数据挖掘 Lab1 Data preprocessing

--10215501435 杨茜雅

必要的代码步骤已经在 ipynb 文件中展现，这里阐述一些必要的文字说明

1、 数据质量问题（数据：breast-cancer-wisconsin.data）

1. 缺失值
- 缺失值在数据集中编码为“?”，将缺失值转换为NaN，并计算每列数据中缺失值的数量
  data.replace('?',np.NaN)
- 将缺失值替换为该列的中值：fillna()
- 丢弃包含缺失值的数据点：dropna()

起初我在疑惑第二部已经将缺失值替换为中值了，那么按理来说数据集中将没有缺失值，为什么第三步还要丢弃包含缺失值的数据点？后来得知每一个步骤都是独立在原始数据集上操作的，所以在实验过程中，每一步都新建了一个数据副本，避免下一步造成影响。

1.1 缺失值

16 个缺失值都在 Bare Nuclei 列，它的数据类型是 object，所以在下一步以中值替换缺失值之前，需要把它转换成数值型，不然填充完还是缺 16 个。

```
每列的缺失值数量：
 Sample code                    0
Clump Thickness                 0
Uniformity of Cell Size         0
Uniformity of Cell Shape        0
Marginal Adhesion               0
Single Epithelial Cell Size     0
Bare Nuclei                    16
Bland Chromatin                 0
Normal Nucleoli                 0
Mitoses                         0
Class                           0
dtype: int64
```

```
float64
在data_for_median副本上填充缺失值后的缺失值统计：
 Sample code                    0
Clump Thickness                 0
Uniformity of Cell Size         0
Uniformity of Cell Shape        0
Marginal Adhesion               0
Single Epithelial Cell Size     0
Bare Nuclei                     0
Bland Chromatin                 0
Normal Nucleoli                 0
Mitoses                         0
Class                           0
dtype: int64
```

（图片为填充前 vs 填充后）

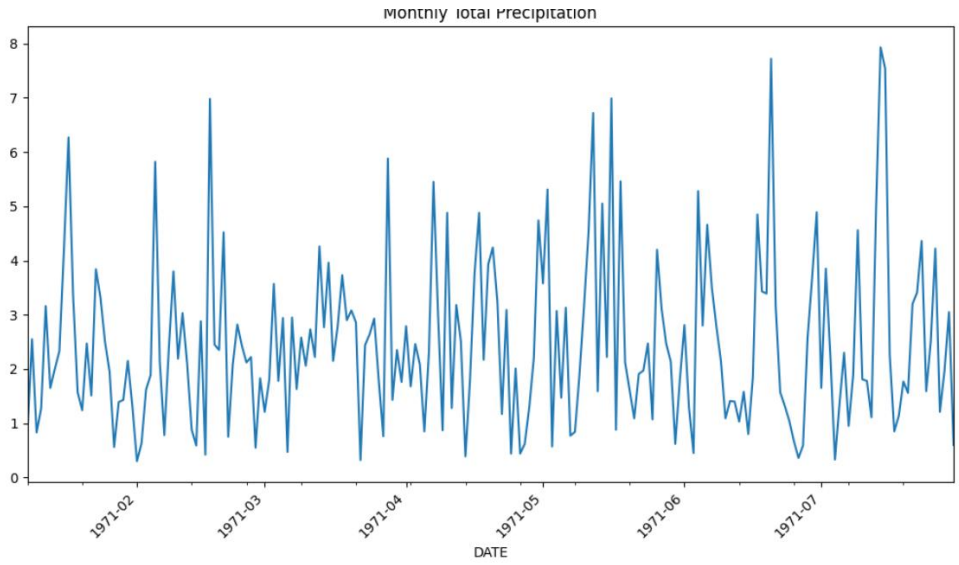
2、 数据聚合（DTW_prec.csv）

按月聚合:

之前画图的时候横坐标一直很混乱，不简洁，理论上来说按月份聚合的话横坐

标就应该是 1-12 个月份这么简单。后来打印一下输出，发现是我按月份聚合的逻辑写错了，因为按年份是 annual_precip = df['PRCP'].resample('YE').sum()，我就想当然认为按月份就是 annual_precip = df['PRCP'].resample('ME').sum()，但其是 monthly_precip_grouped = df['PRCP'].groupby(df.index.month).sum()

