作业 #1: 数据探索性分析与预处理

姓名： 王欣欣　　学号：2120161059　　日期：2017.4.6

课题名称：马的疝病分析——数据分析要求

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

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。

实验过程：

1. 预处理：

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
import operator
```

2. 数据处理：将数据集转为 csv 格式

```
# **step1: 数据处理**
#
# 将原始的data文件转化为csv文件
# 从原始数据文件中读出，然后写入csv文件中

origin_fp= open("../data_set/horse-colic.data",'r')
modified_fp = open("../data_set/horse-colic.csv",'w')

line = origin_fp.readline()
while(line):
    temp = line.strip().split()
    temp = ','.join(temp) + '\n'
    modified_fp.write(temp)
    line = origin_fp.readline()

origin_fp.close()
modified_fp.close()
```

3. 为数据标注属性名，分为：category 型数据、数值型数据、字符型数据

1）整个数据集可分为：

att_category：category 型数据列

att_value：除去 category 型以外的数据列

2） att_value 型还可细分为：

att_value_num：数值型数据列

att_value_str：字符串型数据列

```
# 为每列数据添加别名
att_title = ["surgery","Age","HN","rt","pulse","rr","toe",
                "pp","mm","crt","pain","peristalsis","ad","nt",
                "nr","nrP","re","abdomen","pcv","tp","aa",
                "atp","outcome","sl","ls1","ls2","ls3","cd"]
att_category = ["surgery","Age","toe","pp","mm","crt","pain","peristalsis",
                "ad","nt","nr","re","abdomen","aa","outcome","sl","cd"]
att_value = ["HN","rt","pulse","rr","nrP","pcv","tp","atp","ls1","ls2","ls3"]

att_value_num = ["rt","pulse","rr","nrP","pcv","tp","atp"]
att_value_str = ["HN","ls1","ls2","ls3"]

origin_data = pd.read_csv("../data_set/horse-colic.csv",
                          names = att_title,
                          index_col = False)
origin_data = origin_data.replace('?', np.nan)

# 将字符数据转换为category，以便进行统计
for item in att_category:
    origin_data[item] = origin_data[item].astype('category')
```

4. 对标称属性，给出每个可能取值的频数

```
# **step2: 数据摘要**
#
# ==> 对标称属性，给出每个可能取值的频数
for item in att_category:
    print(str(item) + '的频度为: \n' + str(pd.value_counts(origin_data[item].values)) + '\n')
```

显示：

```
In [1]: runfile('F:/python/test/DM.py', wdir='F:/python/test')
surgery的频度为
1    180
2    119
dtype: int64

Age的频度为:
1    276
9     24
dtype: int64

toe的频度为:
3    109
1     78
2     30
4     27
dtype: int64

pp的频度为:
1    115
3    103
4      8
2      5
dtype: int64

nr的频度为:
1    120
3     39
2     35
dtype: int64

re的频度为:
4    79
1    57
3    49
2    13
dtype: int64

abdomen的频度为:
5    79
4    43
1    28
2    19
3    13
dtype: int64

mm的频度为:
1    79
3    58
4    41
2    30
5    25
6    20
dtype: int64

crt的频度为:
1    188
2     78
3      2
dtype: int64

pain的频度为:
3    67
2    59
5    42
4    39
1    38
dtype: int64

aa的频度为:
2    48
3    46
1    41
dtype: int64

outcome的频度为:
1    178
2     77
3     44
dtype: int64

sl的频度为:
1    191
2    109
dtype: int64

cd的频度为:
2    201
1     99
dtype: int64

peristalsis的频度为:
3    128
4     73
1     39
2     16
dtype: int64

ad的频度为:
1    76
3    65
2    65
4    38
dtype: int64

nt的频度为:
2    102
1     71
3     23
dtype: int64
```

5. 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

1）最大值：

```
# 以下操作只针对数值型属性，不针对category型
#
# ==> 给出各属性最大值的个数；

for item in att_value:
    maxItem = origin_data[item].astype(float).max(skipna = True)
    maxNum = np.sum(origin_data[item].astype(float) == maxItem)
    print(str(item) + '的最大值为: ' + str(maxItem) + ',  个数为: ' + str(maxNum))

print("==================================")
```

显示：

```
HN的最大值为: 5305629.0,  个数为: 1
rt的最大值为: 40.8,  个数为: 1
pulse的最大值为: 184.0,  个数为: 1
rr的最大值为: 96.0,  个数为: 2
nrP的最大值为: 7.5,  个数为: 2
pcv的最大值为: 75.0,  个数为: 2
tp的最大值为: 89.0,  个数为: 1
atp的最大值为: 10.1,  个数为: 1
ls1的最大值为: 41110.0,  个数为: 1
ls2的最大值为: 7111.0,  个数为: 1
ls3的最大值为: 2209.0,  个数为: 1
==================================
```

2）最小值：

```
# ==> 给出各属性最小值的个数；

for item in att_value:
    mixItem = origin_data[item].astype(float).min(skipna = True)
    minNum = np.sum(origin_data[item].astype(float) == mixItem)
    print(str(item) + '的最小值为: ' + str(mixItem) + ',  个数为: ' + str(minNum))

print("==================================")
```

显示：

```
HN的最小值为: 518476.0,  个数为: 1
rt的最小值为: 35.4,  个数为: 1
pulse的最小值为: 30.0,  个数为: 2
rr的最小值为: 8.0,  个数为: 1
nrP的最小值为: 1.0,  个数为: 2
pcv的最小值为: 23.0,  个数为: 1
tp的最小值为: 3.3,  个数为: 1
atp的最小值为: 0.1,  个数为: 1
ls1的最小值为: 0.0,  个数为: 56
ls2的最小值为: 0.0,  个数为: 293
ls3的最小值为: 0.0,  个数为: 299
==================================
```

3）均值：

