数据挖掘作业 1 数据探索性分析与预处理

马的疝病分析

姓名：于立冬

学号：2120161072

日期：2017.4.11

1. 问题描述

疝病是描述马胃肠痛的术语，这种病不一定源自马的胃肠问题，其他问题也可能引发马疝病。所给数据集是医院检测的一些指标。

2. 数据说明

下载数据: 地址

共 368 个样本，27 个特征。关于特征的详细说明见下载链接。

3. 数据分析要求

3.1 数据可视化和摘要

数据摘要

- 对标称属性，给出每个可能取值的频数，
- 数值属性，给出最大、最小、均值、中位数、四分位数及缺失值的个数。

数据的可视化

针对数值属性，

- 绘制直方图，如 mxPH，用 qq 图检验其分布是否为正态分布。

- 绘制盒图，对离群值进行识别

3.2 数据缺失的处理

数据集中有 30%的值是缺失的，因此需要先处理数据中的缺失值。

分别使用下列四种策略对缺失值进行处理:

- 将缺失部分剔除
- 用最高频率值来填补缺失值
- 通过属性的相关关系来填补缺失值
- 通过数据对象之间的相似性来填补缺失值

处理后，可视化地对比新旧数据集。

4. 提交内容

- 分析过程的报告
- 分析程序
- 预处理后的数据集

代码解读：

使用python3，pandas，numpy，matplotlib

```
import operator
import pandas as pd
import numpy as np
import statsmodels.api as sm
import scipy.stats as stats
import matplotlib.pyplot as plt
import matplotlib
matplotlib.style.use('ggplot')
```

使用数据分析库pandas进行数据分析，使用matplotlib进行可视化处理

将数据从txt转化为excel的csv格式，方便读写

```
fp_origin = open("./data.txt", 'r')
fp_modified = open("./data.csv", 'w')
```

```
line = fp_origin.readline()
while(line):
    temp = line.strip().split()
    temp = ','.join(temp)+'\n'
    fp_modified.write(temp)
    line = fp_origin.readline()

fp_origin.close()
fp_modified.close()
```

读取csv文件，根据数据集文档进行属性赋值

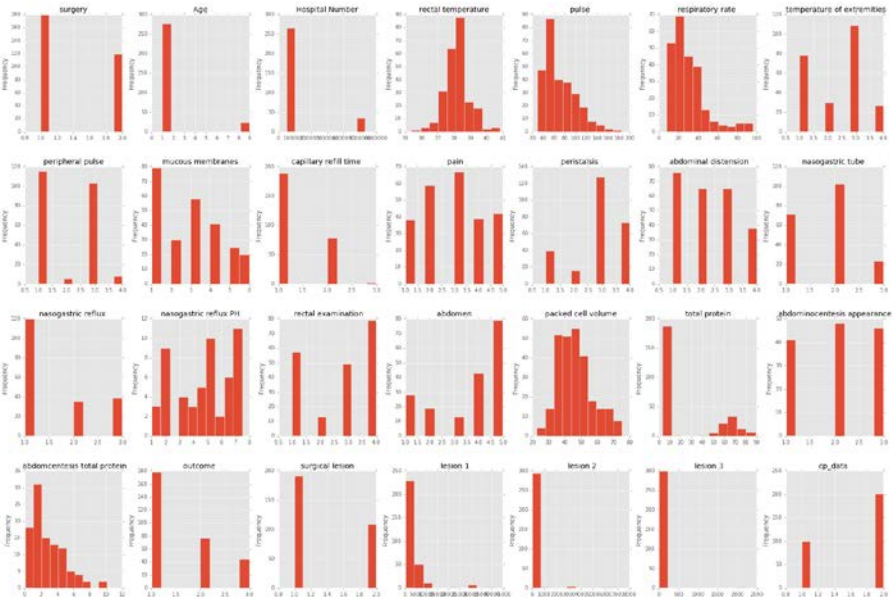
```
# 定义数据特征
attribute=["surgery"," Age ","Hospital Number","rectal temperature","pulse ","respiratory rate ","
temperature of extremities","peripheral pulse","mucous membranes","capillary refill time ","
pain","peristalsis "," abdominal distension","nasogastric tube ","nasogastric reflux "," nasogastric
reflux PH "," rectal examination"," abdomen "," packed cell volume "," total protein ","
abdominocentesis appearance "," abdomcentesis total protein","outcome ","surgical lesion","
lesion 1"," lesion 2"," lesion 3","cp_data "]



# 读取数据
data_origin = pd.read_csv("./data.csv",
                          names = attribute,
                          na_values = "?")
# 给出每个可能属性的频数

# 使用value_counts函数统计每个标称属性的取值频数
for item in attribute:
    print(item, '的频数为：\n', pd.value_counts(data_origin[item].values), '\n')
# 对数值属性，给出最大、最小、均值、中位数、四分位数及缺失值的个数。

# 最大值
data_show = pd.DataFrame(data = data_origin[attribute].max(), columns = ['max'])
# 最小值
data_show['min'] = data_origin[attribute].min()
# 均值
data_show['mean'] = data_origin[attribute].mean()
# 中位数
data_show['median'] = data_origin[attribute].median()
# 四分位数
data_show['quartile'] = data_origin[attribute].describe().loc['25%']
# 缺失值个数
data_show['missing'] = data_origin[attribute].describe().loc['count'].apply(lambda x : 200-x)

print (data_show)
```

