# First_DataMining

2020 年 5 月 3 日

[1]:
```
#data set: Chicago Building Violations
#There are 32 columns of data in the dataset

# 代码仓库:https://github.com/syb-5213/DataMining

#load data
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import  linear_model
import itertools
import math
```

[2]:
```
# 加载数据
print("读取数据中...")
data = pd.read_csv('building-violations.csv')
print("读取完毕。")
print("属性有: ID、VIOLATION LAST MODIFIED DATE、VIOLATION DATE、VIOLATION CODE、
VIOLATION STATUS、VIOLATION STATUS DATE、VIOLATION DESCRIPTION、VIOLATION␣
 ↪LOCATION、VIOLATION INSPECTOR COMMENTS、VIOLATION ORDINANCE、INSPECTOR ID、
INSPECTION NUMBER、INSPECTIONSTATUS、INSPECTION WAIVED、INSPECTION CATEGORY、
DEPARTMENT BUREAU、ADDRESS、STREET NUMBER、STREET DIRECTION、STREET NAME、STREET␣
 ↪TYPE、PROPERTYGROUP、SSA、LATITUDE、LONGITUDE、LOCATION、Community Areas、Zip␣
 ↪Codes、Boundaries-ZIP Codes、Census Tracts、Wards、Historical Wards 2003-2015")
```

```
读取数据中...
读取完毕。
属性有: ID、VIOLATION LAST MODIFIED DATE、VIOLATION DATE、VIOLATION CODE、VIOLATION
```

STATUS、VIOLATION STATUS DATE、VIOLATION DESCRIPTION、VIOLATION LOCATION、VIOLATION INSPECTOR COMMENTS、VIOLATION ORDINANCE、INSPECTOR ID、INSPECTION NUMBER、INSPECTIONSTATUS、INSPECTION WAIVED、INSPECTION CATEGORY、DEPARTMENT BUREAU、ADDRESS、STREET NUMBER、STREET DIRECTION、STREET NAME、STREET TYPE、PROPERTYGROUP、SSA、LATITUDE、LONGITUDE、LOCATION、Community Areas、Zip Codes、Boundaries-ZIP Codes、Census Tracts、Wards、Historical Wards 2003-2015

[3]:
```
# 提取数据集中数值属性
v_title=["LATITUDE","LONGITUDE","Census Tracts","Wards","Historical Wards␣
 ↪2003-2015"]
val_data=data[v_title]
print("数值属性有: LATITUDE, LONGITUDE, Census Tracts, Wards, Historical Wards␣
 ↪2003-2015;")
```

数值属性有: LATITUDE, LONGITUDE, Census Tracts, Wards, Historical Wards 2003-2015;

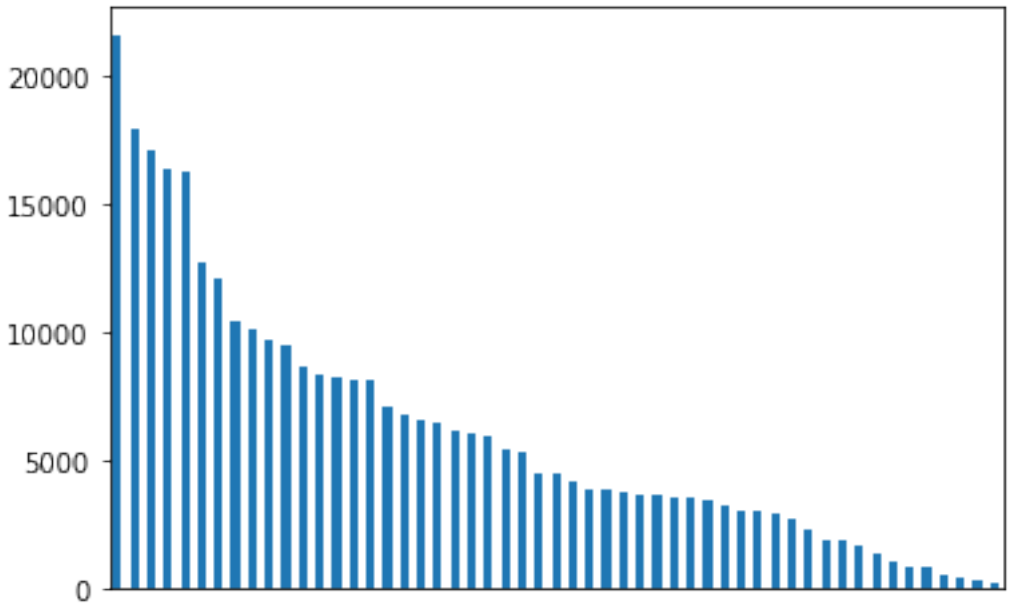
[4]:
```
# 提取数据集中标称属性
c_title=["SSA"]
print("标称属性有: ID、VIOLATION LAST MODIFIED DATE、VIOLATION DATE、VIOLATION␣
 ↪CODE、VIOLATIONSTATUS、VIOLATION STATUS DATE、VIOLATION DESCRIPTION、
VIOLATION LOCATION、VIOLATION INSPECTOR COMMENTS、VIOLATION ORDINANCE、INSPECTOR␣
 ↪ID、INSPECTION NUMBER、INSPECTIONSTATUS、INSPECTION WAIVED、INSPECTION␣
 ↪CATEGORY、DEPARTMENT BUREAU、ADDRESS、STREET NUMBER、STREET DIRECTION、STREET␣
 ↪NAME、STREET TYPE、PROPERTYGROUP、SSA、LOCATION、Community Areas、Zip Codes、
Boundaries-ZIP Codes;")
```

标称属性有: ID、VIOLATION LAST MODIFIED DATE、VIOLATION DATE、VIOLATION CODE、VIOLATIONSTATUS、VIOLATION STATUS DATE、VIOLATION DESCRIPTION、VIOLATION LOCATION、VIOLATION INSPECTOR COMMENTS、VIOLATION ORDINANCE、INSPECTOR ID、INSPECTION NUMBER、INSPECTIONSTATUS、INSPECTION WAIVED、INSPECTION CATEGORY、DEPARTMENT BUREAU、ADDRESS、STREET NUMBER、STREET DIRECTION、STREET NAME、STREET TYPE、PROPERTYGROUP、SSA、LOCATION、Community Areas、Zip Codes、Boundaries-ZIP Codes;

[5]:
```
# 列出标称属性的频度图，由于属性太多只显示 1 个 SSA
print("标称属性 SSA 频度图 (隐去横坐标):")
data['SSA'].value_counts().plot.bar()
```

```
plt.xticks([])
plt.show()
```

标称属性 SSA 频度图 (隐去横坐标):




[6]:
```
# 列出每个数值属性的五数
print("各个数值属性五数概括:")
val_title=val_data.columns.values.tolist()
for i in val_title:
    t_data=val_data[i]
    print(i+":")
    print("\tMin:\t"+str(np.nanmin(t_data)))
    print("\tQ1:\t"+str(np.nanpercentile(t_data,25)))
    print("\tMedian:\t"+str(np.nanmedian(t_data)))
    print("\tQ3:\t"+str(np.nanpercentile(t_data,75)))
    print("\tMax:\t"+str(np.nanmax(t_data)))
    print("")
```

各个数值属性五数概括:

LATITUDE:

Min: 41.644670131999995
Q1: 41.770896504250004
Median: 41.85400233599999
Q3: 41.913504192
Max: 42.02268599

LONGITUDE:
Min: -87.914435848
Q1: -87.71391769799999
Median: -87.6698535045
Q3: -87.632882744
Max: -87.524679151

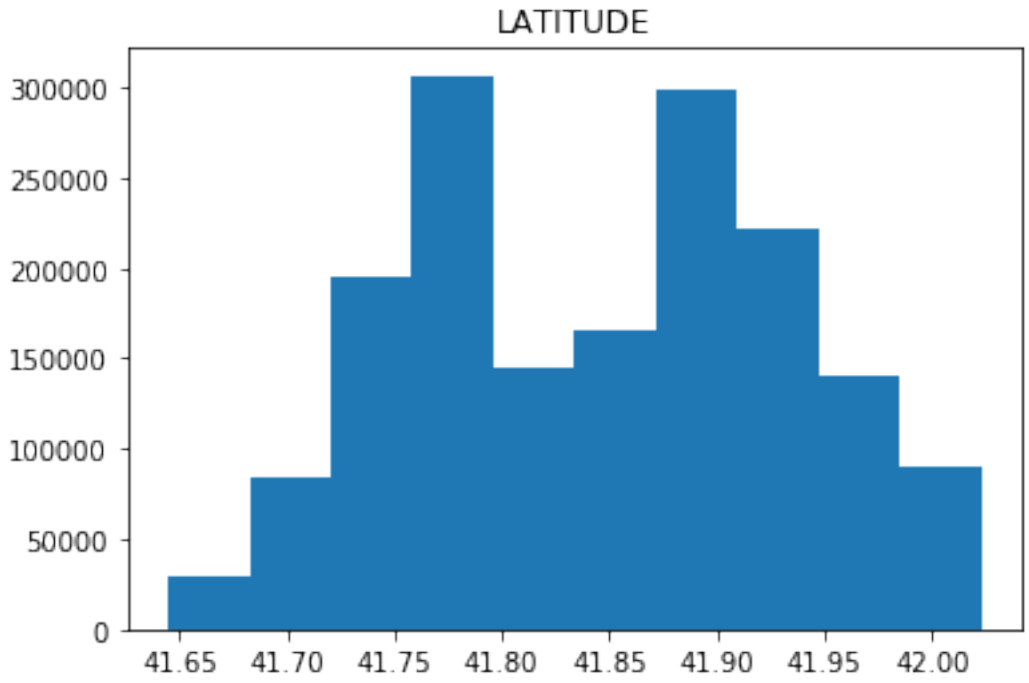
Census Tracts:
Min: 1.0
Q1: 179.0
Median: 374.0
Q3: 572.0
Max: 801.0

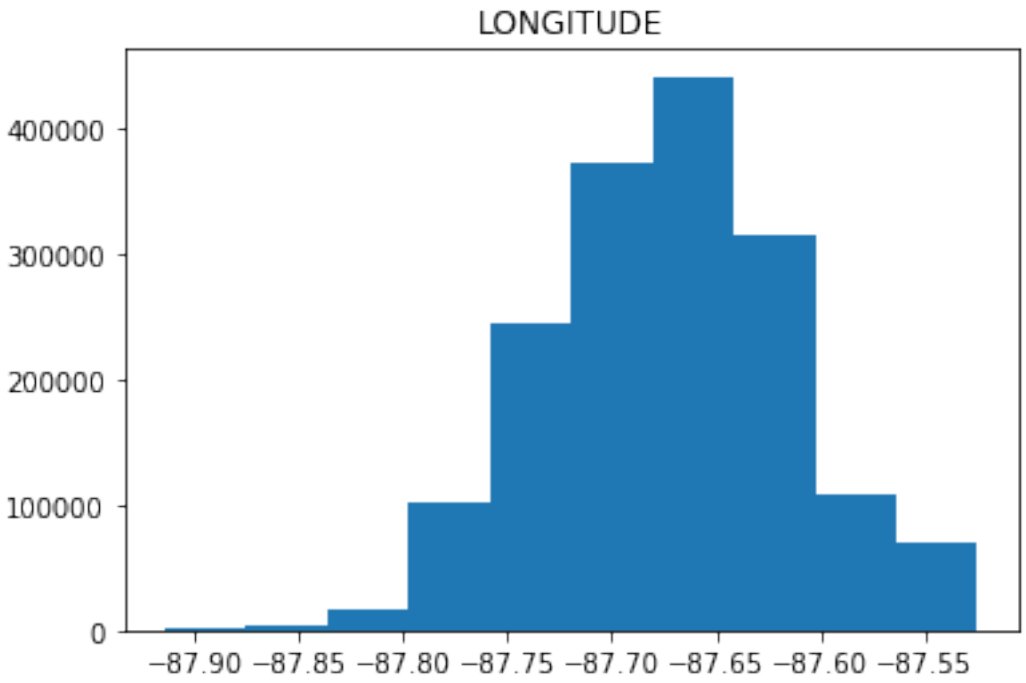
Wards:
Min: 1.0
Q1: 12.0
Median: 25.0
Q3: 37.0
Max: 50.0

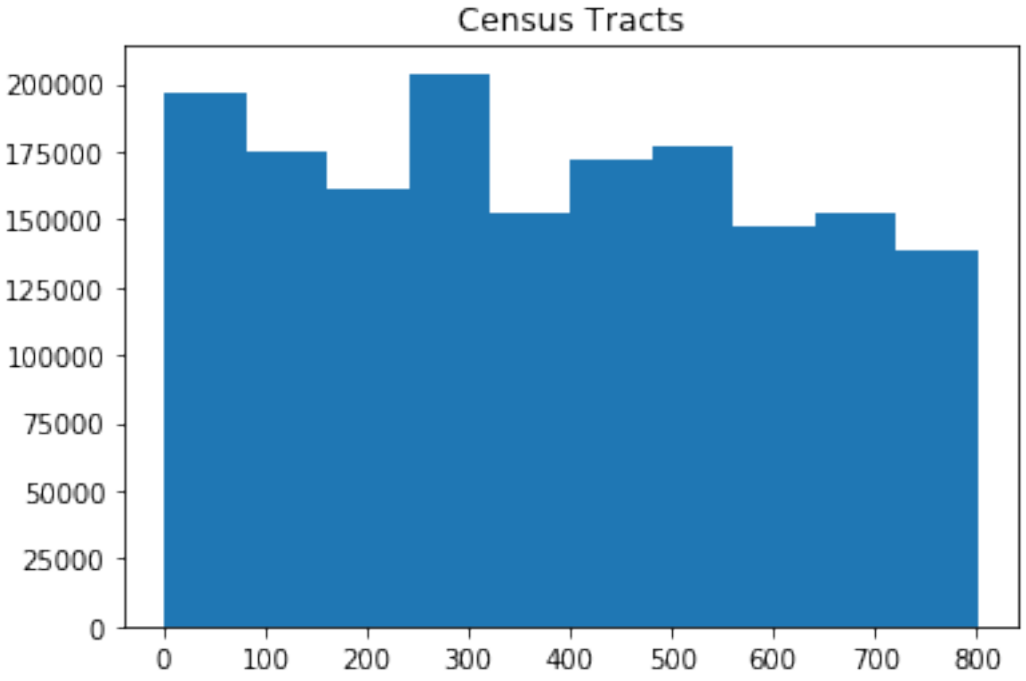
Historical Wards 2003-2015:
Min: 1.0
Q1: 14.0
Median: 28.0
Q3: 41.0
Max: 53.0