```
# ==> 给出各属性均值的个数：

for item in att_value:
    meanItem = origin_data[item].astype(float).mean(skipna = True)
    meanNum =  np.sum(origin_data[item].astype(float) == meanItem)
    print(str(item) + '的均值为: ' + str(meanItem) + ',  个数为: ' + str(meanNum))

print("==================================")
```

显示：

```
HN的均值为: 1085888.8333333333,  个数为: 0
rt的均值为: 38.16791666666669,  个数为: 0
pulse的均值为: 71.91304347826087,  个数为: 0
rr的均值为: 30.417355371900825,  个数为: 0
nrP的均值为: 4.707547169811321,  个数为: 0
pcv的均值为: 46.29520295202952,  个数为: 0
tp的均值为: 24.456928838951317,  个数为: 0
atp的均值为: 3.0196078431372553,  个数为: 0
ls1的均值为: 3657.88,  个数为: 0
ls2的均值为: 90.22666666666667,  个数为: 0
ls3的均值为: 7.363333333333333,  个数为: 0
==================================
```

4）中位数：

```
# ==> 给出各属性中位数的个数：

for item in att_value:
    medianItem = origin_data[item].astype(float).median(skipna = True)
    medianNum =  np.sum(origin_data[item].astype(float) == medianItem)
    print(str(item) + '的中位数为: ' + str(medianItem) + ',  个数为: ' + str(medianNum))

print("==================================")
```

显示：

```
HN的中位数为: 530305.5,  个数为: 0
rt的中位数为: 38.2,  个数为: 16
pulse的中位数为: 64.0,  个数为: 8
rr的中位数为: 24.5,  个数为: 0
nrP的中位数为: 5.0,  个数为: 4
pcv的中位数为: 45.0,  个数为: 14
tp的中位数为: 7.5,  个数为: 13
atp的中位数为: 2.25,  个数为: 0
ls1的中位数为: 2673.5,  个数为: 0
ls2的中位数为: 0.0,  个数为: 293
ls3的中位数为: 0.0,  个数为: 299
==================================
```

5）四分位数：

```
# ==> 给出各属性四分位数的个数；

for item in att_value:
    quartItem = origin_data[item][origin_data[item].isnull() == False].astype(float).describe()
    quartNum =  np.sum(origin_data[item].astype(float) == quartItem)
    print(str(item) + '的四分位数为: ' + str(quartItem) + '，  个数为: ' + str(quartNum))

print("==================================")
```

显示：

```
HN的四分位数为: 528904.0，  个数为: 2
rt的四分位数为: 37.8，  个数为: 17
pulse的四分位数为: 48.0，  个数为: 28
rr的四分位数为: 18.5，  个数为: 0
nrP的四分位数为: 3.0，  个数为: 3
pcv的四分位数为: 38.0，  个数为: 9
tp的四分位数为: 6.5，  个数为: 15
atp的四分位数为: 2.0，  个数为: 26
ls1的四分位数为: 2111.75，  个数为: 0
ls2的四分位数为: 0.0，  个数为: 293
ls3的四分位数为: 0.0，  个数为: 299
==================================
```

6）缺失值：