错误图片与输出：

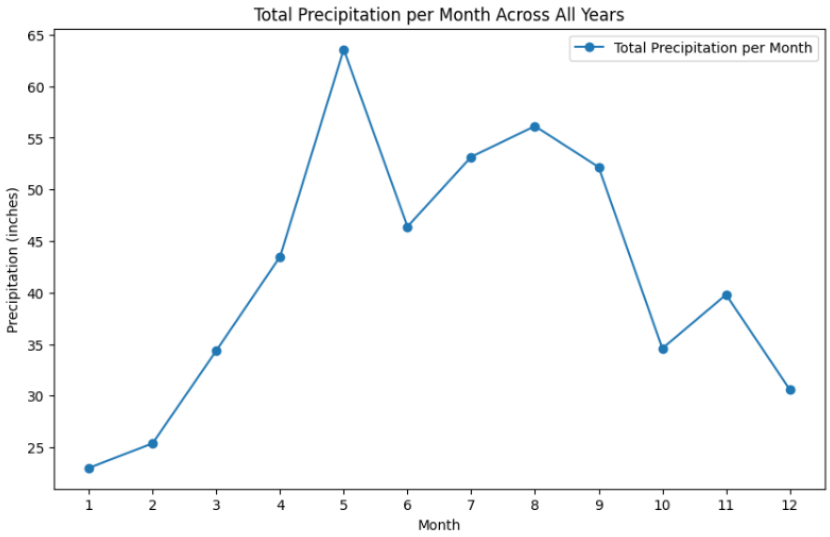


每月降水量的方差： 2.4241160509031197

```
# 保证之前的代码已经执行过，df已经按照时间索引排序
import matplotlib.dates as mdates
# 获取每月末的降水量总和
monthly_precip = df['PRCP'].resample('ME').sum()
# 打印每个月
print(monthly_precip)
```

```
DATE
2001-01-31    0.84
2001-02-28    2.55
2001-03-31    0.83
2001-04-30    1.27
2001-05-31    3.16
              ...
2017-08-31    4.22
2017-09-30    1.21
2017-10-31    1.96
2017-11-30    3.05
2017-12-31    0.60
Freq: ME, Name: PRCP, Length: 204, dtype: float64
```

正确图片与输出：



数据的方差： 163.5337356060606

```
DATE
1     23.01
2     25.39
3     34.38
4     43.48
5     63.57
6     46.40
7     53.16
8     56.14
9     52.17
10    34.59
11    39.80
12    30.60
Name: PRCP, dtype: float64
```

4、 离散化（数据：breast-cancer-wisconsin.data）

回答这个问题：为什么第一个 bin 区间的左端点是 0.991？如果想要将左端点变成 0.99，怎么改？

```
# equal width：应用cut()将属性离散为4个间隔宽度相似的bin
# 使用value_counts()确定每个bin中的实例数
# Equal width bins
clump_thickness_equal_width = pd.cut(clump_thickness, bins=4)
equal_width_counts = clump_thickness_equal_width.value_counts()
print("Equal width bins的实例数:")
print(equal_width_counts)
```

```
Equal width bins的实例数:
1
(0.991, 3.25]    303
(3.25, 5.5]      210
(7.75, 10.0]     129
(5.5, 7.75]       57
Name: count, dtype: int64
```

回答：pandas.cut 函数默认情况下会找到数据中的最小值和最大值，并在这个范围内创建等宽的箱子。由于它是根据数据自动计算边界的，所以有时候这些边界的数字会有很多小数位。要控制 pandas.cut 函数生成的区间的精度，可以使用 precision 参数来指定小数点后的位数，设置 precision=2 即可

```
# equal width：应用cut()将属性离散为4个间隔宽度相似的bin
# 使用value_counts()确定每个bin中的实例数
# Equal width bins
clump_thickness_equal_width = pd.cut(clump_thickness, bins=4, precision=2)
equal_width_counts = clump_thickness_equal_width.value_counts()
print("Equal width bins的实例数:")
print(equal_width_counts)
```

```
Equal width bins的实例数:
1
(0.99, 3.25]    303
(3.25, 5.5]     210
(7.75, 10.0]    129
(5.5, 7.75]      57
Name: count, dtype: int64
```