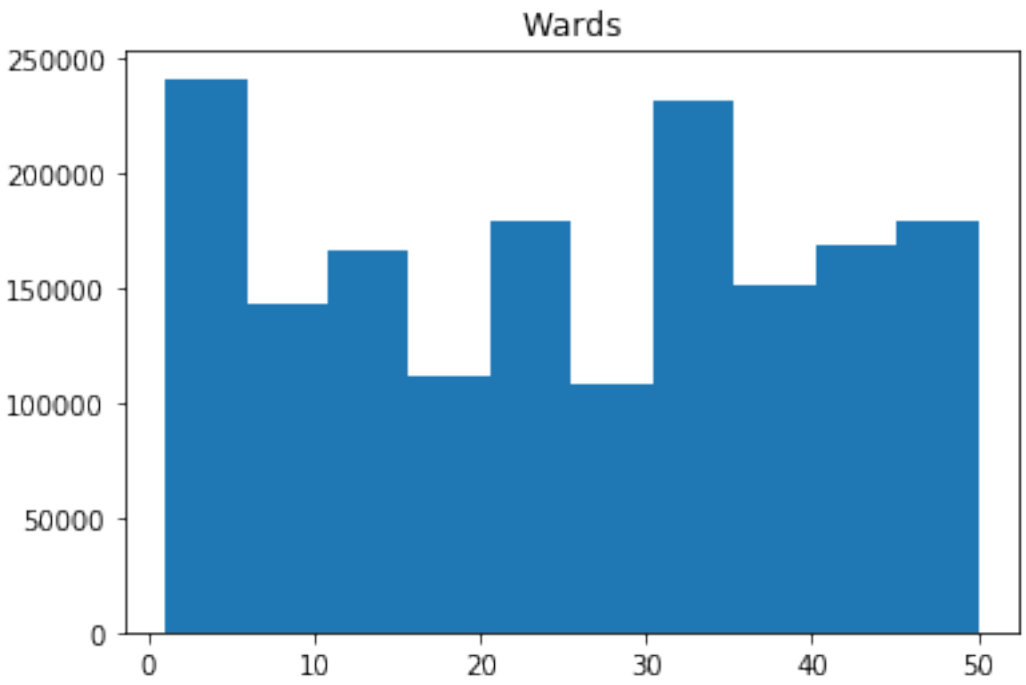
[7]:
```
# 分别画出每个数值数据直方图
print("各个数值属性直方图:")
val_title=val_data.columns.values.tolist()
for i in val_title:
    val_data.hist(grid=False,column=i)
    plt.show()
```

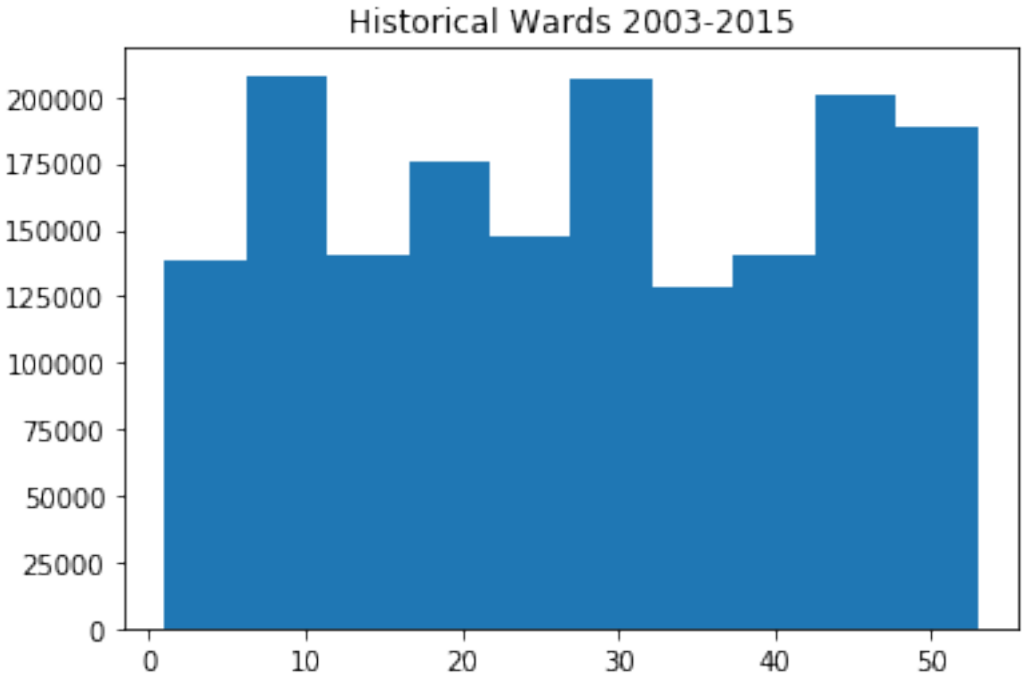
各个数值属性直方图:




LONGITUDE
400000
300000
200000
100000
0
−87.90 −87.85 −87.80 −87.75 −87.70 −87.65 −87.60 −87.55

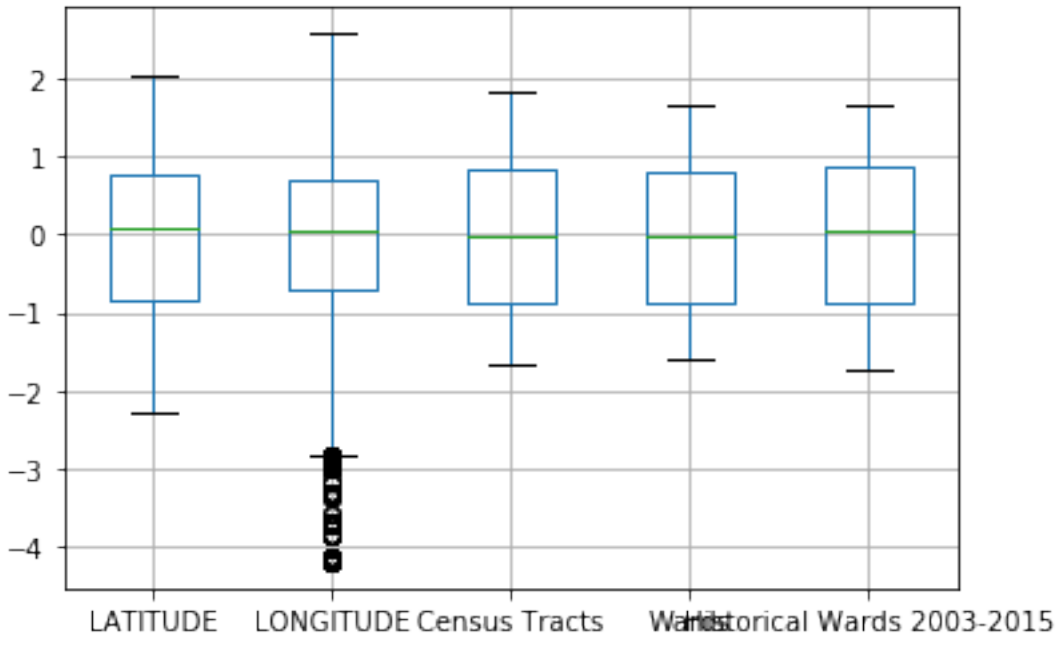

Census Tracts
200000
175000
150000
125000
100000
75000
50000
25000
0
0 100 200 300 400 500 600 700 800


Wards
250000
200000
150000
100000
50000
0
0
10
20
30
40
50


Historical Wards 2003-2015
200000
175000
150000
125000
100000
75000
50000
25000
0
0
10
20
30
40
50

[8]:
```
# 分别画出每个数据盒图
print("各个数值属性盒图:")
temp_data=val_data
temp_data = (temp_data - np.mean(temp_data,axis=0)) / np.std(temp_data,axis=0)
boxplot=temp_data.boxplot()
plt.show()
```

各个数值属性盒图:




[9]:
```
# 属性离群点
print("离群点:由每个属性的盒图可以初步看出 LONGITUDE 属性具有一部分离群点，其余数值属性未发现离群点。")
```

离群点:由每个属性的盒图可以初步看出 LONGITUDE 属性具有一部分离群点，其余数值属性未发现离群点。

[10]:
```
# 列出每个属性缺失值数量
print("各属性缺失值概括:")
nan_number=data.isnull().sum()
print(nan_number)
```

```
# 分析数值属性缺失原因
print("数值属性缺失值原因分析:由各属性缺失值统计可以看出 LATITUDE 与 LONGITUDE 以及 Wards 与 Historical Wards 2003-2015 总是成对缺失，进一步观察缺失项发现大部分 ward 缺失项的经纬坐标基本都是 42.00，-87.91，所以考虑是该位置的设备原因造成的数据缺失。")
```

各属性缺失值概括:

| | |
|---|---|
| ID | 0 |
| VIOLATION LAST MODIFIED DATE | 0 |
| VIOLATION DATE | 0 |
| VIOLATION CODE | 0 |
| VIOLATION STATUS | 0 |
| VIOLATION STATUS DATE | 1036199 |
| VIOLATION DESCRIPTION | 10768 |
| VIOLATION LOCATION | 897282 |
| VIOLATION INSPECTOR COMMENTS | 175463 |
| VIOLATION ORDINANCE | 47581 |
| INSPECTOR ID | 0 |
| INSPECTION NUMBER | 0 |
| INSPECTION STATUS | 16 |
| INSPECTION WAIVED | 0 |
| INSPECTION CATEGORY | 0 |
| DEPARTMENT BUREAU | 0 |
| ADDRESS | 0 |
| STREET NUMBER | 0 |
| STREET DIRECTION | 0 |
| STREET NAME | 0 |
| STREET TYPE | 13541 |
| PROPERTY GROUP | 0 |
| SSA | 1356267 |
| LATITUDE | 1510 |
| LONGITUDE | 1510 |
| LOCATION | 1510 |
| Community Areas | 2279 |
| Zip Codes | 1510 |
| Boundaries - ZIP Codes | 2279 |
| Census Tracts | 1545 |

```
Wards                                2279
Historical Wards 2003-2015           2279
dtype: int64
```

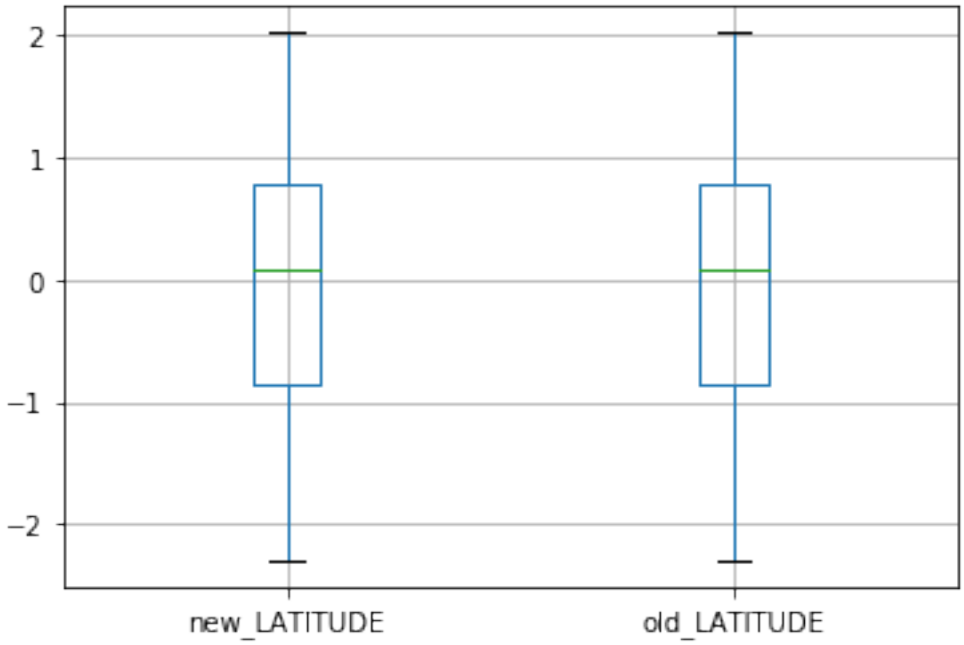
数值属性缺失值原因分析:由各属性缺失值统计可以看出 LATITUDE 与 LONGITUDE 以及 Wards 与 Historical Wards 2003-2015 总是成对缺失，进一步观察缺失项发现大部分 ward 缺失项的经纬坐标基本都是 42.00，-87.91，所以考虑是该位置的设备原因造成的数据缺失。