	max	min	mean	median	\
surgery	2.0	1.0	1.397993	1.00	
Age	9.0	1.0	1.640000	1.00	
Hospital Number	5305629.0	518476.0	1085888.833333	530305.50	
rectal temperature	40.8	35.4	38.167917	38.20	
pulse	184.0	30.0	71.913043	64.00	
respiratory rate	96.0	8.0	30.417355	24.50	
temperature of extremities	4.0	1.0	2.348361	3.00	
peripheral pulse	4.0	1.0	2.017316	2.00	
mucous membranes	6.0	1.0	2.853755	3.00	
capillary refill time	3.0	1.0	1.305970	1.00	
pain	5.0	1.0	2.951020	3.00	
peristalsis	4.0	1.0	2.917969	3.00	
abdominal distension	4.0	1.0	2.266393	2.00	
nasogastric tube	3.0	1.0	1.755102	2.00	
nasogastric reflux	3.0	1.0	1.582474	1.00	
nasogastric reflux PH	7.5	1.0	4.707547	5.00	
rectal examination	4.0	1.0	2.757576	3.00	
abdomen	5.0	1.0	3.692308	4.00	
packed cell volume	75.0	23.0	46.295203	45.00	
total protein	89.0	3.3	24.456929	7.50	
abdominocentesis appearance	3.0	1.0	2.037037	2.00	
abdomcentesis total protein	10.1	0.1	3.019608	2.25	
outcome	3.0	1.0	1.551839	1.00	
surgical lesion	2.0	1.0	1.363333	1.00	
lesion 1	41110.0	0.0	3657.880000	2673.50	
lesion 2	7111.0	0.0	90.226667	0.00	
lesion 3	2209.0	0.0	7.363333	0.00	
cp_data	2.0	1.0	1.670000	2.00	

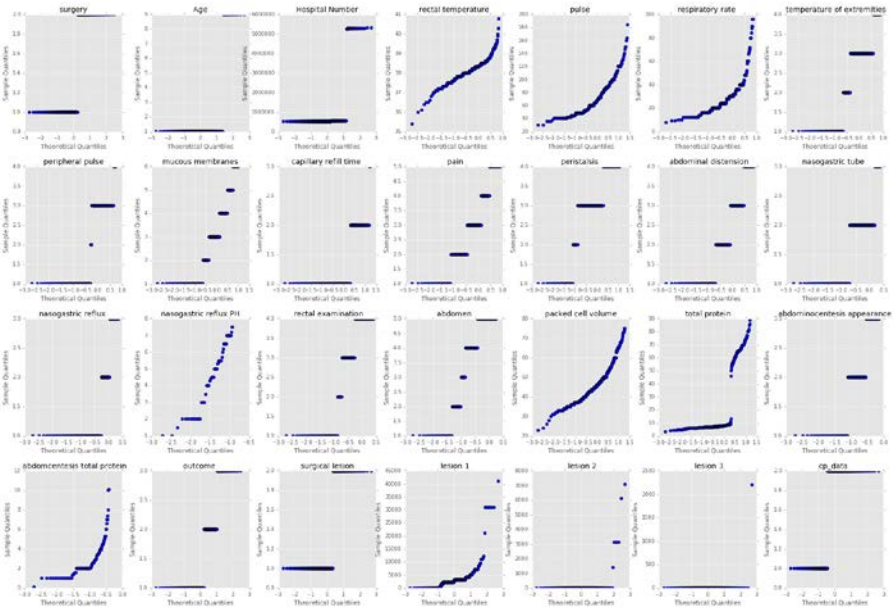
	quartile	missing
surgery	1.00	-99
Age	1.00	-100
Hospital Number	528904.00	-100
rectal temperature	37.80	-40
pulse	48.00	-76
respiratory rate	18.50	-42
temperature of extremities	1.00	-44
peripheral pulse	1.00	-31
mucous membranes	1.00	-53
capillary refill time	1.00	-68
pain	2.00	-45
peristalsis	3.00	-56
abdominal distension	1.00	-44
nasogastric tube	1.00	4
nasogastric reflux	1.00	6
nasogastric reflux PH	3.00	147
rectal examination	1.00	2
abdomen	2.00	18
packed cell volume	38.00	-71
total protein	6.50	-67
abdominocentesis appearance	1.00	65
abdomcentesis total protein	2.00	98
outcome	1.00	-99
surgical lesion	1.00	-100
lesion 1	2111.75	-100
lesion 2	0.00	-100
lesion 3	0.00	-100
cp_data	1.00	-100