```
# ==> 给出各属性缺失值的个数；
for item in att_value:
    lostNum =  np.sum(origin_data[item].isnull())
    print(str(item) + '缺失值的个数为: '+ str(lostNum))
```

显示：

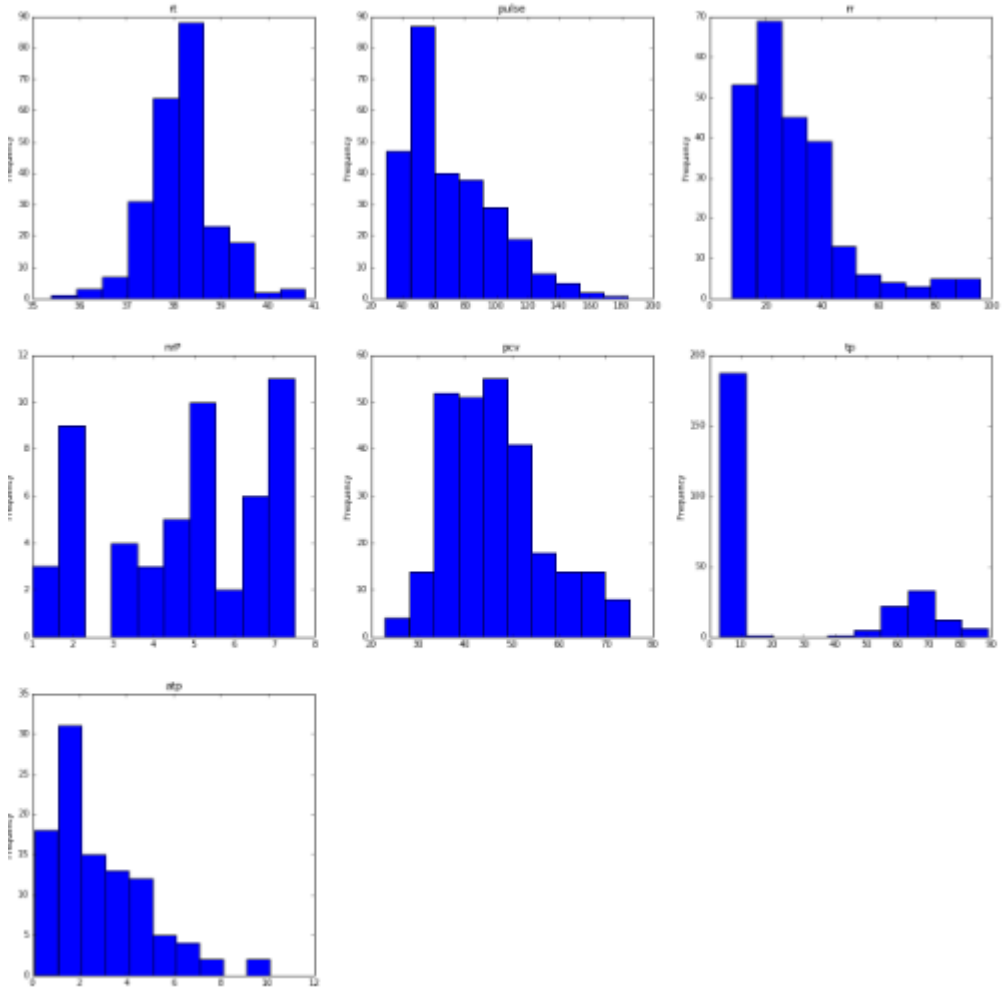
```
HN缺失值的个数为: 0
rt缺失值的个数为: 60
pulse缺失值的个数为: 24
rr缺失值的个数为: 58
nrP缺失值的个数为: 247
pcv缺失值的个数为: 29
tp缺失值的个数为: 33
atp缺失值的个数为: 198
ls1缺失值的个数为: 0
ls2缺失值的个数为: 0
ls3缺失值的个数为: 0
```

6. 数据的可视化：针对数值属性，绘制直方图、qq 图、盒图

1）直方图：

```
# **step3: 数据的可视化**
#
# 针对数值型属性，
# 绘制直方图，如mxPH，用qq图检验其分布是否为正太分布。
#
# 直方图

fig = plt.figure(figsize = (20,20))
i = 1
for item in att_value_num:
    ax = fig.add_subplot(3,3,i)
    origin_data[item].astype(float).plot(kind = 'hist', title = item, ax = ax)
    i += 1
fig.savefig("../data_set/hist.png")
```

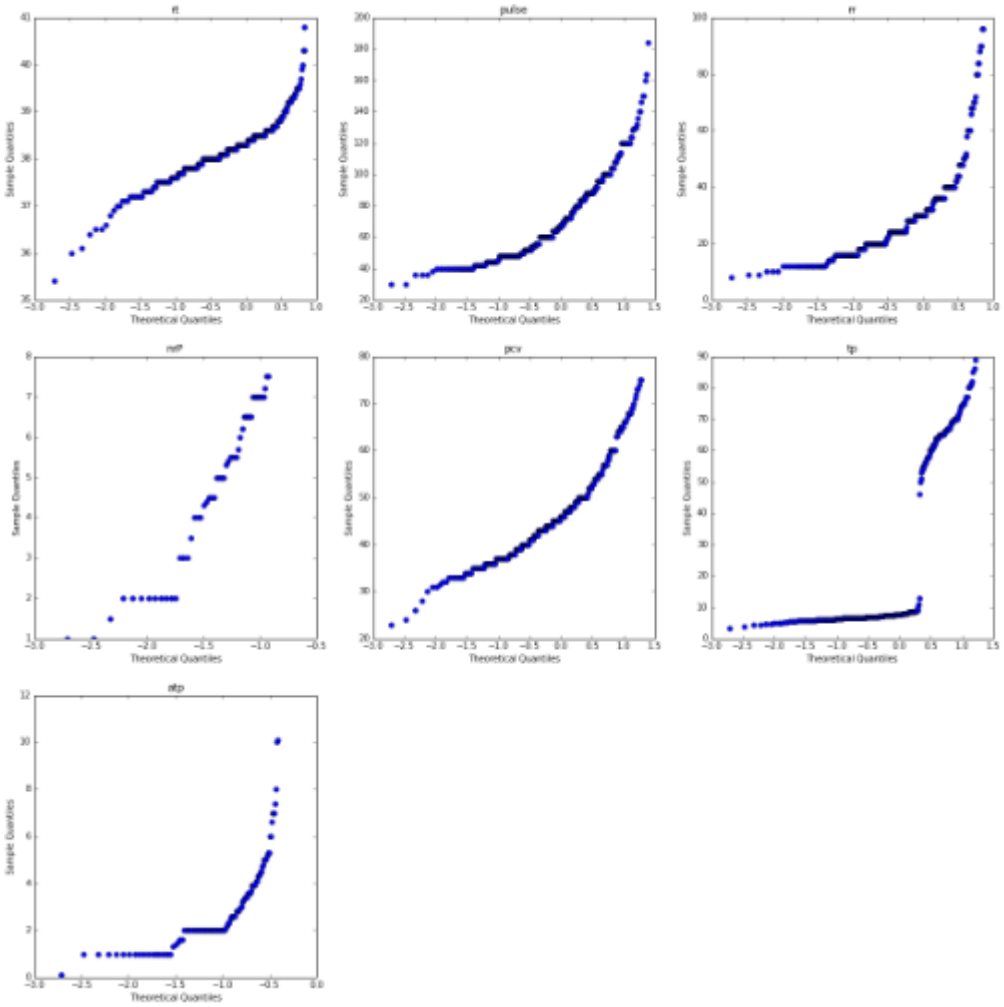
显示：



2）qq 图：