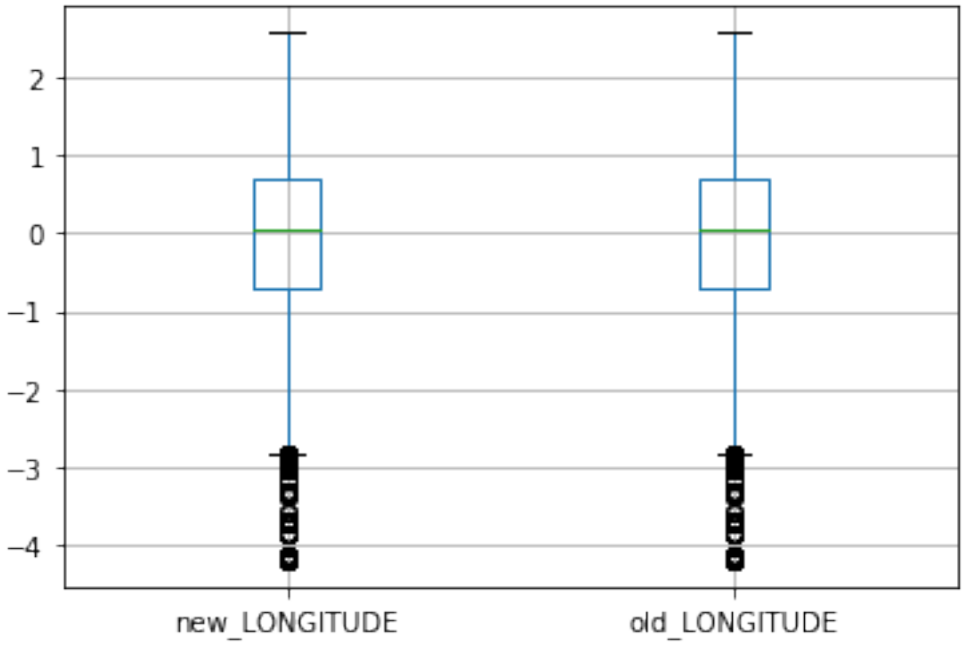
[11]:
```
# 缺失值填充
print("缺失值填充")
```

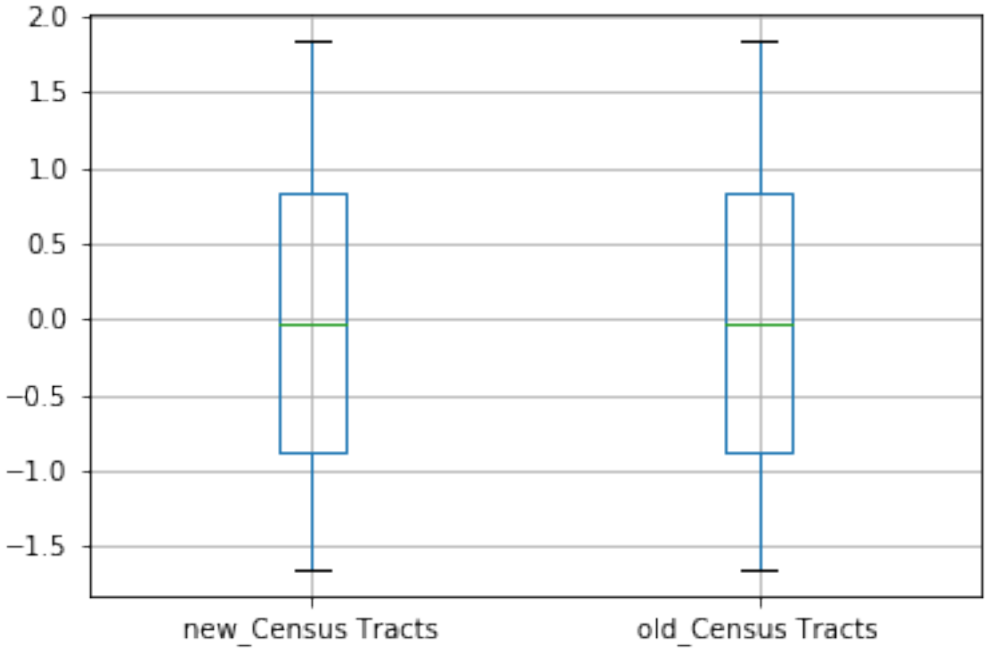
缺失值填充

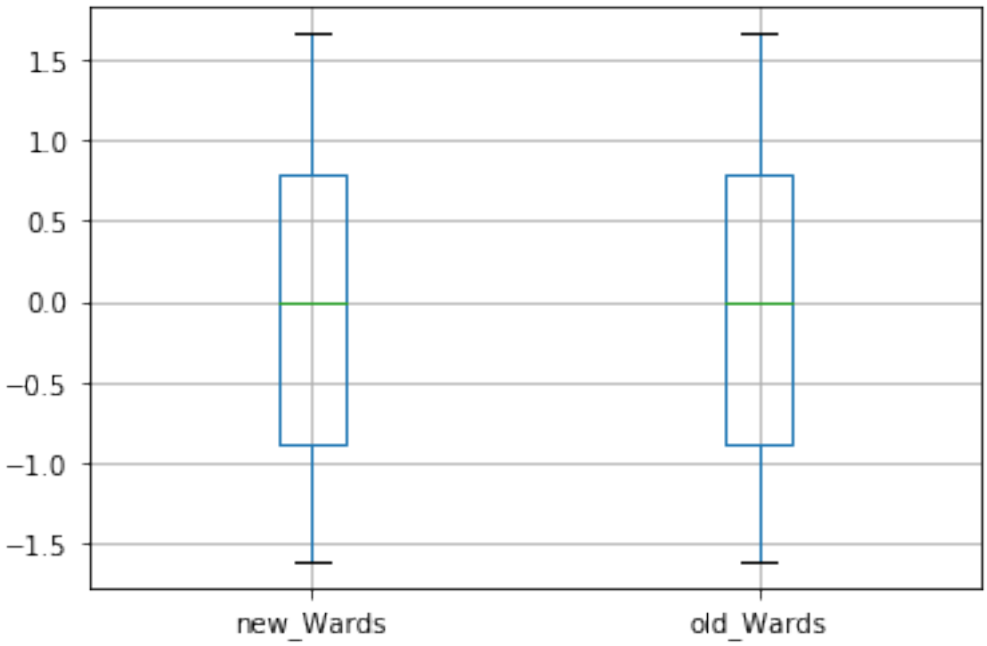
[12]:
```
#1.将缺失部分剔除
print("1.将缺失部分剔除")
print("各数值属性剔除缺失值前后各属性盒图对比:")
val_title=val_data.columns.values.tolist()
for i in val_title:
    t_data=val_data[i]
    d_val_data=t_data.dropna()
    contrast=pd.DataFrame(list(itertools.
 ↪zip_longest(d_val_data,t_data)),columns=['new_'+i, 'old_'+i])
    contrast = (contrast - np.mean(contrast,axis=0)) / np.std(contrast,axis=0)
    boxplot=contrast.boxplot()
    plt.show()
```

1.将缺失部分剔除
各数值属性剔除缺失值前后各属性盒图对比:


2
1
0
−1
−2
new_LATITUDE
old_LATITUDE

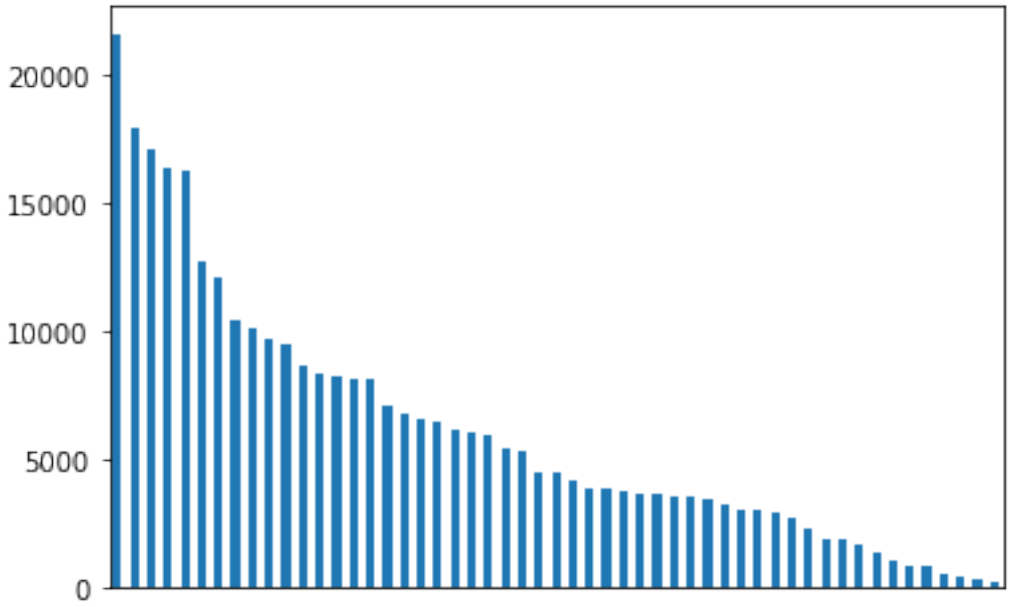

2
1
0
−1
−2
−3
−4
new_LONGITUDE
old_LONGITUDE


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
new_Census Tracts
old_Census Tracts

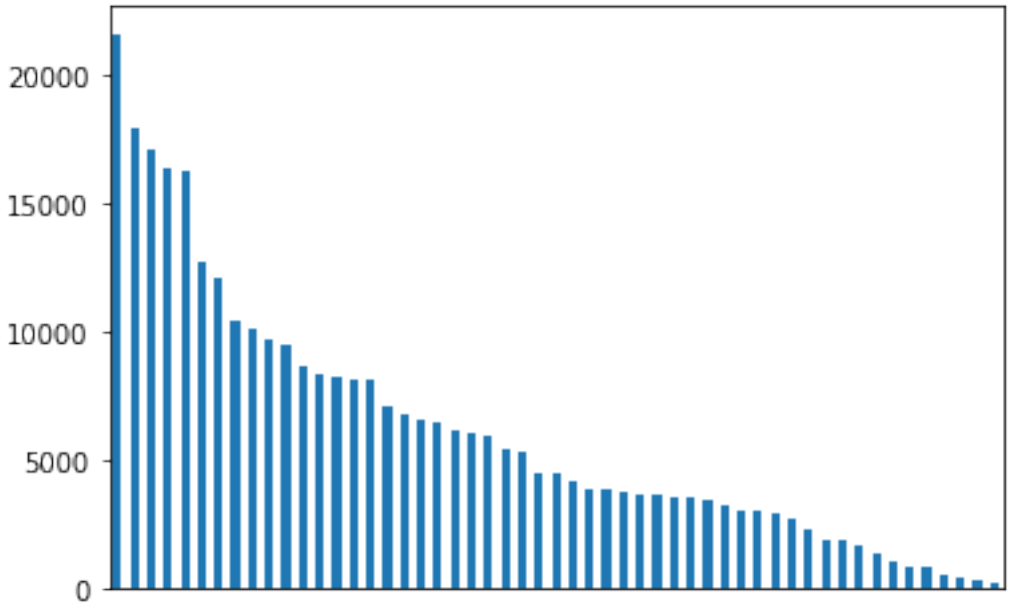

1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
new_Wards
old_Wards


[13]:
```
print("标称属性 SSA 剔除缺失值前后频度图对比:")
print("剔除前:")
data['SSA'].value_counts().plot.bar()
plt.xticks([])
plt.show()
print("剔除后:")
data['SSA'].dropna().value_counts().plot.bar()
plt.xticks([])
plt.show()
```

标称属性 SSA 剔除缺失值前后频度图对比:
剔除前:




剔除后：



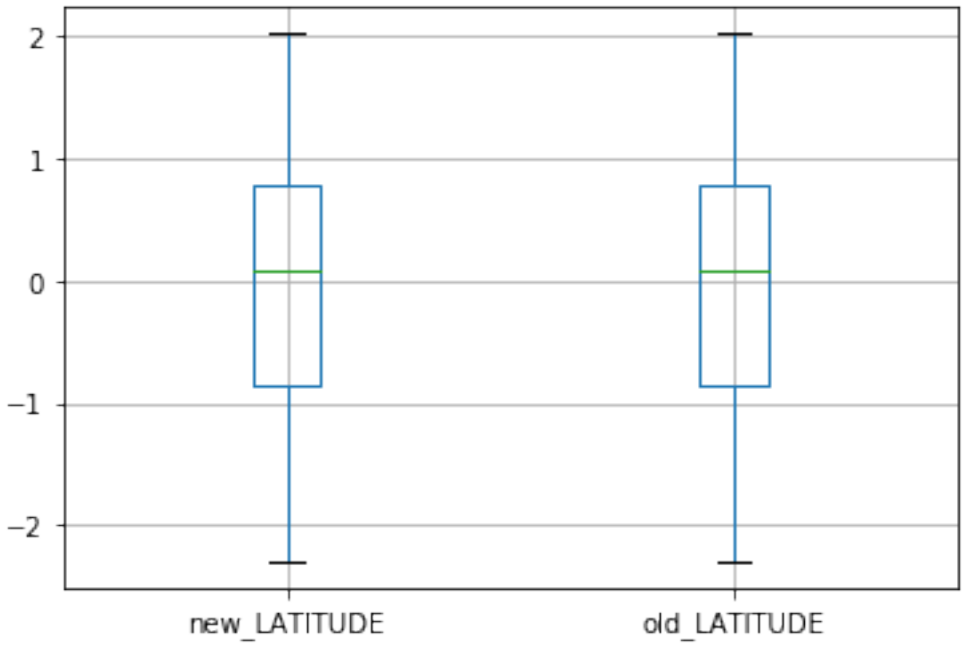

[14]:
```
#2.用最高频率值来填补缺失值
print("2.用最高频率值来填补缺失值")
```

```
print("各数值属性填充前后各属性盒图对比:")
val_title=val_data.columns.values.tolist()
for i in val_title:
    t_data=val_data[i]
    print(i+"数据集中最高频率值为"+str(t_data.mode()[0])+"。")
    d_val_data=t_data.fillna(value=t_data.mode()[0])
    contrast=pd.DataFrame(list(itertools.
 ↪zip_longest(d_val_data,t_data)),columns=['new_'+i, 'old_'+i])
    contrast = (contrast - np.mean(contrast,axis=0)) / np.std(contrast,axis=0)
    boxplot=contrast.boxplot()
    plt.show()
```

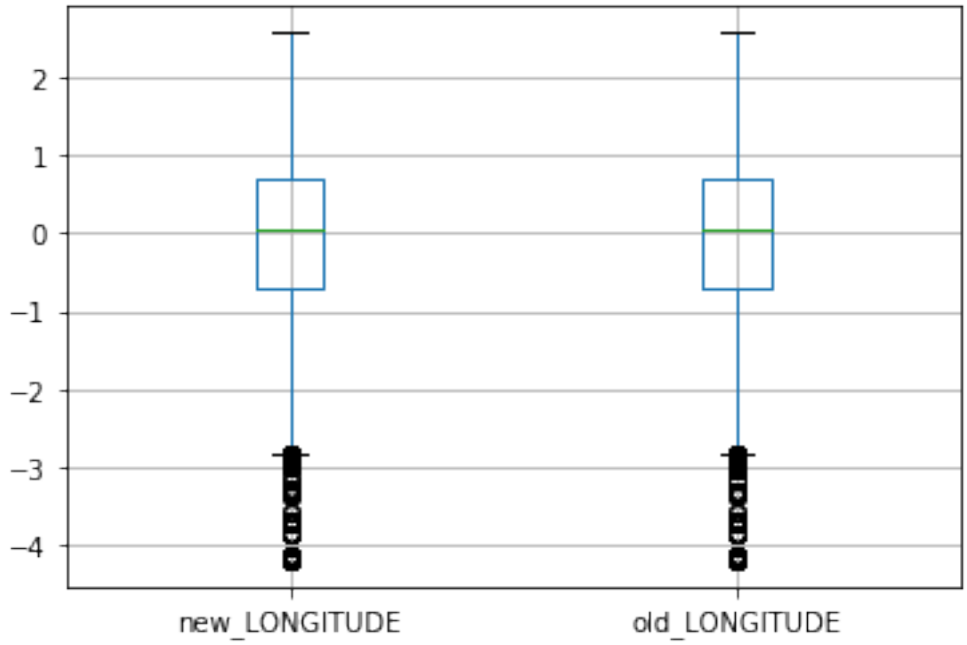
2.用最高频率值来填补缺失值

各数值属性填充前后各属性盒图对比:

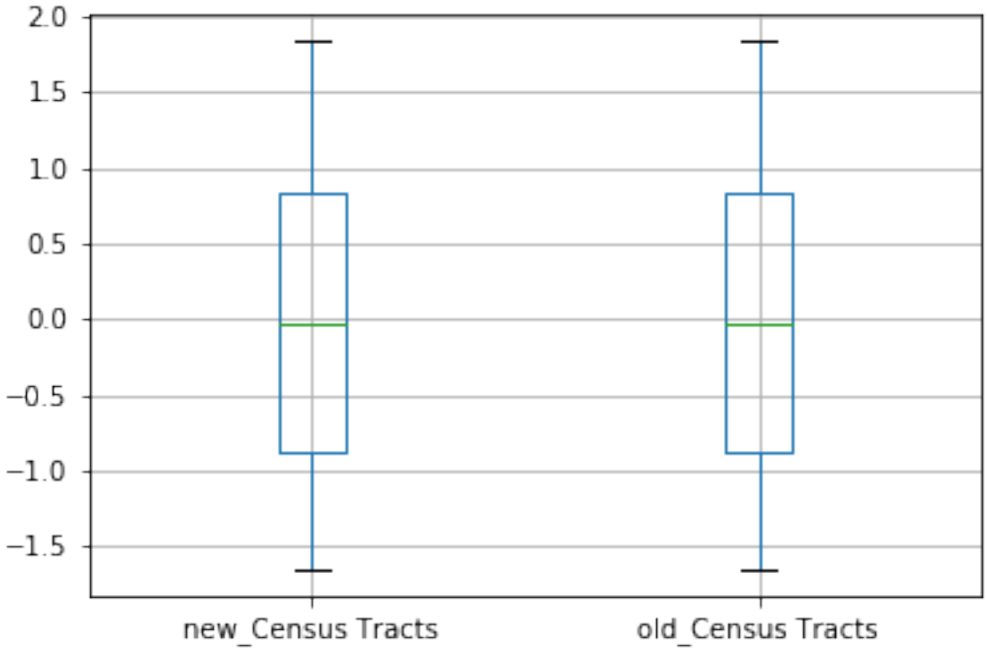
LATITUDE 数据集中最高频率值为 41.914820426999995。



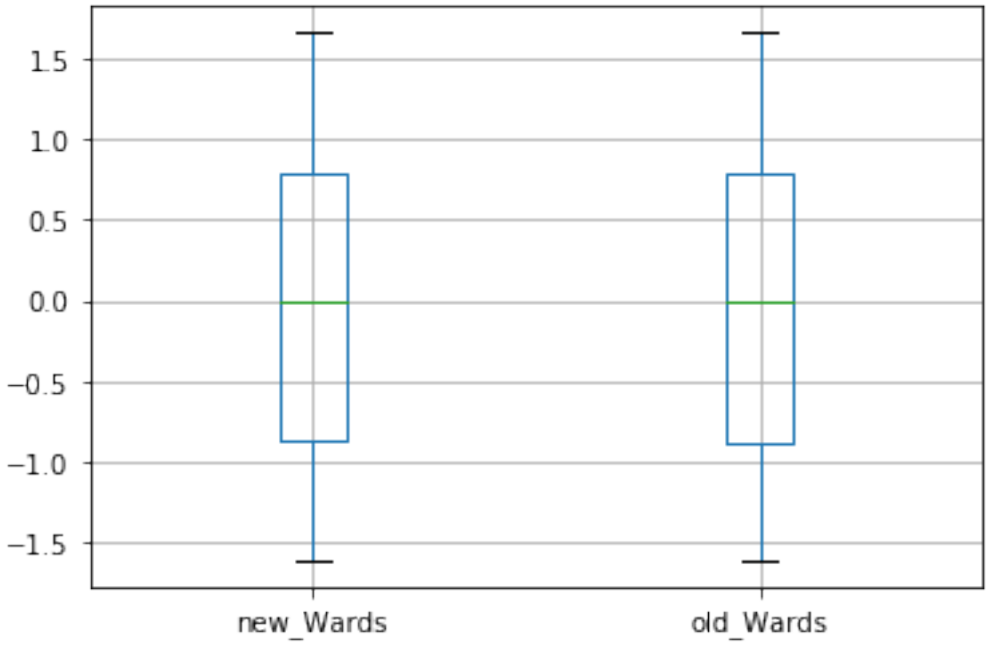

LONGITUDE 数据集中最高频率值为-87.77555967。




Census Tracts 数据集中最高频率值为 92.0。



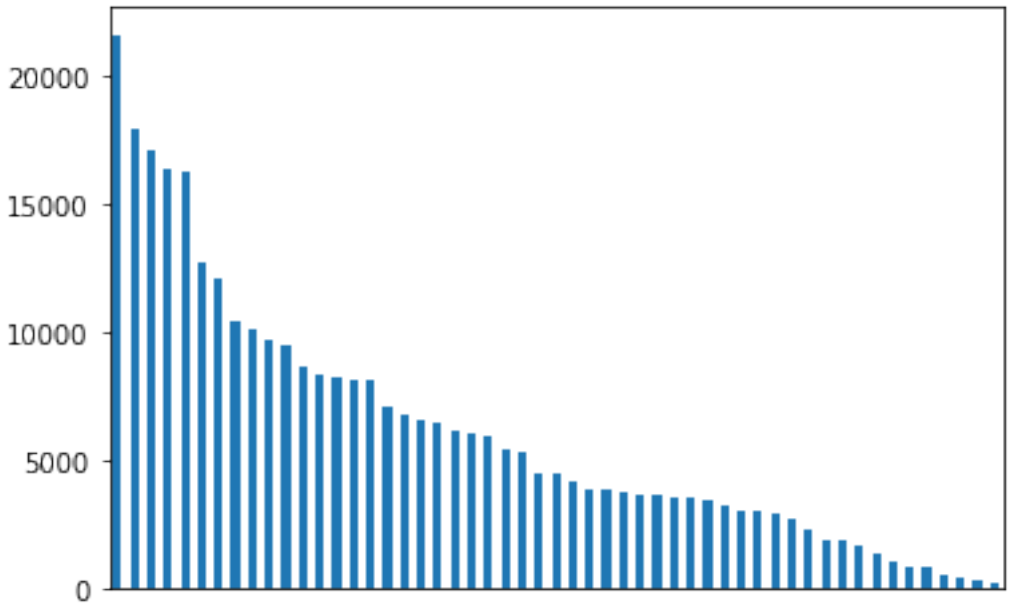
Wards 数据集中最高频率值为 2.0。




Historical Wards 2003-2015 数据集中最高频率值为 36.0。

[15]:
```
print("标称属性 SSA 填充缺失值前后频度图对比:")
print("填充前:")
data['SSA'].value_counts().plot.bar()
plt.xticks([])
plt.show()
print("SSA 数据集中最高频率值为"+str(data['SSA'].mode()[0])+", 用其填充后:")
data['SSA'].fillna(value=data['SSA'].mode()[0]).value_counts().plot.bar()
plt.xticks([])
plt.show()
```

标称属性 SSA 填充缺失值前后频度图对比:
填充前:




SSA 数据集中最高频率值为 51.0, 用其填充后:


[19]:
```
#3.通过属性的相关关系来填补缺失值
print("3.通过属性的相关关系来填补缺失值")
print("首先剔除掉全空数据，然后计算各数值属性间的 Pearson 相关系数矩阵，找到与缺失属性最相似的属性进行线性拟合，最后推测出缺失数据进行填充。")
nan_data=val_data[val_data.isnull().values==True]
nan_data=nan_data.dropna(axis=0,how='all')
cor=val_data.corr()
print("各数值属性间相关系数矩阵:")
corr=cor
corr.rename(index={'Historical Wards 2003-2015':'Historical'}, inplace=True)
corr.rename(columns={'Historical Wards 2003-2015':'Historical'}, inplace=True)
print(corr)
cor=cor.values
nan_title=nan_data.columns.values.tolist()
number=0
for i in nan_title:
    t_data=nan_data[i]
    if(t_data.isnull().sum()>0):
        maxx=-1
        maxn=-1
```

```
        for j in range(len(cor[0])):
            if(j!=number):
                if(abs(cor[j][number])>maxx):
                    maxx=abs(cor[j][number])
                    maxn=j
        clf = linear_model.LinearRegression()
        clf.fit(pd.DataFrame(val_data.dropna()[i].values), val_data.
 ↪dropna()[v_title[maxn]].values)
        for j in range(len(nan_data[i])):
            if(np.isnan(nan_data[i][j:j+1].values[0])):
                if(not(np.isnan(nan_data[v_title[maxn]][j:j+1].values[0]))):
                    nan_data[i][j:j+1].values[0]=clf.predict(np.
 ↪array(nan_data[v_title[maxn]][j:j+1].values[0]).reshape(-1, 1))
                else:
                    nan_data[i][j:j+1].values[0]=val_data[i].mode()[0]
    number=number+1
print("各数值属性填充数据"+str(len(nan_data[i]))+"个。")
print("填充前后各属性盒图对比:")
val_title=val_data.columns.values.tolist()
for i in val_title:
    t_data=val_data[i]
    d_val_data=t_data.dropna()
    d_val_data=pd.concat([d_val_data, nan_data[i]])
    contrast=pd.DataFrame(list(itertools.
 ↪zip_longest(d_val_data,t_data)),columns=['new_'+i, 'old_'+i])
    contrast = (contrast - np.mean(contrast,axis=0)) / np.std(contrast,axis=0)
    boxplot=contrast.boxplot()
    plt.show()
```

3.通过属性的相关关系来填补缺失值

首先剔除掉全空数据，然后计算各数值属性间的 Pearson 相关系数矩阵，找到与缺失属性最相似的属性进行线性拟合，最后推测出缺失数据进行填充。

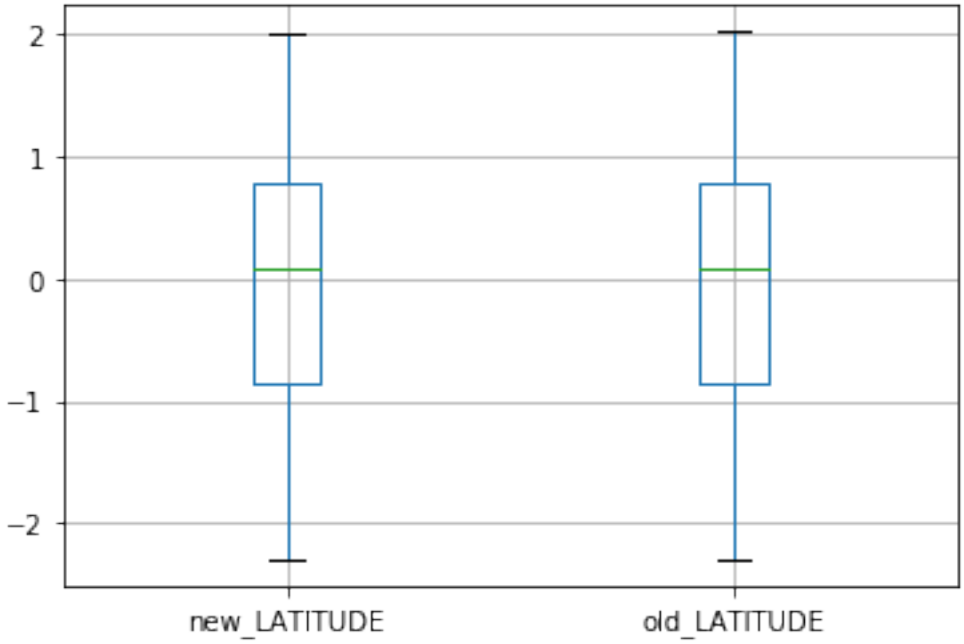
各数值属性间相关系数矩阵:

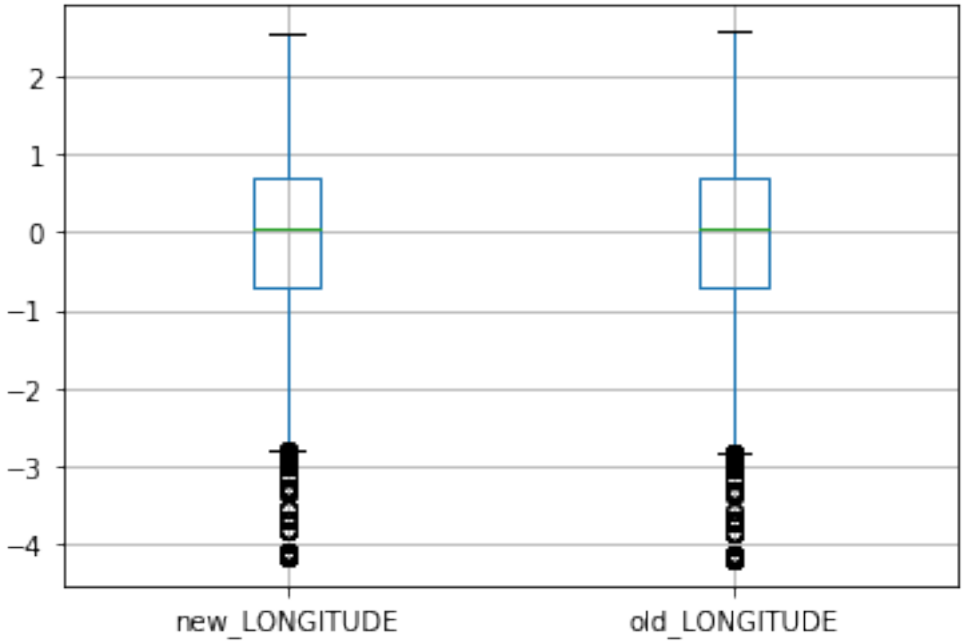
| | LATITUDE | LONGITUDE | Census Tracts | Wards | Historical |
|---|---|---|---|---|---|
| LATITUDE | 1.000000 | -0.575267 | -0.070431 | 0.008941 | -0.155921 |
| LONGITUDE | -0.575267 | 1.000000 | -0.048546 | 0.120544 | 0.164372 |

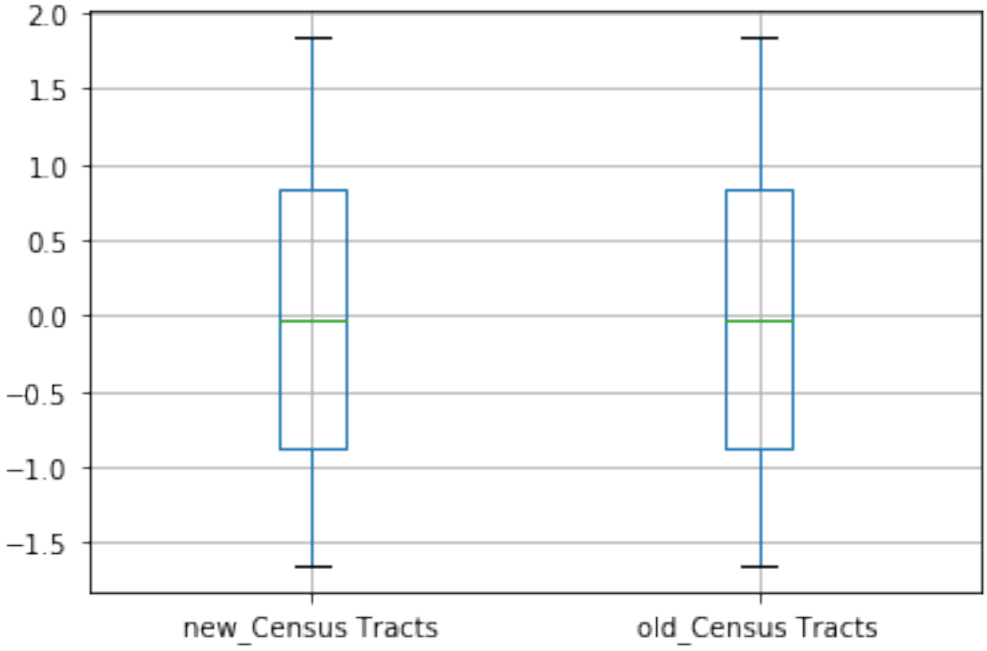
| | | | | | |
|---|---|---|---|---|---|
| Census Tracts | -0.070431 | -0.048546 | 1.000000 | 0.017829 | 0.015609 |
| Wards | 0.008941 | 0.120544 | 0.017829 | 1.000000 | -0.087439 |
| Historical | -0.155921 | 0.164372 | 0.015609 | -0.087439 | 1.000000 |