绘制直方图，如mxPH，用qq图检验其分布是否为正态分布。

```
# 直方图
fig = plt.figure(figsize = (30,20))
i = 1
for item in attribute:
    ax = fig.add_subplot(4,7,i)
    data_origin[item].plot(kind = 'hist', title = item, ax = ax)
    i += 1
plt.subplots_adjust(wspace = 0.3, hspace = 0.3)
fig.savefig('./image/histogram.jpg')
print ('histogram saved at ./image/histogram.jpg')
```

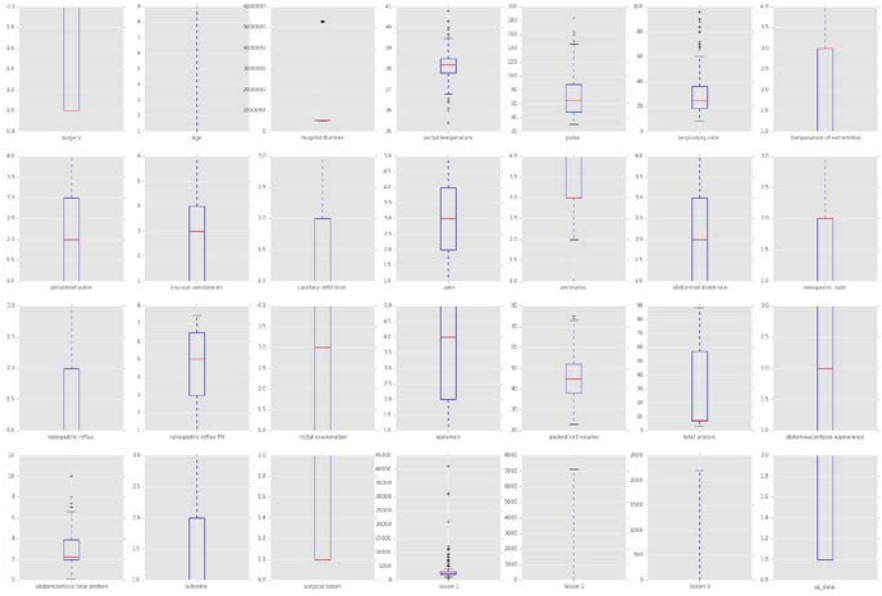



```
# qq图
fig = plt.figure(figsize = (30,20))
i = 1
for item in attribute:
    ax = fig.add_subplot(4,7,i)
    sm.qqplot(data_origin[item], ax = ax)
    ax.set_title(item)
    i += 1
plt.subplots_adjust(wspace = 0.3, hspace = 0.3)
fig.savefig('./image/qqplot.jpg')
print ('qqplot saved at ./image/qqplot.jpg')
```