```
# qq图

fig = plt.figure(figsize = (20,20))
i = 1
for item in att_value_num:
    ax = fig.add_subplot(3,3,i)
    sm.qqplot(origin_data[item].astype(float), ax = ax)
    ax.set_title(item)
    i += 1
fig.savefig("../data_set/qq.png")

## qq图满足正太分布的条件：
### qq图上的点近似地在一条直线附近，而且该直线的斜率为标准差，截距为均值
### 由此判断属性rt(rectal temperature) 和属性 pcv(packed cell volume)符合正态分布，其余均不符合
```

显示：



分析：

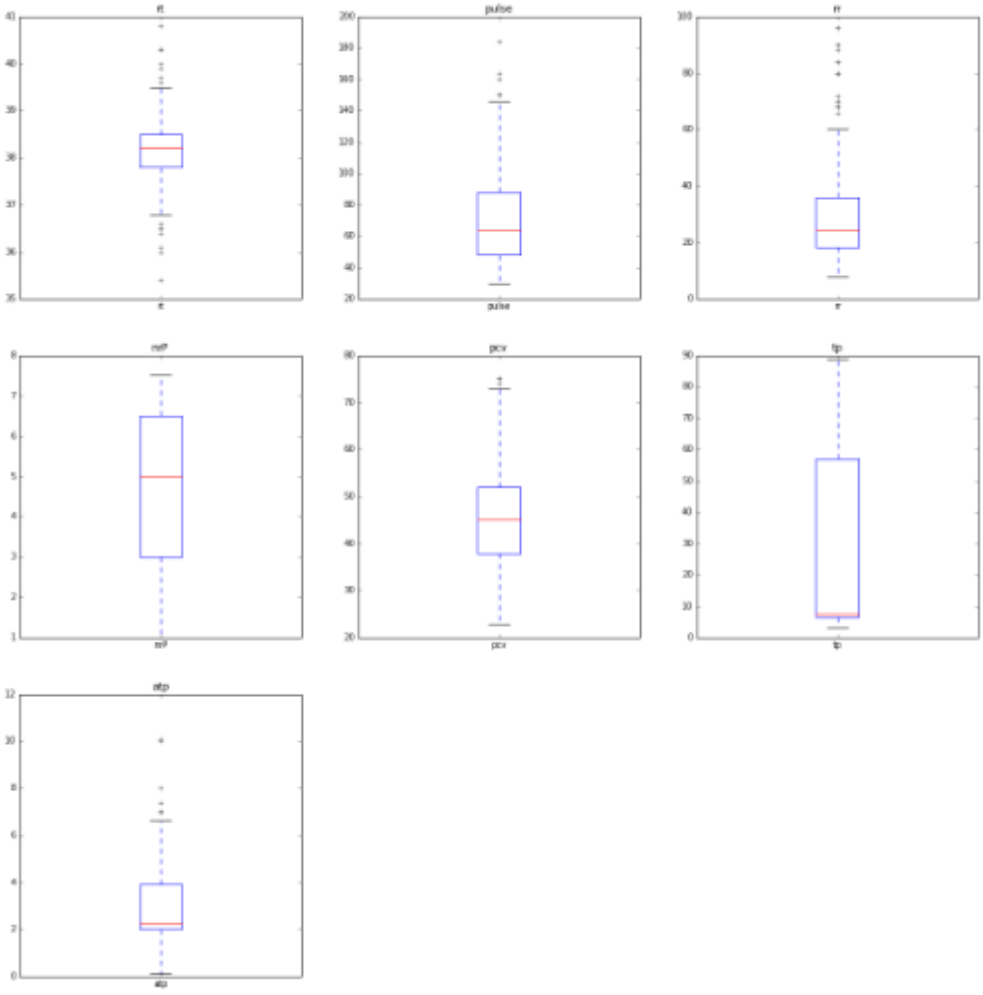
由于 qq 图满足正太分布的条件为：qq 图上的点近似地在一条直线附近，而且该直线的斜率为标准差，截距为均值

由此判断属性 rt(rectal temperature) 和属性 pcv(packed cell volume)符合正态分布，其余均不符合

3) 盒图：