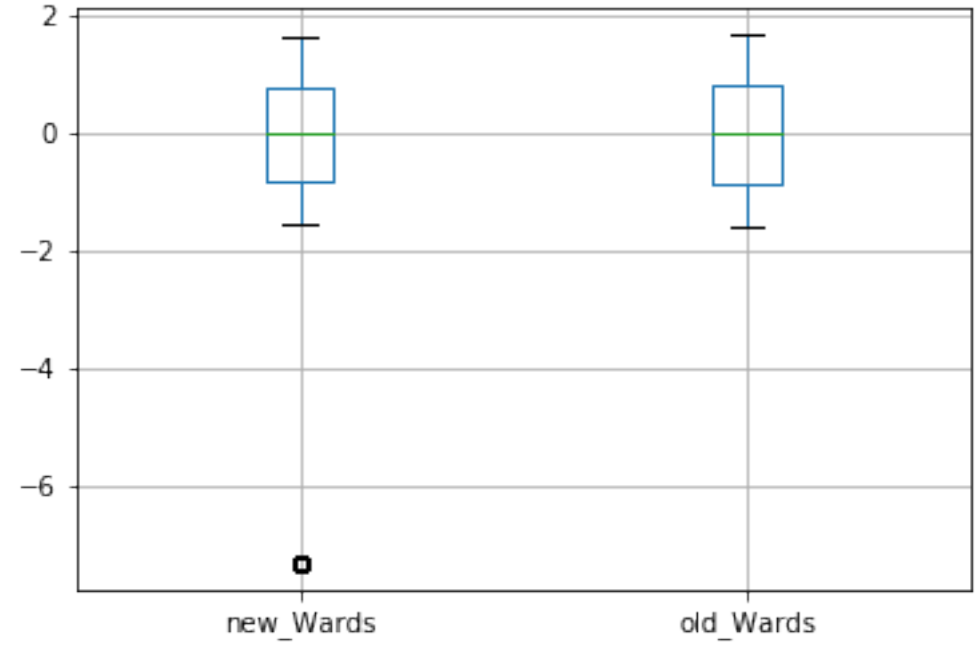
各数值属性填充数据 1573 个。

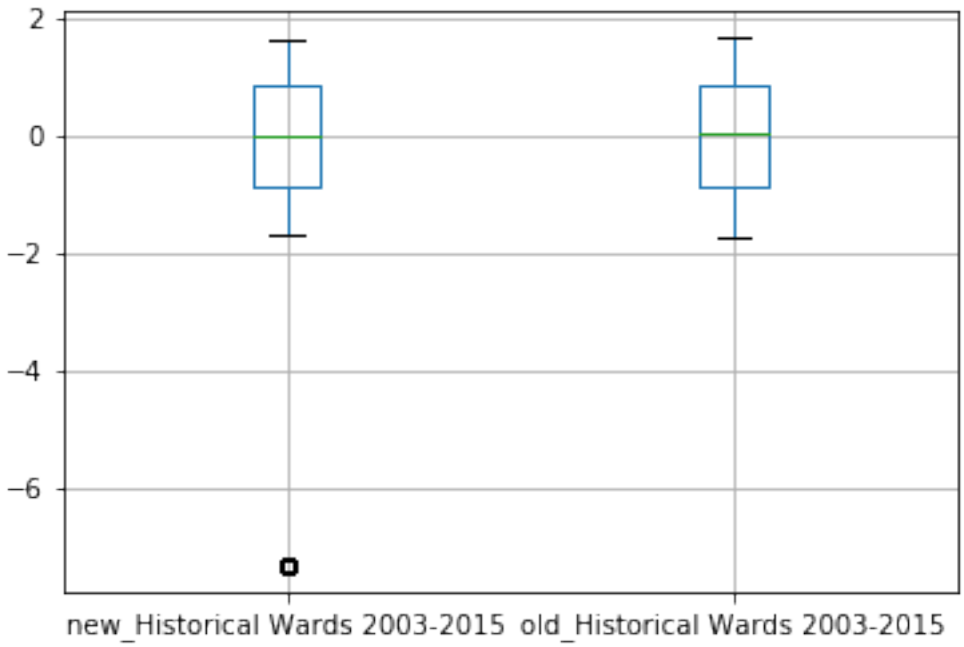
填充前后各属性盒图对比:




2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
new_Census Tracts
old_Census Tracts


2
0
−2
−4
−6
new_Wards
old_Wards




[20]:
```
#4.通过数据对象之间的相似性来填补缺失值
print("4.通过数据对象之间的相似性来填补缺失值")
print("首先剔除掉全空数据，然后找出与缺失数据其余属性间距离最近的项，按照该项属性值进行缺失值填充。")
nan_data=val_data[val_data.isnull().values==True]
nan_data=nan_data.dropna(axis=0,how='all')
nan_data.drop_duplicates(keep='first',inplace=True)
nan_title=nan_data.columns.values.tolist()
print("开始填充")
for i in nan_title:
    dic={}
    for j in range(len(nan_data[i])):
        if(np.isnan(nan_data[i][j:j+1].values[0])):
            a1=-1 if np.isnan(nan_data[v_title[0]][j:j+1].values[0]) else␣
↪nan_data[v_title[0]][j:j+1].values[0]
            a2=-1 if np.isnan(nan_data[v_title[1]][j:j+1].values[0]) else␣
↪nan_data[v_title[1]][j:j+1].values[0]
            a3=-1 if np.isnan(nan_data[v_title[2]][j:j+1].values[0]) else␣
↪nan_data[v_title[2]][j:j+1].values[0]
```

```
            a4=-1 if np.isnan(nan_data[v_title[3]][j:j+1].values[0]) else␣
↪nan_data[v_title[3]][j:j+1].values[0]
            a5=-1 if np.isnan(nan_data[v_title[4]][j:j+1].values[0]) else␣
↪nan_data[v_title[4]][j:j+1].values[0]
            if (a1,a2,a3,a4,a5) in dic:
                nan_data[i][j:j+1]=dic[(a1,a2,a3,a4,a5)]
            else:
                temp=nan_data[nan_data[i].isnull().values==False]
                mins=99999999
                re=-1
                for k in range(len(temp)):
                    a=nan_data.iloc[[j]]
                    b=temp.iloc[[k]]
                    s=0
                    for l in v_title:
                        if(not(np.isnan(a[l].values[0]) or np.isnan(b[l].
↪values[0]))):
                            s=s+math.pow(a[l].values[0]-b[l].values[0],2)
                    tt=s
                    if(tt<mins):
                        mins=tt
                        re=temp.iloc[[1]][i].values[0]
                if(not(re==-1)):
                    nan_data[i][j:j+1]=re
                    dic[(a1,a2,a3,a4,a5)]=re
    print("属性"+i+"填充完毕")
print("共填充"+str(len(nan_data))+"个数据")
print("填充前后各属性盒图对比:")
val_title=val_data.columns.values.tolist()
for i in val_title:
    t_data=val_data[i]
    d_val_data=t_data.dropna()
    d_val_data=pd.concat([d_val_data, nan_data[i]])
    contrast=pd.DataFrame(list(itertools.
↪zip_longest(d_val_data,t_data)),columns=['new_'+i, 'old_'+i])
    contrast = (contrast - np.mean(contrast,axis=0)) / np.std(contrast,axis=0)
```

```
    boxplot=contrast.boxplot()
    plt.show()
```