可以看出属性19满足正态分布

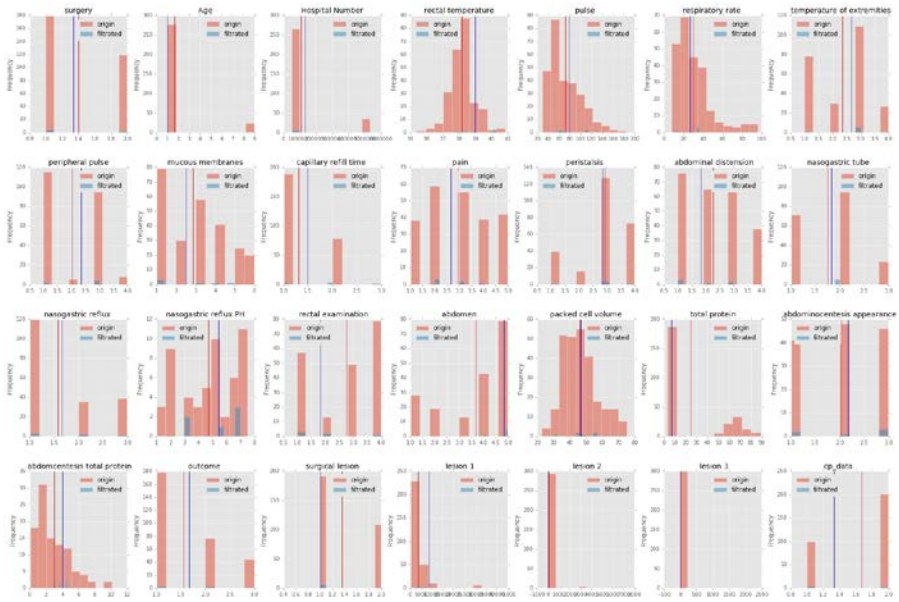
```
# 绘制盒图，对离群值进行识别。
# 盒图
fig = plt.figure(figsize = (30,20))
i = 1
for item in attribute:
    ax = fig.add_subplot(4,7,i)
    data_origin[item].plot(kind = 'box')
    i += 1
fig.savefig('./image/boxplot.jpg')
print ('boxplot saved at ./image/boxplot.jpg')
```



对于二值的属性，离群值参考性不大，多取值属性离群值可以通过盒图看出。

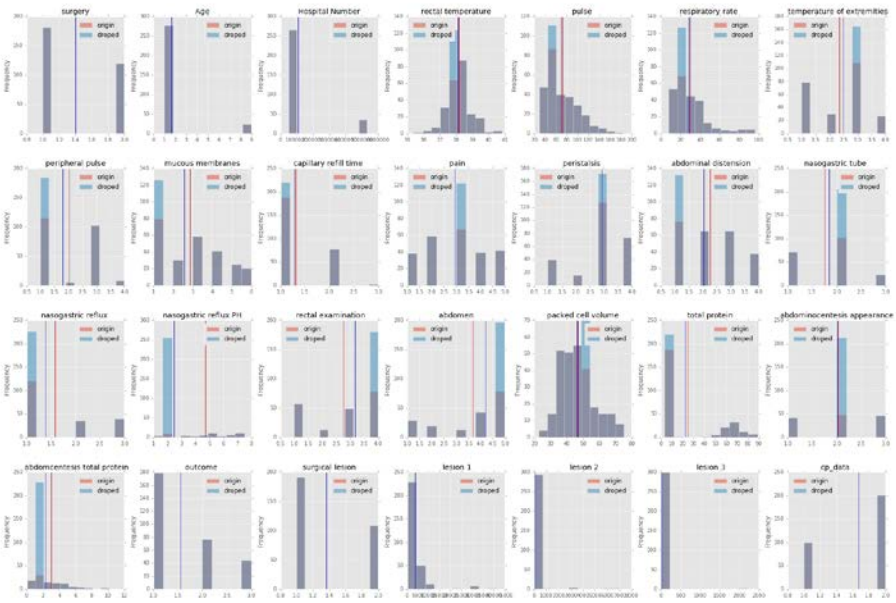
```
# 找出含有缺失值的数据条目索引值
nan_list = pd.isnull(data_origin).any(1).nonzero()[0]

# 使用dropna()函数操作删除缺失值
# 将缺失值对应的数据整条剔除，生成新数据集
data_filtrated = data_origin.dropna()

# 绘制可视化图
fig = plt.figure(figsize = (30,20))

i = 1

# 对数值属性，绘制直方图
for item in attribute:
    ax = fig.add_subplot(4,7,i)
    ax.set_title(item)
    data_origin[item].plot(ax = ax, alpha = 0.5, kind = 'hist', label = 'origin', legend = True)
    data_filtrated[item].plot(ax = ax, alpha = 0.5, kind = 'hist', label = 'filtrated', legend = True)
    ax.axvline(data_origin[item].mean(), color = 'r')
    ax.axvline(data_filtrated[item].mean(), color = 'b')
    i += 1
plt.subplots_adjust(wspace = 0.3, hspace = 0.3)
```

```
# 保存图像和处理后数据
fig.savefig('./image/missing_data_delete.jpg')
data_filtrated.to_csv('./data_output/missing_data_delete.csv', mode = 'w', encoding='utf-8',
index = False,header = False)
print ('filted_missing_data1 saved at ./image/missing_data_delete.jpg')
print ('data after analysis saved at ./data_output/missing_data_delete.csv')
```