```
# 绘制盒图，对离群值进行识别

fig = plt.figure(figsize = (20,20))
i = 1
for item in att_value_num:
    ax = fig.add_subplot(3,3,i)
    origin_data[item].astype(float).plot(kind = 'box', title = item, ax = ax)
    i += 1
fig.savefig("../data_set/box.png")
```

显示：



7. 处理缺失数据

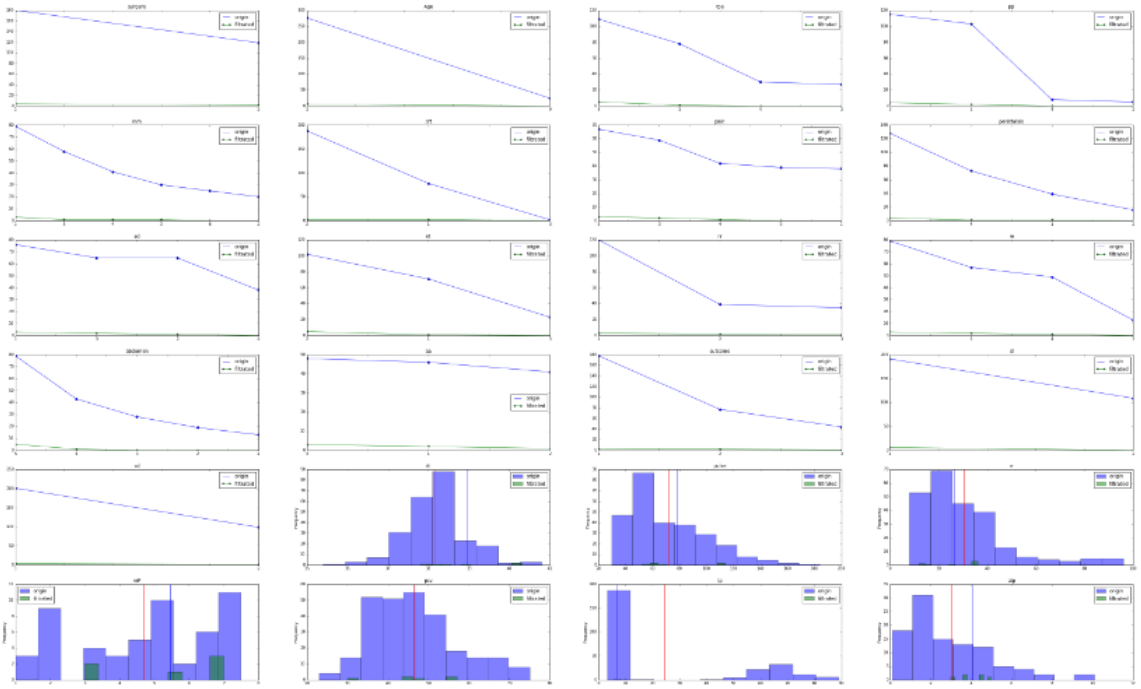
方法一：将缺失部分剔除

```
# **Step 4. 数据缺失的处理**
#
### 数据集中有30%的值是缺失的，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：
#***** 1. 将缺失部分剔除
#***** 2. 用最高频率值来填补缺失值
#***** 3. 通过属性的相关关系来填补缺失值
#***** 4. 通过数据对象之间的相似性来填补缺失值

# 1. 将缺失部分剔除

# 对原始数据集进行拷贝
filtrated_data_1 = origin_data.copy()

## -使用dropna()整行删除缺失值
filtrated_data_1 = filtrated_data_1.dropna()
```

画图并保存处理后的数据集：

```
## -绘制可视化图像


## --对标称属性，绘制折线图

fig = plt.figure(figsize = (50,30))
i = 1
for item in att_category:
    ax = fig.add_subplot(6,4,i)
    ax.set_title(item)
    pd.value_counts(origin_data[item].values).plot(ax = ax, marker = 'o', label = 'origin', legend = True)
    pd.value_counts(filtrated_data_1[item].values).plot(ax = ax, marker = '*', label = 'filtrated', legend = True)
    i += 1


## --对数值型属性，绘制直方图
i = 18
for item in att_value_num:
    ax = fig.add_subplot(6,4,i)
    ax.set_title(item)
    origin_data[item].astype(float).plot(kind = 'hist', ax = ax, label = 'origin', legend = True, alpha = 0.5)
    filtrated_data_1[item].astype(float).plot(kind = 'hist', ax = ax, label = 'filtrated', legend = True, alpha = 0.5)
    ax.axvline(origin_data[item].astype(float).mean(), color = 'r')
    ax.axvline(filtrated_data_1[item].astype(float).mean(), color = 'b')
    i += 1