4.通过数据对象之间的相似性来填补缺失值

首先剔除掉全空数据，然后找出与缺失数据其余属性间距离最近的项，按照该项属性值进行缺失值填充。

开始填充

属性 LATITUDE 填充完毕

属性 LONGITUDE 填充完毕

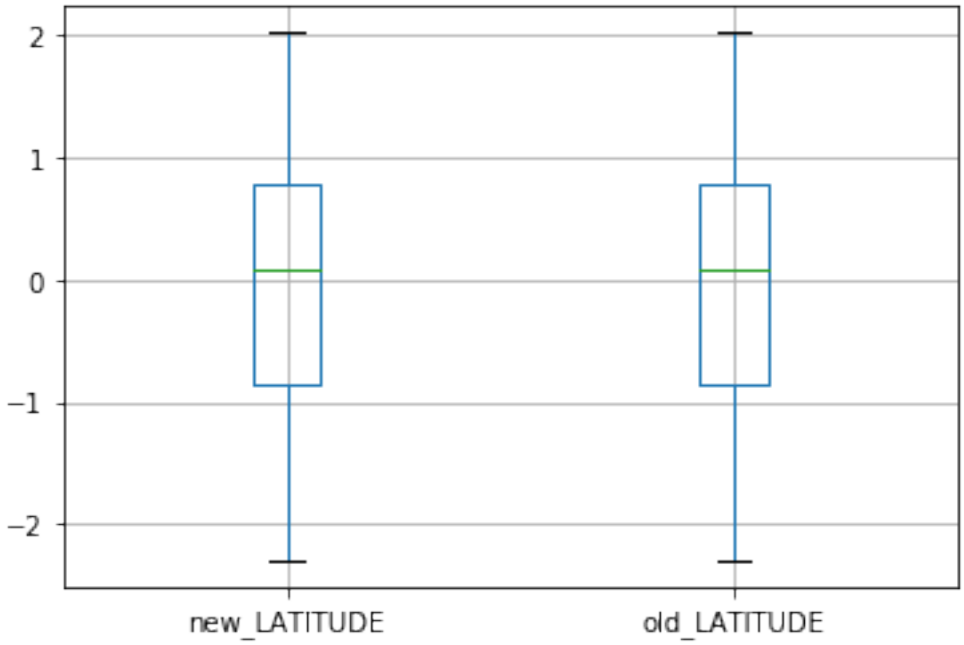
属性 Census Tracts 填充完毕

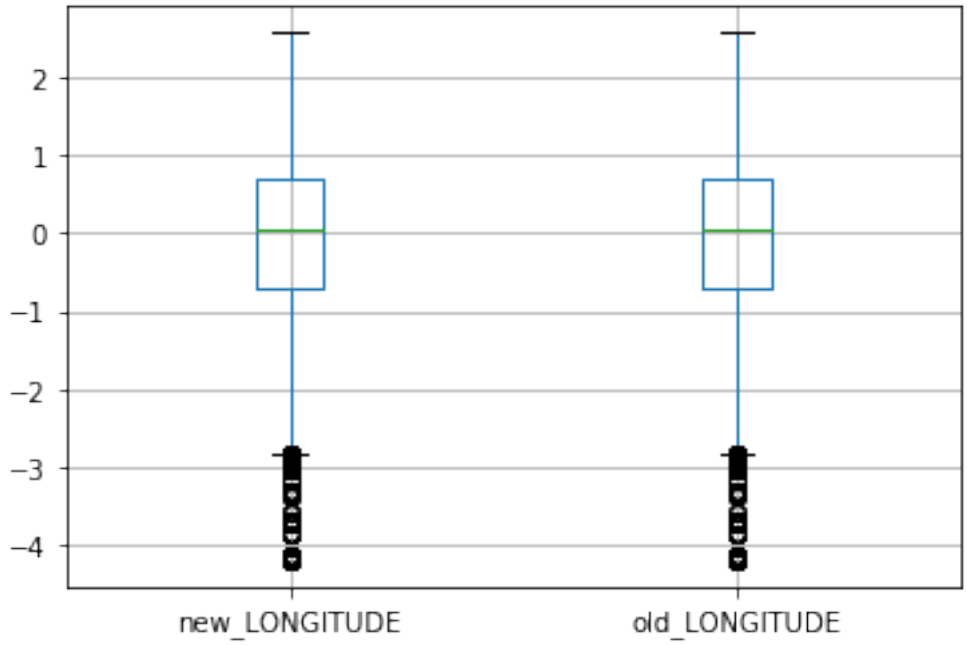
属性 Wards 填充完毕

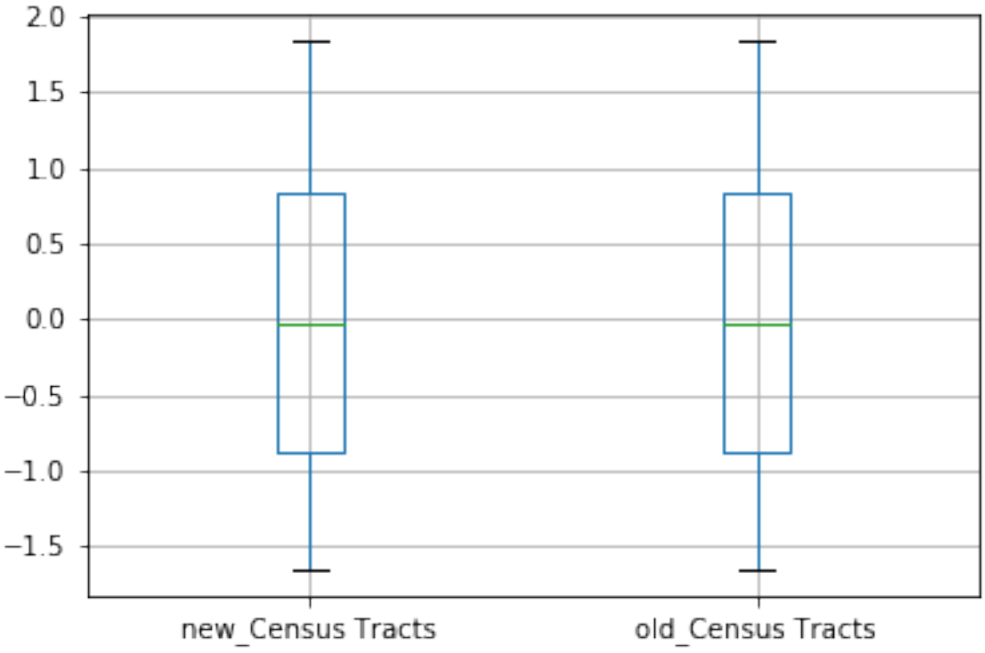
属性 Historical Wards 2003-2015 填充完毕

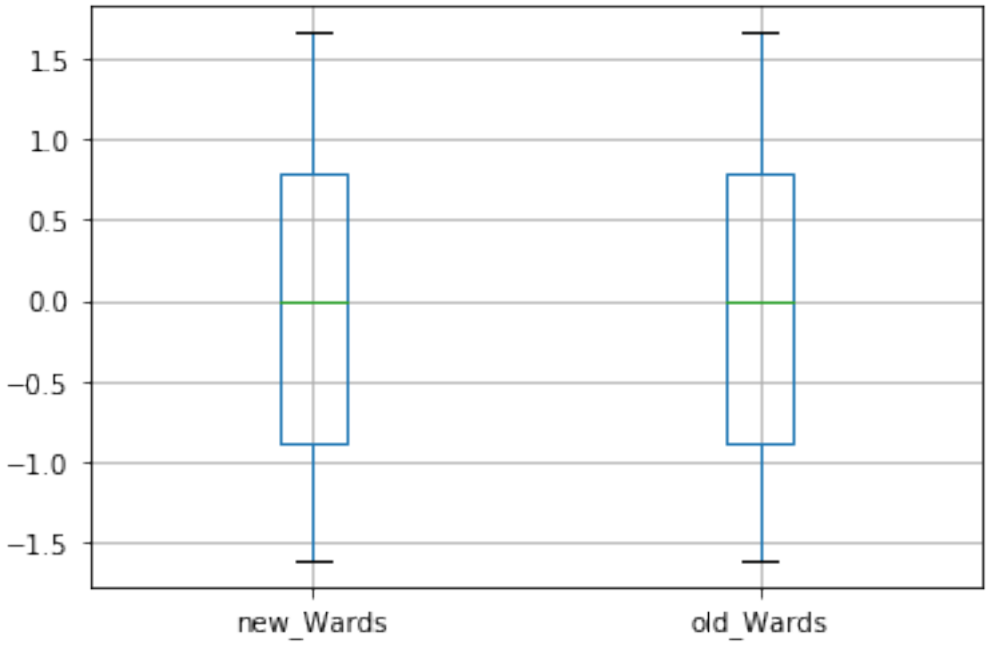
共填充 16 个数据

填充前后各属性盒图对比:




2
1
0
−1
−2
−3
−4
new_LONGITUDE
old_LONGITUDE


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
new_Census Tracts
old_Census Tracts





[21]:
```
print("由于缺失数据项相比于全部数据项较少，所以处理前后数据盒图变化不明显。")
```

由于缺失数据项相比于全部数据项较少，所以处理前后数据盒图变化不明显。

# Second_DataMining

2020 年 5 月 3 日

[16]:
```
#data set: Consumer & Visitor Insights For Neighborhoods
#There are 13 columns of data in the dataset

#load data
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import  linear_model
import itertools
import math
```

[2]:
```
print("读取数据中...")
data = pd.read_csv('cbg_patterns.csv')
print("读取完毕。")
print("属性有: census_block_group、date_range_start、date_range_end、
raw_visit_count、raw_visitor_count、visitor_home_cbgs、visitor_work_cbgs、
distance_from_home、related_same_day_brand、related_same_month_brand、top_brands、
popularity_by_hour、popularity_by_day")
```

```
读取数据中...
读取完毕。
属性有: census_block_group、date_range_start、date_range_end、raw_visit_count、
raw_visi
tor_count、visitor_home_cbgs、visitor_work_cbgs、distance_from_home、
related_same_da
y_brand、related_same_month_brand、top_brands、popularity_by_hour、
popularity_by_day
```

[3]:
```
# 提取数据集中数值属性
v_title=["raw_visit_count","raw_visitor_count","distance_from_home"]
val_data=data[v_title]
print("数值属性有:raw_visit_count、raw_visitor_count、distance_from_home;")
```

```
数值属性有:raw_visit_count、raw_visitor_count、distance_from_home;
```

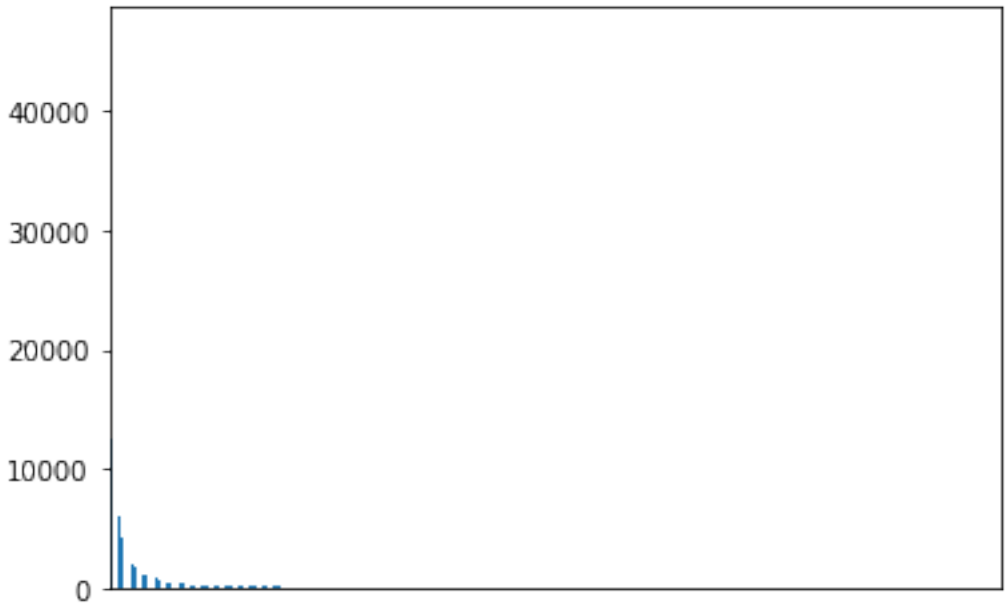
[4]:
```
# 提取数据集中标称属性
c_title=["related_same_day_brand"]
print("标称属性有:census_block_group、date_range_start、date_range_end、
visitor_home_cbgs、visitor_work_cbgs、related_same_day_brand、
related_same_month_brand、top_brands、popularity_by_hour、popularity_by_day;")
```

```
标称属性有:census_block_group、date_range_start、date_range_end、visitor_home_cbgs、
visit
or_work_cbgs、related_same_day_brand、related_same_month_brand、top_brands、
populari
ty_by_hour、popularity_by_day;
```

[5]:
```
# 列出标称属性 related_same_day_brand 的频度图
print("标称属性 related_same_day_brand 频度图 (隐去横坐标，并且由于各种类商品数据
并不是直接给出，所以先转换格式后再统计个数):
 ↪")
a=data["related_same_day_brand"]
t=[]
for i in range(len(a)):
    b=a.iloc[[i]].values[0]
    c=b[1:len(b)-1]
    if(not(len(c)==0)):
        d=c.split(',')
        for j in range(len(d)):
            e=d[j][1:len(d[j])-1]
            t.append(e)
series= pd.Series(t)
series.value_counts().plot.bar()
plt.xticks([])
```

```
plt.show()
```

标称属性 related_same_day_brand 频度图（隐去横坐标，并且由于各种类商品数据并不是直接给出，所以先转换格式后再统计个数）：




[6]:
```
# 列出每个数值属性的五数
print("各个数值属性五数概括:")
val_title=val_data.columns.values.tolist()
for i in val_title:
    t_data=val_data[i]
    print(i+":")
    print("\tMin:\t"+str(np.nanmin(t_data)))
    print("\tQ1:\t"+str(np.nanpercentile(t_data,25)))
    print("\tMedian:\t"+str(np.nanmedian(t_data)))
    print("\tQ3:\t"+str(np.nanpercentile(t_data,75)))
    print("\tMax:\t"+str(np.nanmax(t_data)))
    print("")
```

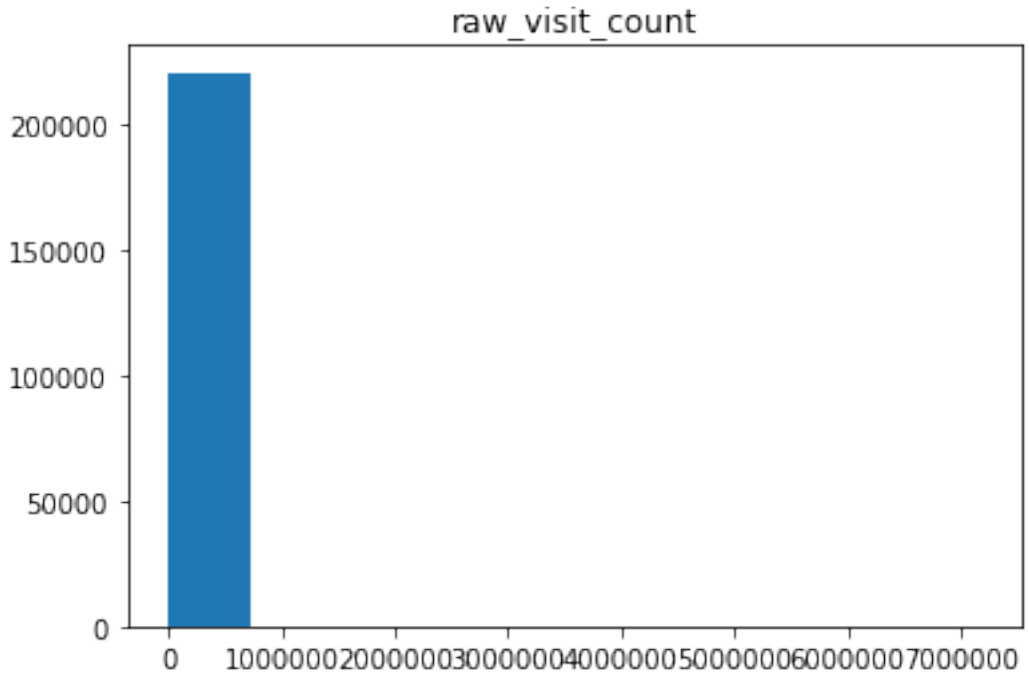
各个数值属性五数概括:

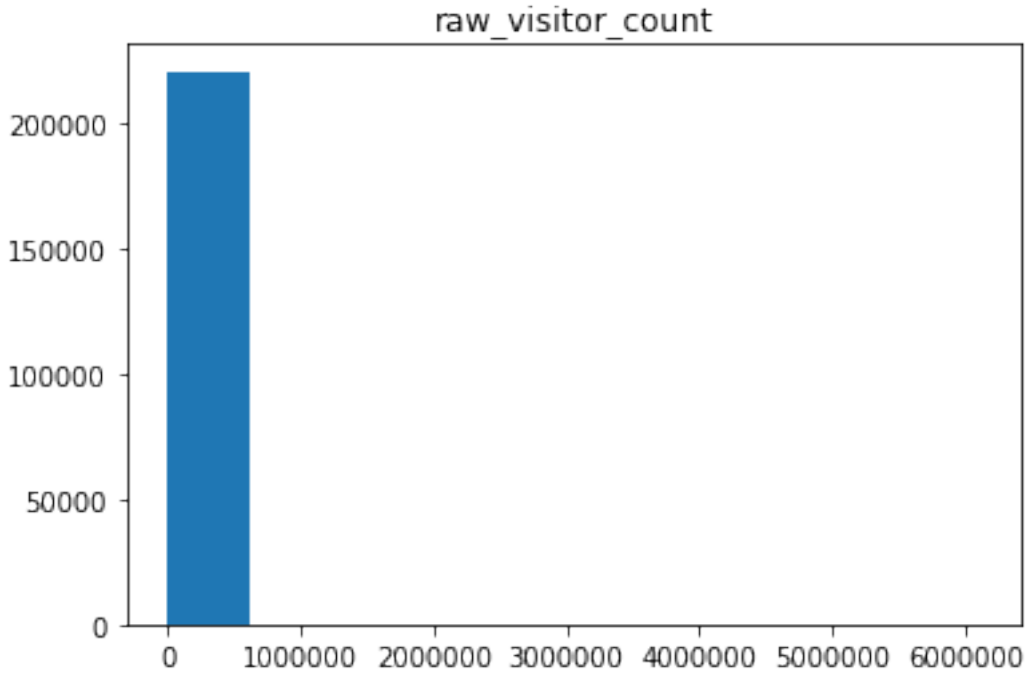
raw_visit_count:

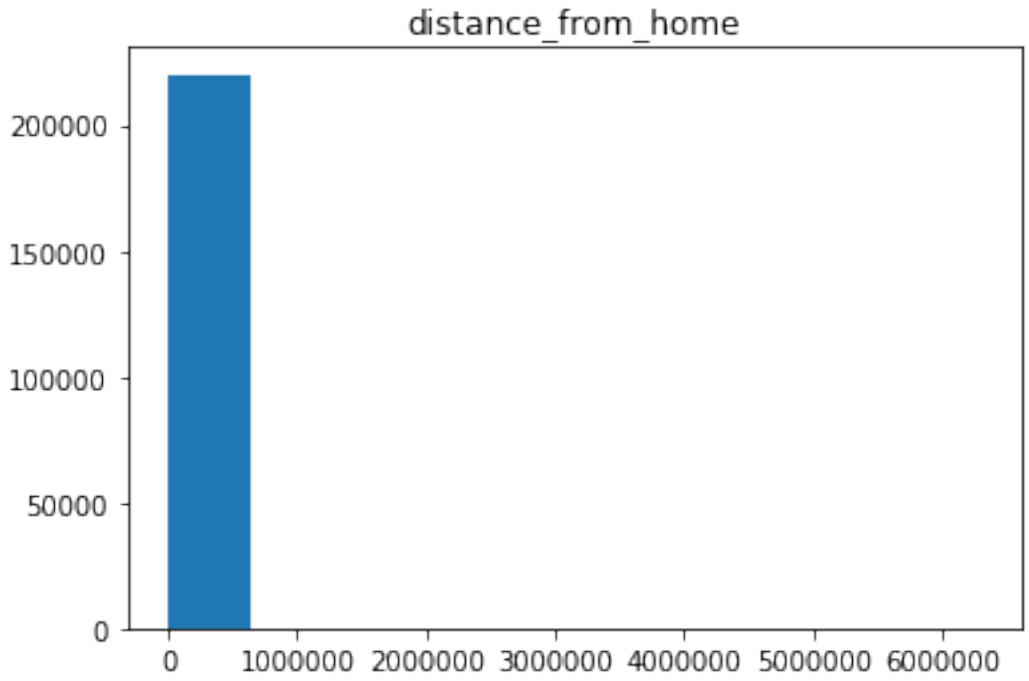
```
        Min:    60.0
        Q1:     17042.0
        Median: 30640.0
        Q3:     56678.0
        Max:    7179900.0

raw_visitor_count:
        Min:    50.0
        Q1:     3430.0
        Median: 6541.0
        Q3:     13099.0
        Max:    6113949.0

distance_from_home:
        Min:    706.0
        Q1:     8584.0
        Median: 14614.0
        Q3:     31397.75
        Max:    6297845.0
```

