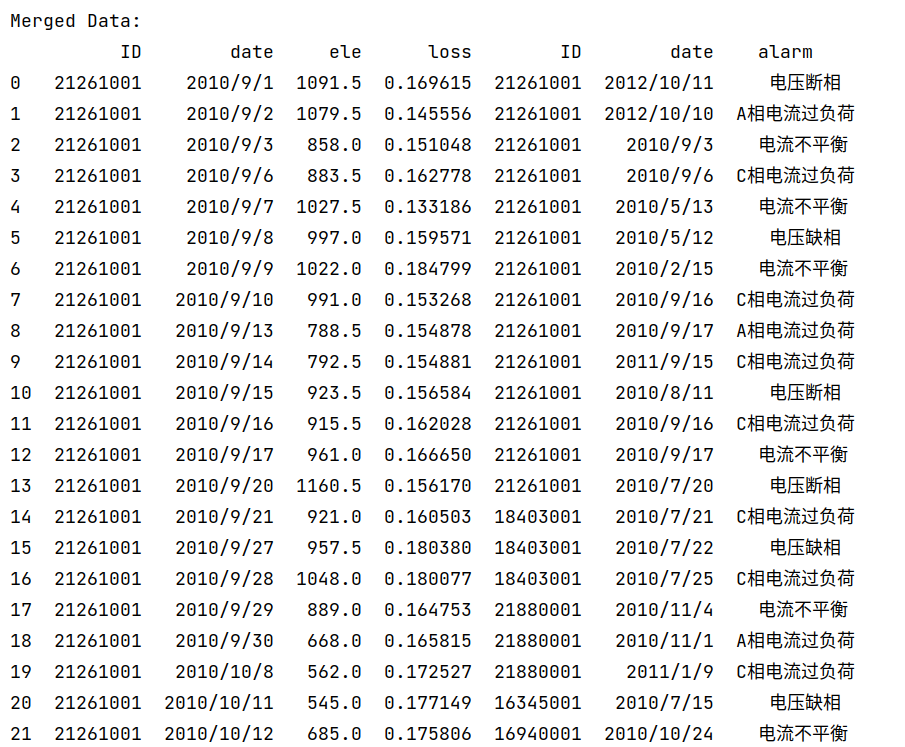
**5）对 table2进行记录去重并显示去重前后的行数；**

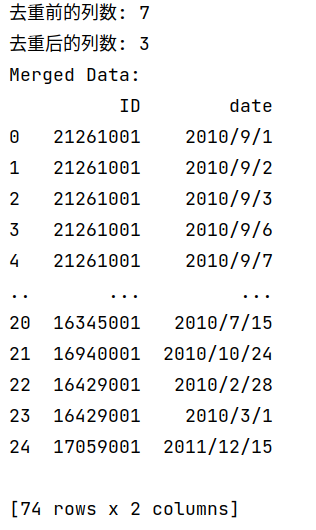
**6）以ID和date为主键对 ele\_loss.csv 和 alarm.csv 进行内连接构成table3并查看合并后的数据；**

**7）对tables的ele和loss使用拉依达准则判断异常值的个数以及异常值的最大和最小值。**

程序源代码 **import pandas as pd  
import numpy as np  
# read the datasets  
  
el = pd.read\_csv('ele\_loss.csv', encoding='gbk')  
a = pd.read\_csv('alarm.csv', encoding='gbk')  
# horizontally stack the datasets and check the merged data  
  
table1 = pd.concat([el, a], axis=1, join='inner')  
print("Merged Data:")  
print(table1)  
# feature deduplication  
  
print("去重前的列数:", table1.shape[1])  
# function to compare feature equality  
  
def FeatureEquals(df):  
 dfEquals = pd.DataFrame([], columns=df.columns, index=df.columns)  
 for i in df.columns:  
 for j in df.columns:  
 dfEquals.loc[i, j] = df.loc[:, i].equals(df.loc[:, j])  
 return dfEquals  
  
detEquals = FeatureEquals(table1)  
# check and delete duplicates  
  
lenDet = detEquals.shape[0]  
dupCol = []  
for k in range(lenDet):  
 for l in range(k + 1, lenDet):  
 if detEquals.iloc[k, l] & (detEquals.columns[l] not in dupCol):  
 dupCol.append(detEquals.columns[l])  
table1.drop(dupCol, axis=1, inplace=True)  
print('去重后的列数:', table1.shape[1])  
# vertically stack the datasets and check the merged data  
  
table2 = pd.concat([el, a], axis=0, join='inner')  
print("Merged Data:")  
print(table2)  
# record deduplication  
  
print("去重前行数为:", table2.shape[0])  
print("去重后行数为:", table2.drop\_duplicates().shape[0])  
# join the datasets based on ID and date  
  
table3 = pd.merge(el, a, how='inner', on=['ID', 'date'])  
print("Merged Data:")  
print(table3)  
# Outlier Detection using percentile-based method  
  