fig.savefig("../data_set/loss1.png")
filtrated_data_1.to_csv('../data_set/filtrated_data_1.csv', mode = 'w', encoding = 'utf-8', index = False, header = False)
```

显示：



方法二：用最高频率值来填补缺失值

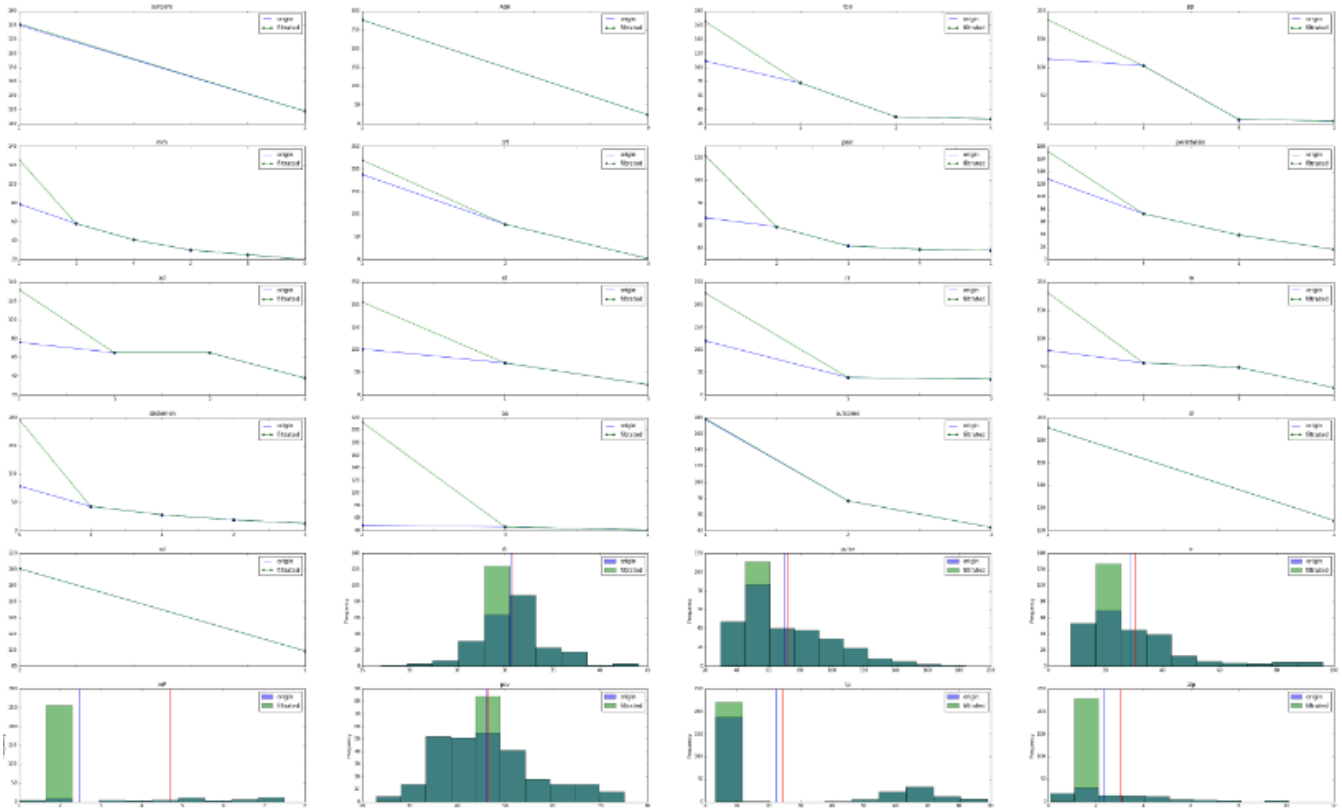
```
# 对原始数据集进行拷贝
filtrated_data_2 = origin_data.copy()

# 对每一列，分别计算高频属性值
for item in att_title:
    most_frq_value = pd.value_counts(origin_data[item].values).idxmax()

    ## -使用fillna()替换缺失值
    filtrated_data_2[item].fillna(value = most_frq_value , inplace = True )
```

画图并保存处理后的数据集：

```
## -绘制可视化图像

## --对标称属性，绘制折线图

fig = plt.figure(figsize = (50,30))
i = 1
for item in att_category:
    ax = fig.add_subplot(6,4,i)
    ax.set_title(item)
    pd.value_counts(origin_data[item].values).plot(ax = ax, marker = 'o', label = 'origin', legend = True)
    pd.value_counts(filtrated_data_2[item].values).plot(ax = ax, marker = '*', label = 'filtrated', legend = True)
    i += 1
```

```
## --对数值型属性，绘制直方图
i = 18
for item in att_value_num:
    ax = fig.add_subplot(6,4,i)
    ax.set_title(item)
    origin_data[item].astype(float).plot(kind = 'hist', ax = ax, label = 'origin', legend = True, alpha = 0.5)
    filtrated_data_2[item].astype(float).plot(kind = 'hist', ax = ax, label = 'filtrated', legend = True, alpha = 0.5)
    ax.axvline(origin_data[item].astype(float).mean(), color = 'r')
    ax.axvline(filtrated_data_2[item].astype(float).mean(), color = 'b')
    i += 1