[7]:
```
# 分别画出每个数据直方图
print("各个数值属性直方图:")
val_title=val_data.columns.values.tolist()
for i in val_title:
    val_data.hist(grid=False,column=i)
    plt.show()
```

各个数值属性直方图:

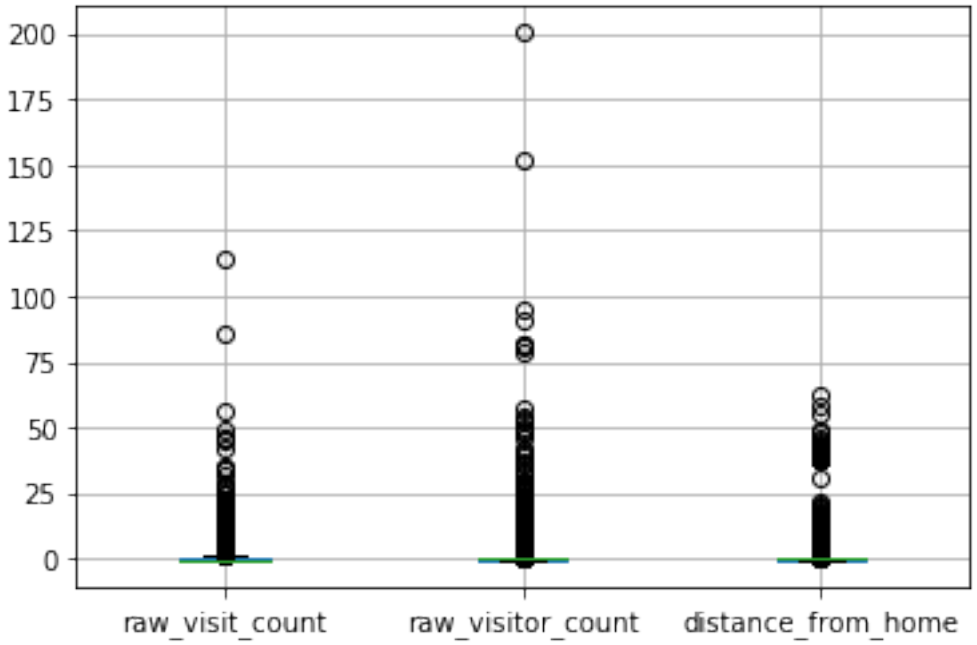

raw_visit_count
200000
150000
100000
50000
0
0
1000000
2000000
3000000
4000000
5000000
6000000
7000000


raw_visitor_count
200000
150000
100000
50000
0
0
1000000
2000000
3000000
4000000
5000000
6000000




[8]:
```
# 分别画出每个数据盒图
print("各个数值属性盒图:")
temp_data=val_data
temp_data = (temp_data - np.mean(temp_data,axis=0)) / np.std(temp_data,axis=0)
boxplot=temp_data.boxplot()
plt.show()
```

各个数值属性盒图:




[9]:
```
# 属性离群点
print("离群点:由属性的盒图可以初步看出每个属性都存在大量超过上盒子上限的离群点。")
```

离群点:由属性的盒图可以初步看出每个属性都存在大量超过上盒子上限的离群点。

[10]:
```
# 列出每个属性缺失值数量
print("各属性缺失值概括:")
nan_number=data.isnull().sum()
print(nan_number)
# 分析数值属性缺失原因
print("数值属性缺失值原因分析:该数据集缺失项除了标称属性 census_block_group 中的 1 项，其余均为数值属性，进一步观察发现大部分的缺失项都同时缺少这三种数值属性，所以考虑是由于录入数据时导致的数据缺失。")
```

各属性缺失值概括:

```
census_block_group          1
date_range_start            0
date_range_end              0
raw_visit_count           106
raw_visitor_count         106
```

```
visitor_home_cbgs             0
visitor_work_cbgs             0
distance_from_home          217
related_same_day_brand        0
related_same_month_brand      0
top_brands                    0
popularity_by_hour            0
popularity_by_day             0
dtype: int64
```

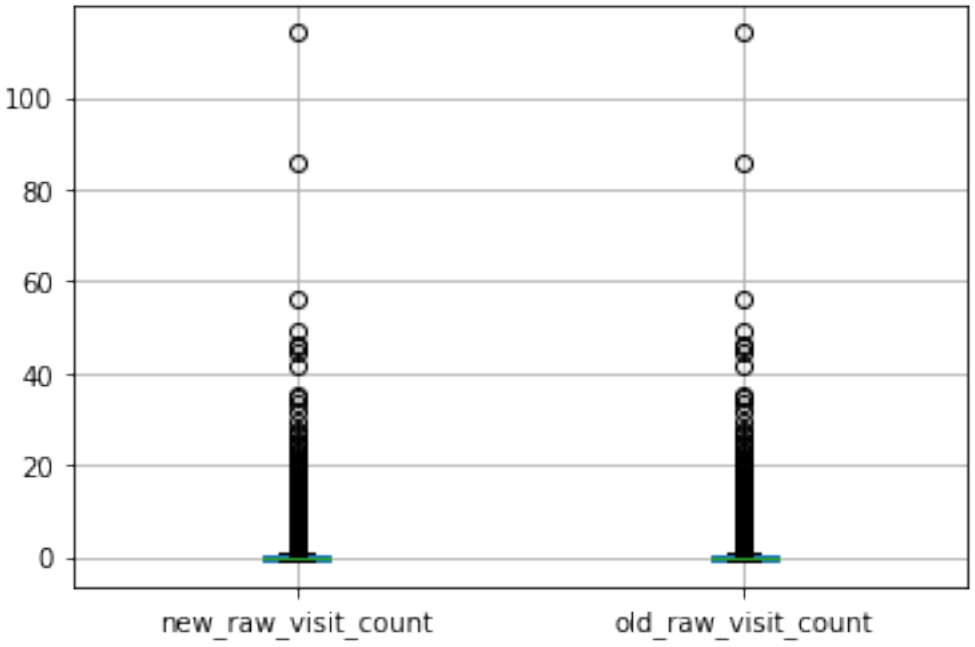
数值属性缺失值原因分析:该数据集缺失项除了标称属性 census_block_group 中的 1 项，其余均为数值属性，进一步观察发现大部分的缺失项都同时缺少这三种数值属性，所以考虑是由于录入数据时导致的数据缺失。

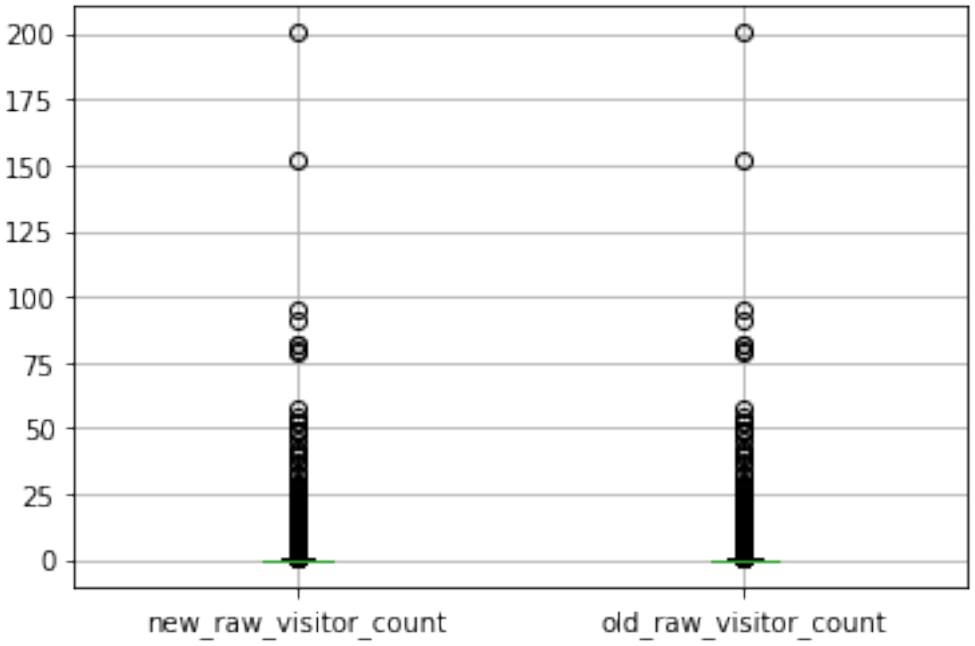
[11]:
```
# 缺失值填充
print("缺失值填充")
#1.将缺失部分剔除
print("1.将缺失部分剔除")
print("各数值属性剔除缺失值前后各属性盒图对比:")
val_title=val_data.columns.values.tolist()
for i in val_title:
    t_data=val_data[i]
    d_val_data=t_data.dropna()
    contrast=pd.DataFrame(list(itertools.
 ↪zip_longest(d_val_data,t_data)),columns=['new_'+i, 'old_'+i])
    contrast = (contrast - np.mean(contrast,axis=0)) / np.std(contrast,axis=0)
    boxplot=contrast.boxplot()
    plt.show()
```

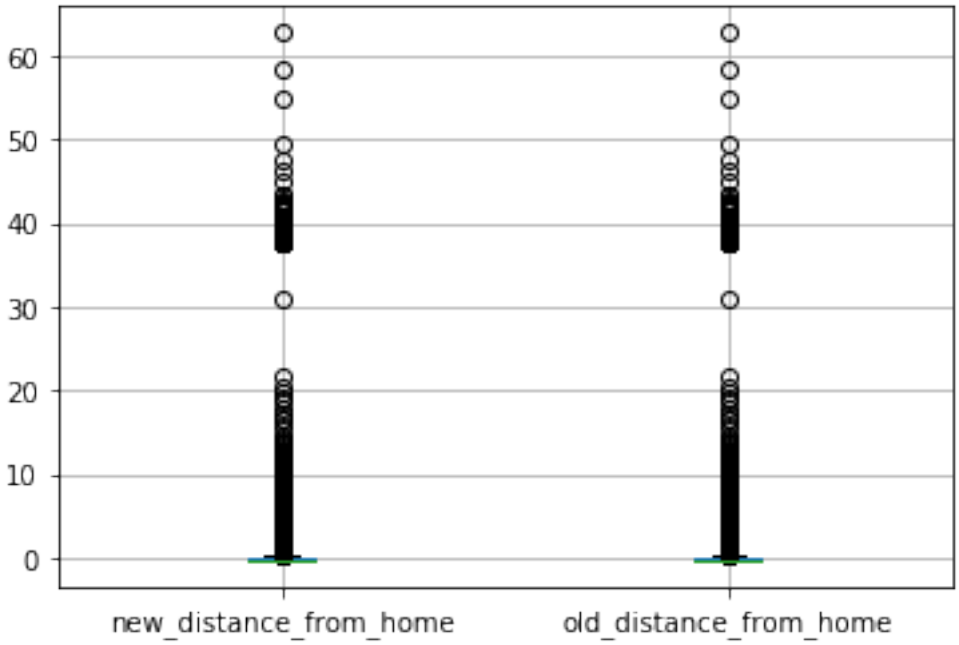
缺失值填充

1.将缺失部分剔除

各数值属性剔除缺失值前后各属性盒图对比:

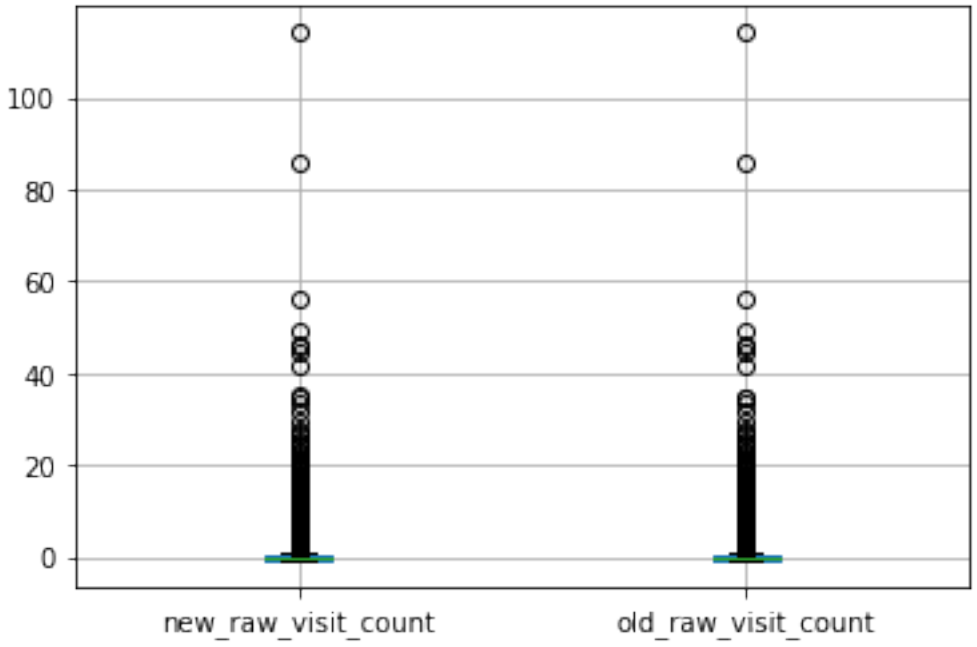

100
80
60
40
20
0
new_raw_visit_count
old_raw_visit_count


200
175
150
125
100
75
50
25
0
new_raw_visitor_count
old_raw_visitor_count

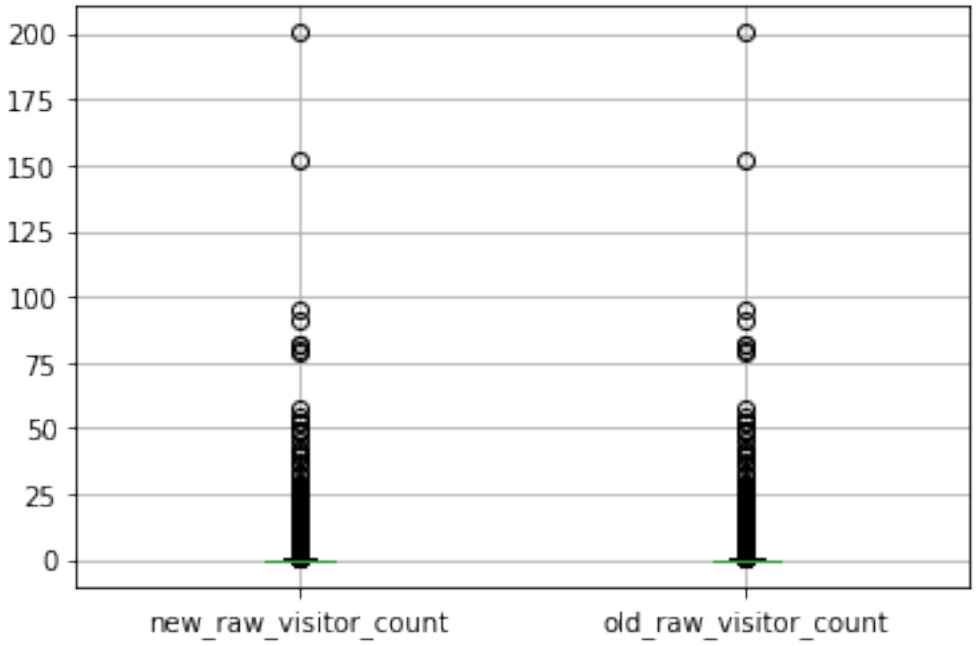



[12]:
```
#2.用最高频率值来填补缺失值
print("2.用最高频率值来填补缺失值")
print("各数值属性填充前后各属性盒图对比:")
val_title=val_data.columns.values.tolist()
for i in val_title:
    t_data=val_data[i]
    print(i+"数据集中最高频率值为"+str(t_data.mode()[0])+"。")
    d_val_data=t_data.fillna(value=t_data.mode()[0])
    contrast=pd.DataFrame(list(itertools.
 ↪zip_longest(d_val_data,t_data)),columns=['new_'+i, 'old_'+i])
    contrast = (contrast - np.mean(contrast,axis=0)) / np.std(contrast,axis=0)
    boxplot=contrast.boxplot()
    plt.show()
```