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  
  
ele = outRange(table3['ele'])  
print('使用拉依达准则判定ele异常值个数为:', ele.shape[0])  
print('ele异常值的最大值为:', ele.max())  
print('ele异常值的最小值为:', ele.min())  
  
loss = outRange(table3['loss'])  
print('使用拉依达准则判定loss异常值个数为:', loss.shape[0])  
print('异常值的最大值为:', loss.max())  
print('异常值的最小值为:', loss.min())**

#### 运行结果：







## 实验题目三：（30分）

**使用Pandas完成以下实验题目，数据集在思源学堂中下载。**

**1）读取model.csv；**

**2）对前三列数据进行标准差标准化并查看标准化后的数据；**

**3）对前三列数据进行离差标准化并查看标准化后的数据；**

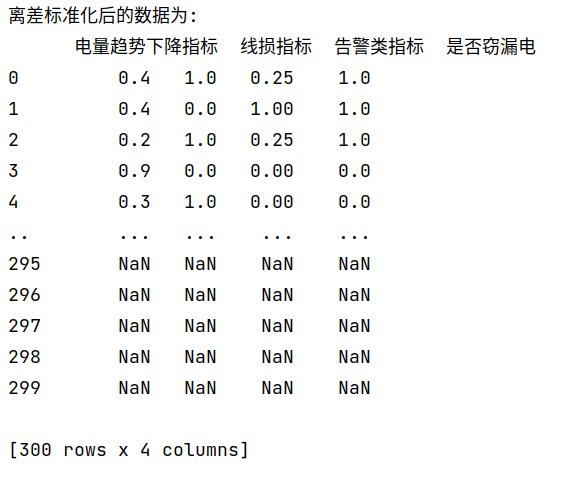
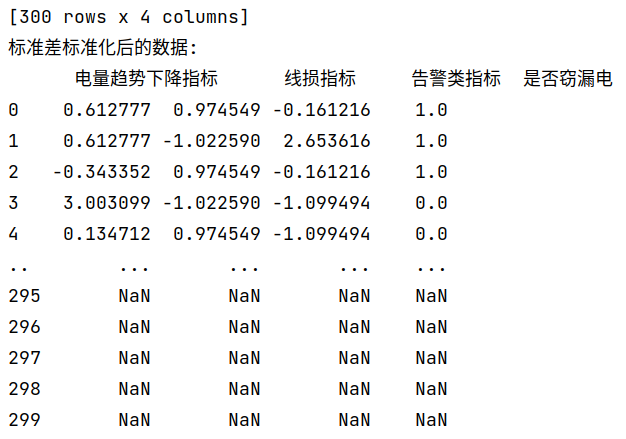
**4）对前三列数据进行小数定标标准化并查看标准化后的数据；**

#### 程序源代码：

**import pandas as pd  
import numpy as np  
  
model = pd.read\_csv("model.csv", sep=',', encoding='gbk')  
print(model)  
  
def standard(data):  
 return (data - data.mean()) / data.std()  
  
  
model\_s\_0 = standard(model.iloc[:, 0])  
model\_s\_1 = standard(model.iloc[:, 1])  
model\_s\_2 = standard(model.iloc[:, 2])  
model\_3 = model.iloc[:, 3]  
model\_s = pd.concat([model\_s\_0, model\_s\_1, model\_s\_2, model\_3], axis='columns')  
print("标准差标准化后的数据:\n", model\_s)  
  
def myStandard(data):  
 return (data - data.min()) / (data.max() - data.min())  
  
model\_myStandard\_0 = myStandard(model.iloc[:, 0])  
model\_myStandard\_1 = myStandard(model.iloc[:, 1])  
model\_myStandard\_2 = myStandard(model.iloc[:, 2])  
model\_3 = model.iloc[:, 3]  
# 把各行结果连接起来  
model\_myStandard = pd.concat([model\_myStandard\_0, model\_myStandard\_1, model\_myStandard\_2, model\_3], axis='columns')  
print("离差标准化后的数据为:\n", model\_myStandard)**

#### 运行结果：





# **实验小结**

**在本次实验中，我们使用了Python中的Pandas和NumPy库来处理和标准化给定的数据集。通过读取和可视化原始数据集，我们了解了数据的形式和特点。然后，我们分别使用三种数据标准化方法（标准差标准化、离差标准化和小数定标标准化）来提高数据的可比性和可分析性。我们编写了Python函数实现每种方法，然后应用在数据集的特定列中。最终，我们把标准化后的数据与原始数据连接起来，比较了三种标准化处理对数据的影响。通过本次实验，我们掌握了在Python中完成数据预处理和分析的基本方法，对我们进一步分析和理解数据具有重要意义。**