fig.savefig("../data_set/loss2.png")
filtrated_data_2.to_csv('../data_set/filtrated_data_2.csv', mode = 'w', encoding = 'utf-8', index = False, header = False)
```

显示：



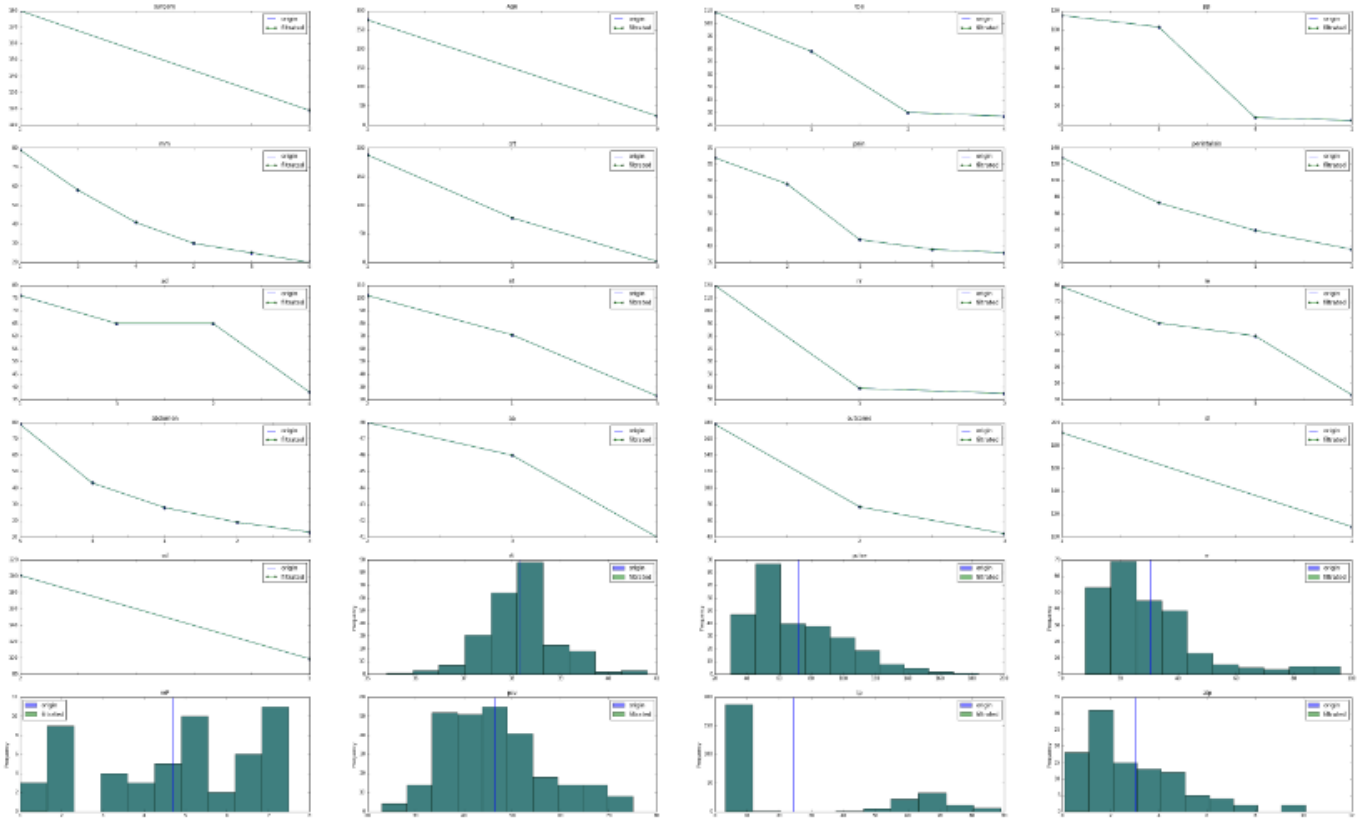
方法三：通过属性的相关关系来填补缺失值

```
# 3. 通过属性的相关关系来填补缺失值

# 对原始数据集进行拷贝
filtrated_data_3 = origin_data.copy()

# 使用 interpolate()，对数值属性进行插值法替换缺失值
for item in att_value_num:
    filtrated_data_3[item].interpolate(inplace = True)
```

画图并保存处理后的数据集：

```
## -绘制可视化图像

## --对标称属性，绘制折线图

fig = plt.figure(figsize = (50,30))
i = 1
for item in att_category:
    ax = fig.add_subplot(6,4,i)
    ax.set_title(item)
    pd.value_counts(origin_data[item].values).plot(ax = ax, marker = 'o', label = 'origin', legend = True)
    pd.value_counts(filtrated_data_3[item].values).plot(ax = ax, marker = '*', label = 'filtrated', legend = True)
    i += 1


## --对数值型属性，绘制直方图
i = 18
for item in att_value_num:
    ax = fig.add_subplot(6,4,i)
    ax.set_title(item)
    origin_data[item].astype(float).plot(kind = 'hist', ax = ax, label = 'origin', legend = True, alpha = 0.5)
    filtrated_data_3[item].astype(float).plot(kind = 'hist', ax = ax, label = 'filtrated', legend = True, alpha = 0.5)
    ax.axvline(origin_data[item].astype(float).mean(), color = 'r')
    ax.axvline(filtrated_data_3[item].astype(float).mean(), color = 'b')
    i += 1

fig.savefig("../data_set/loss3.png")
filtrated_data_3.to_csv('../data_set/filtrated_data_3.csv', mode = 'w', encoding = 'utf-8', index = False, header = False)
```

显示：



方法四：通过数据对象之间的相似性填补缺失值

```
# 4. 通过数据对象之间的相似性来填补缺失值

# 对原始数据集进行拷贝，用来进行正则化处理
copy_data = origin_data.copy()

# 将备份数据集中数值属性的缺失值替换为0
copy_data[att_value_num] = copy_data[att_value_num].fillna(0)

# 对数据进行正则化：
copy_data[att_value_num] = copy_data[att_value_num].astype(float).apply(lambda x : (x - np.mean(x)) / (np.max(x) - np.min(x)))
```