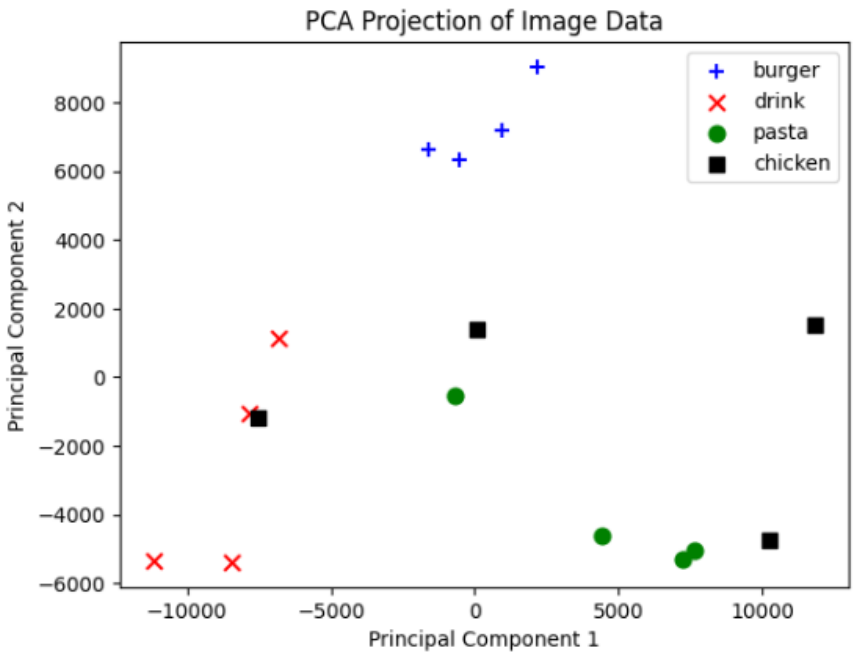
5、 主成分分析 PCA（数据：pics 文件夹 16 张四类食物图片）

多添加了一个确认矩阵维度和特征值个数的步骤

```
# 将图像数据列表转换为numpy数组
image_data_np = np.array(image_data)

# 打印数组的shape属性以确认其维度
print("矩阵的维度:", image_data_np.shape)

# 确认每张图片转换后的特征值数量
feature_count = 111 * 111 * 3  # 对于RGB图像，每个像素包含3个颜色通道
print("每张图片的特征值数量:", feature_count)

# 确认矩阵大小
if image_data_np.shape[1] == feature_count:
    print("确认：每张图像转换后的特征值数量正确。")
else:
    print("特征值数量不匹配。")

if image_data_np.shape == (16, feature_count):
    print("确认：矩阵大小为16x36963。")
else:
    print("矩阵大小不匹配。")
```

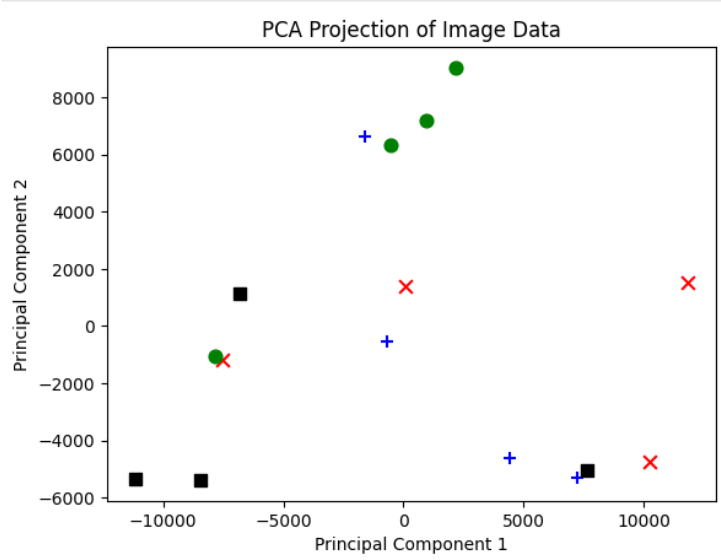
```
矩阵的维度： (16, 36963)
每张图片的特征值数量： 36963
确认：每张图像转换后的特征值数量正确。
确认：矩阵大小为16x36963。
```

图片是 16 张四类，所以画图的时候用不同形状也颜色区分它们，方便观察不同类数据的分布。我们可以发现：汉堡、饮料和意大利面的图像基本都投影到同一区域了，炸鸡(黑色方块)比较分散。



起初我画出来的图是这样的：和正确的图分布是一样，但是数据类型不一样!

由于分布是一样的，所以不可能是 PCA 过程或者转换为特征值矩阵的时候出错，**最大的可能就是图片分类没有正好将四个同类的图片分成一组**。



经过排查原因我发现是在手动给图片分类的时候采用的分类手段错误

```
    image_data.append(pixels)
    # 图像文件1-4为汉堡，5-8为可乐，等等
    labels.append(categories[i // 4])  # 每四个图像更换类别

image_files = sorted(os.listdir(image_folder))  # 确保文件是排序的

for i, file_name in enumerate(image_files):
    with Image.open(os.path.join(image_folder, file_name)) as img:
        img = img.resize((111, 111))  # 确保图像是111x111像素
        pixels = np.array(img).flatten()  # 将图像数据转换为一维数组
        image_data.append(pixels)
        # 图像文件1-4为汉堡，5-8为可乐，等等
        labels.append(categories[i // 4])  # 每四个图像更换类别
```

有可能打乱了图片顺序，所以我更换了图片的手动分类方法。

```
# 对文件名进行排序，确保顺序是正确的
image_files.sort(key=lambda x: int(x.replace('Picture', '').replace('.jpg', '')))

for file_name in image_files:
    category_index = (int(file_name.replace('Picture', '').replace('.jpg', '')) - 1) // 4
    with Image.open(os.path.join(image_folder, file_name)) as img:
        img = img.resize((111, 111))  # 确保图像是111x111像素
        pixels = np.array(img).flatten()  # 将图像数据转换为一维数组
        image_data.append(pixels)
        labels.append(categories[category_index])  # 分配类别
```