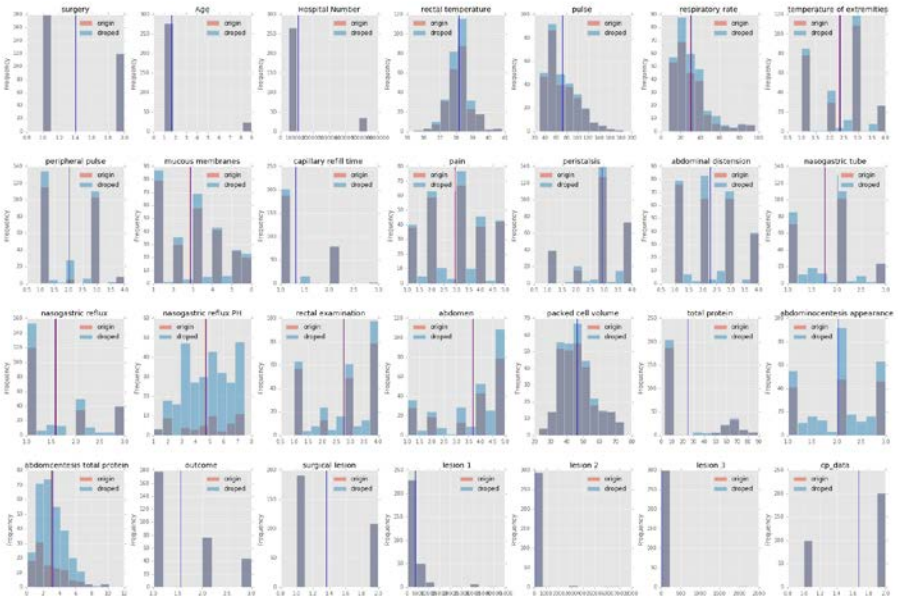

```
# 用最高频率值来填补缺失值
#
# 使用value_counts()函数统计原始数据中，出现频率最高的值，再用fillna()函数将缺失值替换为最高频率值。
# 建立原始数据的拷贝
data_filtrated = data_origin.copy()
# 对每一列数据，分别进行处理
for item in attribute:
    # 计算最高频率的值
    most_frequent_value = data_filtrated[item].value_counts().idxmax()
    # 替换缺失值
    data_filtrated[item].fillna(value = most_frequent_value, inplace = True)

# 绘制可视化图
fig = plt.figure(figsize = (30,20))

i = 1

# 对数值属性，绘制直方图
```

```
for item in attribute:
    ax = fig.add_subplot(4,7,i)
    ax.set_title(item)
    data_origin[item].plot(ax = ax, alpha = 0.5, kind = 'hist', label = 'origin', legend = True)
    data_filtrated[item].plot(ax = ax, alpha = 0.5, kind = 'hist', label = 'droped', legend = True)
    ax.axvline(data_origin[item].mean(), color = 'r')
    ax.axvline(data_filtrated[item].mean(), color = 'b')
    i += 1
plt.subplots_adjust(wspace = 0.3, hspace = 0.3)

# 保存图像和处理后数据
fig.savefig('./image/missing_data_most.jpg')
data_filtrated.to_csv('./data_output/missing_data_most.csv', mode = 'w', encoding='utf-8', index
= False,header = False)
print ('filted_missing_data2 saved at ./image/missing_data_most.jpg')
print ('data after analysis saved at ./data_output/missing_data_most.csv')
```




```
# 通过属性的相关关系来填补缺失值
# 使用pandas中Series的***interpolate()***函数，对数值属性进行插值计算，并替换缺失值。
# 从直方图中可以看出，处理后的数据，添加了若干个值不同的值，并且均值变化不大。
# 建立原始数据的拷贝
data_filtrated = data_origin.copy()
#进行插值运算
for item in attribute:
    data_filtrated[item].interpolate(inplace = True)
```

```
# 绘制可视化图
fig = plt.figure(figsize = (30,20))

i = 1

# 对数值属性，绘制直方图
for item in attribute:
    ax = fig.add_subplot(4,7,i)
    ax.set_title(item)
    data_origin[item].plot(ax = ax, alpha = 0.5, kind = 'hist', label = 'origin', legend = True)
    data_filtrated[item].plot(ax = ax, alpha = 0.5, kind = 'hist', label = 'droped', legend = True)
    ax.axvline(data_origin[item].mean(), color = 'r')
    ax.axvline(data_filtrated[item].mean(), color = 'b')
    i += 1
plt.subplots_adjust(wspace = 0.3, hspace = 0.3)

# 保存图像和处理后数据
fig.savefig('./image/missing_data_corelation.jpg')
data_filtrated.to_csv('./data_output/missing_data_corelation.csv', mode = 'w', encoding='utf-8',
index = False,header = False)
print ('filted_missing_data3 saved at ./image/missing_data_corelation.jpg')