```
# 构造分数表
score = {}
range_length = len(copy_data)
for i in range(0, range_length):
    score[i] = {}
    for j in range(0, range_length):
        score[i][j] = 0
```

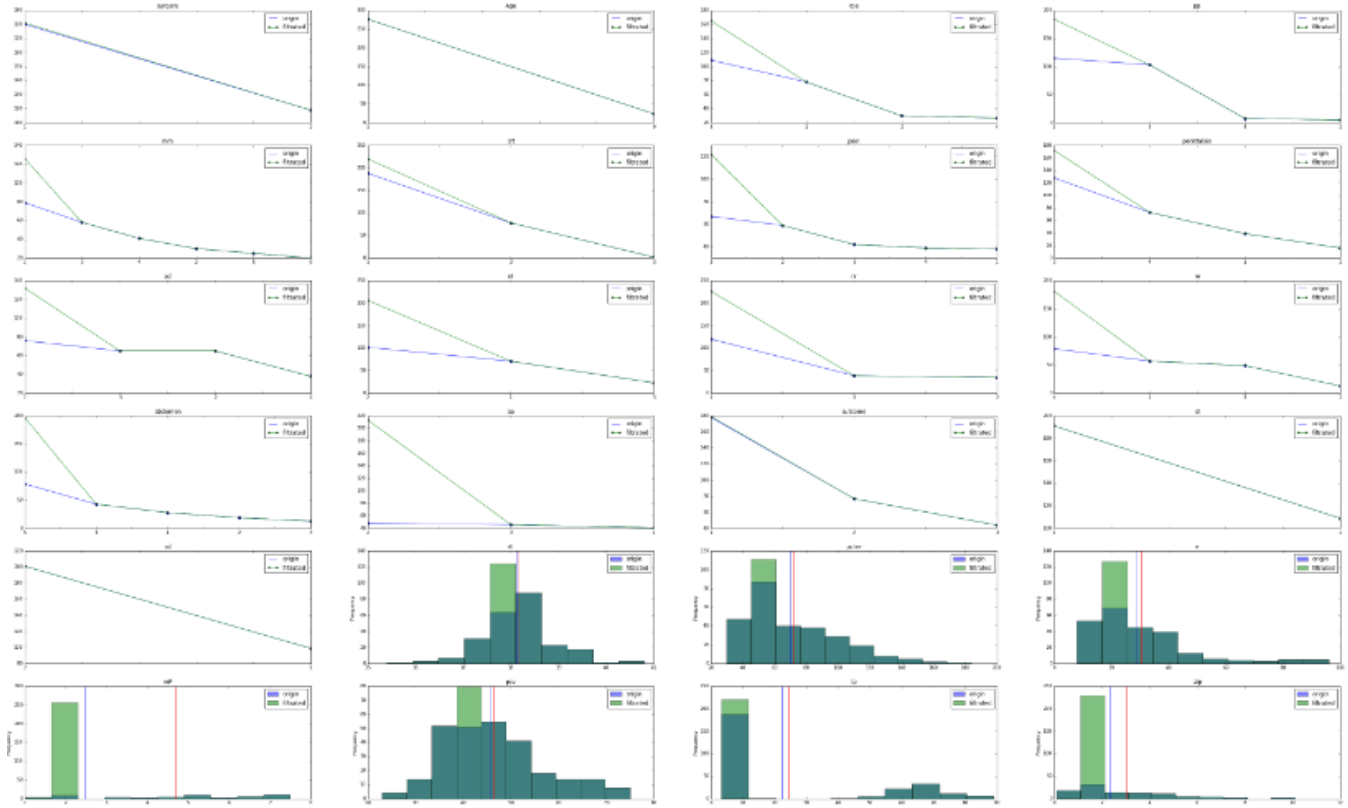
```
# 在处理后的数据中，计算两条数据的差异性得分，分值越高，差异性越大
for i in range(0, range_length):
    for j in range(i, range_length):
        for item in att_category:
            if copy_data.iloc[i][item] != copy_data.iloc[j][item]:
                score[i][j] += 1
        for item in att_value_num:
            temp = abs((copy_data.iloc[i][item]).astype(float) - (copy_data.iloc[j][item]).astype(float))
            score[i][j] += temp
        score[j][i] = score[i][j]
```

```
# 建立原始数据集的拷贝
filtrated_data_4 = origin_data.copy()

# 查找所有具有缺失值的条目
list_nan = pd.isnull(origin_data).any(1).nonzero()[0]

# 对有缺失值的条目，用与之相似度最高（得分最低）的数据条目对应的属性值进行替换
for index in list_nan:
    similar = sorted(score[index].items(), key=operator.itemgetter(1), reverse = False)[1][0]
    for item in att_value:
        if pd.isnull(filtrated_data_4.iloc[index][item]):
            if pd.isnull(origin_data.iloc[similar][item]):
                filtrated_data_4.ix[index,item] = origin_data[item].value_counts().idmax()
            else:
                filtrated_data_4.ix[index,item] = origin_data.iloc[similar][item]
```

画图并保存处理后的数据集：

```
## -绘制可视化图像

## --对标称属性，绘制折线图

fig = plt.figure(figsize = (50,30))
i = 1
for item in att_category:
    ax = fig.add_subplot(6,4,i)
    ax.set_title(item)
    pd.value_counts(origin_data[item].values).plot(ax = ax, marker = 'o', label = 'origin', legend = True)
    pd.value_counts(filtrated_data_4[item].values).plot(ax = ax, marker = '*', label = 'filtrated', legend = True)
    i += 1


## --对数值型属性，绘制直方图
i = 18
for item in att_value_num:
    ax = fig.add_subplot(6,4,i)
    ax.set_title(item)
    origin_data[item].astype(float).plot(kind = 'hist', ax = ax, label = 'origin', legend = True, alpha = 0.5)
    filtrated_data_4[item].astype(float).plot(kind = 'hist', ax = ax, label = 'filtrated', legend = True, alpha = 0.5)
    ax.axvline(origin_data[item].astype(float).mean(), color = 'r')
    ax.axvline(filtrated_data_4[item].astype(float).mean(), color = 'b')
    i += 1

fig.savefig("../data_set/loss4.png")
filtrated_data_4.to_csv('../data_set/filtrated_data_4.csv', mode = 'w', encoding = 'utf-8', index = False, header = False)
```

显示：



文件目录数据集及生成图像截图：