print ('data after analysis saved at ./data_output/missing_data_corelation.csv')
```



```
# 通过数据对象之间的相似性来填补缺失值
# 首先将缺失值设为0，对数据集进行正则化。然后对每两条数据进行差异性计算（分值越高差异性越大）。计算标准为：标称数据不相同记为1分，数值数据差异性分数为数据之间的差值。在处理缺失值时，找到和该条数据对象差异性最小（分数最低）的对象，将最相似的数据条目中对应属性的值替换缺失值。
# 建立原始数据的拷贝，用于正则化处理
data_norm = data_origin.copy()
# 将数值属性的缺失值替换为0
data_norm[attribute] = data_norm[attribute].fillna(0)
# 对数据进行正则化
data_norm[attribute] = data_norm[attribute].apply(lambda x : (x - np.mean(x)) / (np.max(x) - np.min(x)))
# 构造分数表
score = {}
range_length = len(data_origin)
for i in range(0, range_length):
    score[i] = {}
    for j in range(0, range_length):
        score[i][j] = 0

# 在处理后的数据中，对每两条数据条目计算差异性得分，分值越高差异性越大
for i in range(0, range_length):
    for j in range(i, range_length):
        for item in attribute:
            temp = abs(data_norm.iloc[i][item] - data_norm.iloc[j][item])
            score[i][j] += temp
        score[j][i] = score[i][j]

# 建立原始数据的拷贝
data_filtrated = data_origin.copy()

# 对有缺失值的条目，用和它相似度最高（得分最低）的数据条目中对应属性的值替换
for index in nan_list:
    best_friend = sorted(score[index].items(), key=operator.itemgetter(1), reverse = False)[1][0]
    for item in attribute:
        if pd.isnull(data_filtrated.iloc[index][item]):
            if pd.isnull(data_origin.iloc[best_friend][item]):
                data_filtrated.ix[index, item] = data_origin[item].value_counts().idxmax()
            else:
                data_filtrated.ix[index, item] = data_origin.iloc[best_friend][item]

# 绘制可视化图
fig = plt.figure(figsize = (30,20))
```

```
i = 1
# 对数值属性，绘制直方图
for item in attribute:
    ax = fig.add_subplot(4,7,i)
    ax.set_title(item)
    data_origin[item].plot(ax = ax, alpha = 0.5, kind = 'hist', label = 'origin', legend = True)
    data_filtrated[item].plot(ax = ax, alpha = 0.5, kind = 'hist', label = 'droped', legend = True)
    ax.axvline(data_origin[item].mean(), color = 'r')
    ax.axvline(data_filtrated[item].mean(), color = 'b')
    i += 1
plt.subplots_adjust(wspace = 0.3, hspace = 0.3)

# 保存图像和处理后数据
fig.savefig('./image/missing_data_similarity.jpg')
data_filtrated.to_csv('./data_output/missing_data_similarity.csv', mode = 'w', encoding='utf-8',
index = False,header = False)
print ('filted_missing_data4 saved at ./image/filted_missing_data4.jpg')
print ('data after analysis saved at ./data_output/missing_data_similarity.csv')
```