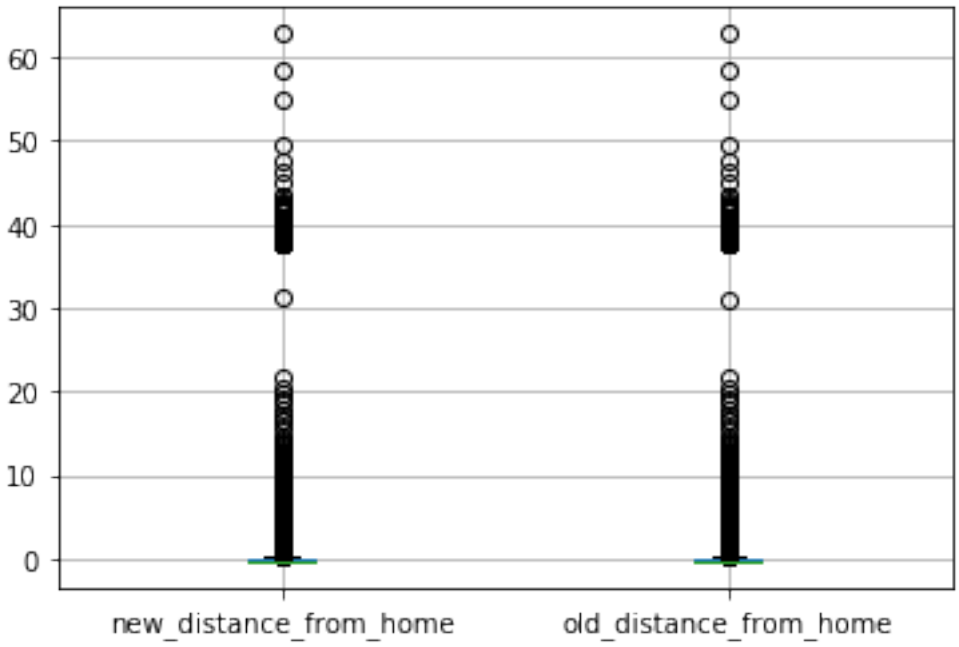
2.用最高频率值来填补缺失值
各数值属性填充前后各属性盒图对比:
raw_visit_count 数据集中最高频率值为 24531.0。




raw_visitor_count 数据集中最高频率值为 2292.0。



distance_from_home 数据集中最高频率值为 8345.0。




[13]:
```
#3.通过属性的相关关系来填补缺失值
print("3.通过属性的相关关系来填补缺失值")
print("首先剔除掉全空数据，然后计算各数值属性间的 Pearson 相关系数矩阵，找到与缺失属性最相似的属性进行线性拟合，最后推测出缺失数据进行填充。")
nan_data=val_data[val_data.isnull().values==True]
nan_data=nan_data.dropna(axis=0,how='all')
cor=val_data.corr()
print("各数值属性间相关系数矩阵:")
print(cor)
cor=cor.values
nan_title=nan_data.columns.values.tolist()
number=0
for i in nan_title:
    t_data=nan_data[i]
    if(t_data.isnull().sum()>0):
        maxx=-1
        maxn=-1
        for j in range(len(cor[0])):
```

```
            if(j!=number):
                if(abs(cor[j][number])>maxx):
                    maxx=abs(cor[j][number])
                    maxn=j
        clf = linear_model.LinearRegression()
        clf.fit(pd.DataFrame(val_data.dropna()[i].values), val_data.
 ↪dropna()[v_title[maxn]].values)
        for j in range(len(nan_data[i])):
            if(np.isnan(nan_data[i][j:j+1].values[0])):
                if(not(np.isnan(nan_data[v_title[maxn]][j:j+1].values[0]))):
                    nan_data[i][j:j+1].values[0]=clf.predict(np.
 ↪array(nan_data[v_title[maxn]][j:j+1].values[0]).reshape(-1, 1))
                else:
                    nan_data[i][j:j+1].values[0]=val_data[i].mode()[0]
    number=number+1
print("各数值属性填充数据"+str(len(nan_data[i]))+"个。")
print("填充前后各属性盒图对比:")
val_title=val_data.columns.values.tolist()
for i in val_title:
    t_data=val_data[i]
    d_val_data=t_data.dropna()
    d_val_data=pd.concat([d_val_data, nan_data[i]])
    contrast=pd.DataFrame(list(itertools.
 ↪zip_longest(d_val_data,t_data)),columns=['new_'+i, 'old_'+i])
    contrast = (contrast - np.mean(contrast,axis=0)) / np.std(contrast,axis=0)
    boxplot=contrast.boxplot()
    plt.show()
```

3.通过属性的相关关系来填补缺失值

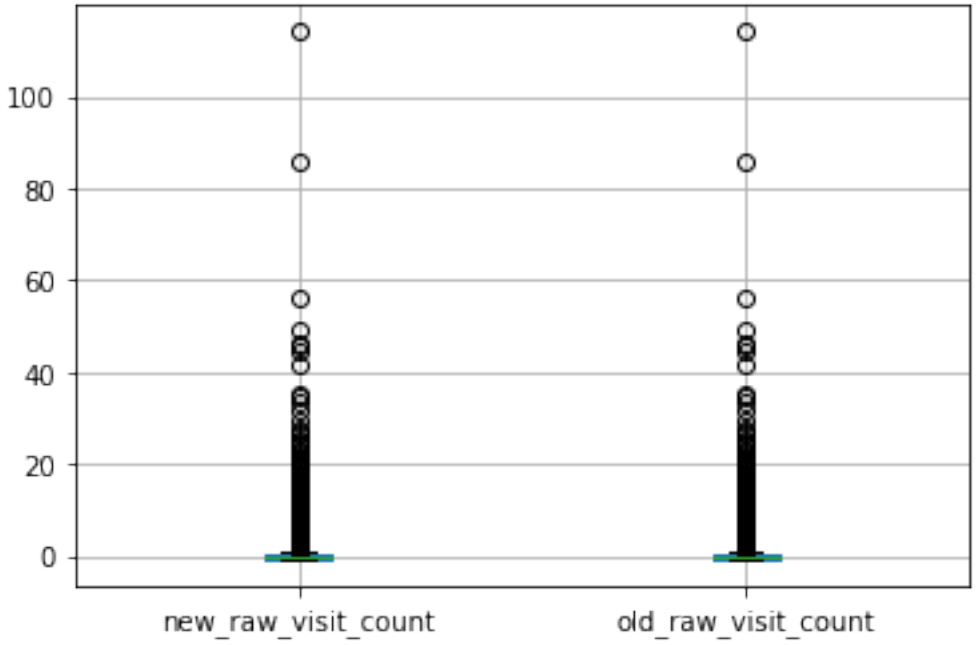
首先剔除掉全空数据，然后计算各数值属性间的 Pearson 相关系数矩阵，找到与缺失属性最相似的属性进行线性拟合，最后推测出缺失数据进行填充。

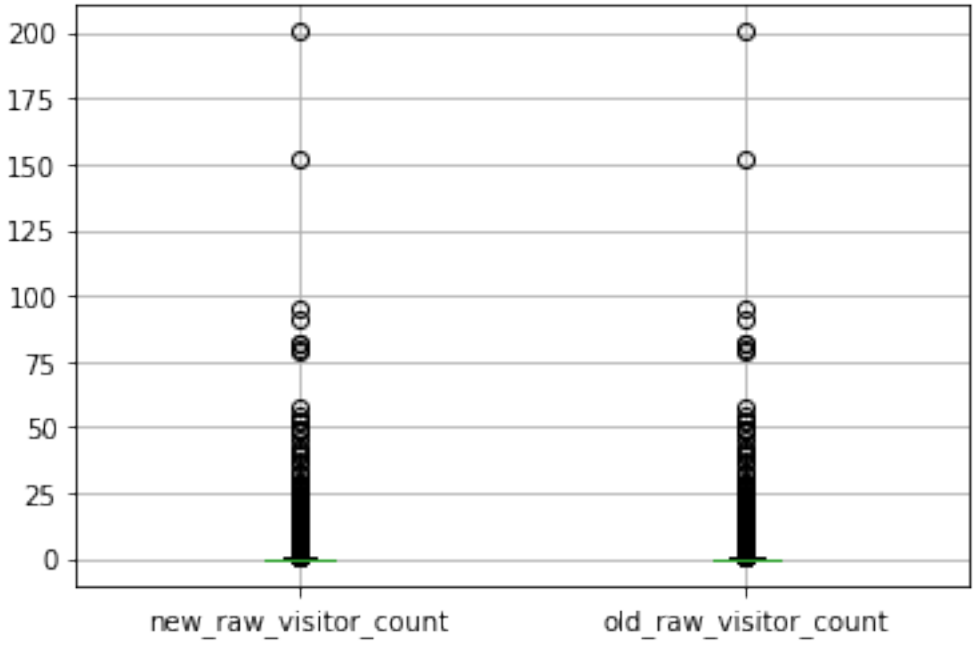
各数值属性间相关系数矩阵:

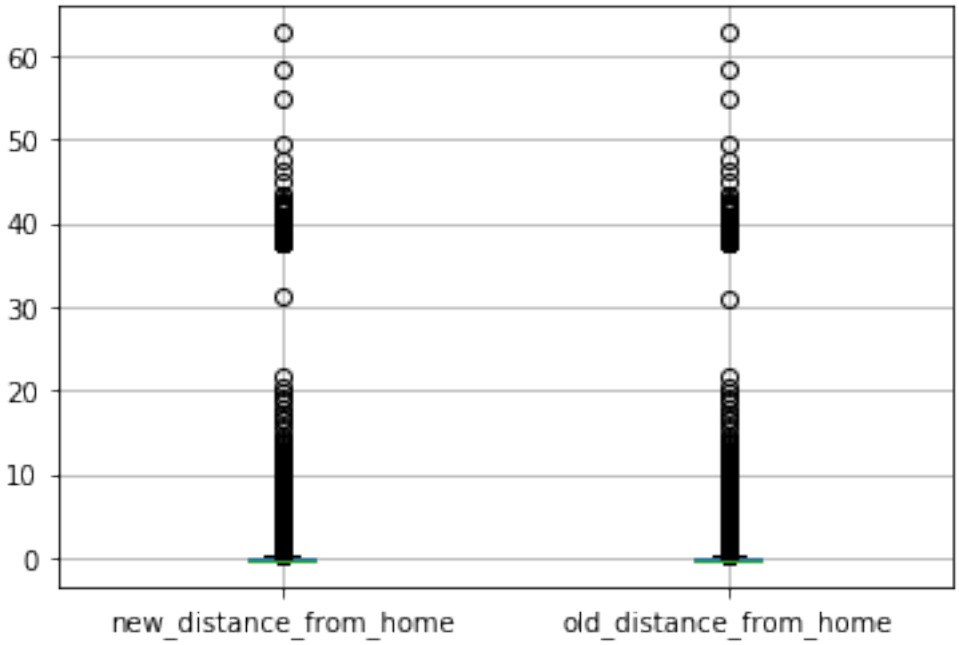
| | raw_visit_count | raw_visitor_count | distance_from_home |
|---|---|---|---|
| raw_visit_count | 1.000000 | 0.815850 | 0.021721 |
| raw_visitor_count | 0.815850 | 1.000000 | 0.032879 |
| distance_from_home | 0.021721 | 0.032879 | 1.000000 |

各数值属性填充数据 111 个。

填充前后各属性盒图对比：






[14]:
```
#4.通过数据对象之间的相似性来填补缺失值
print("4.通过数据对象之间的相似性来填补缺失值")
print("首先剔除掉全空数据，然后找出与缺失数据其余属性间距离最近的项，按照该项属性值进行缺失值填充。")
nan_data=val_data[val_data.isnull().values==True]
nan_data=nan_data.dropna(axis=0,how='all')
nan_data.drop_duplicates(keep='first',inplace=True)
nan_title=nan_data.columns.values.tolist()

print("开始填充")
for i in nan_title:
    dic={}
    for j in range(len(nan_data[i])):
        if(np.isnan(nan_data[i][j:j+1].values[0])):
            a1=-1 if np.isnan(nan_data[v_title[0]][j:j+1].values[0]) else␣
↪nan_data[v_title[0]][j:j+1].values[0]
            a2=-1 if np.isnan(nan_data[v_title[1]][j:j+1].values[0]) else␣
↪nan_data[v_title[1]][j:j+1].values[0]
```

```
            a3=-1 if np.isnan(nan_data[v_title[2]][j:j+1].values[0]) else␣
↪nan_data[v_title[2]][j:j+1].values[0]
            if (a1,a2,a3) in dic:
                nan_data[i][j:j+1]=dic[(a1,a2,a3)]
            else:
                temp=nan_data[nan_data[i].isnull().values==False]
                mins=99999999
                re=-1
                for k in range(len(temp)):
                    a=nan_data.iloc[[j]]
                    b=temp.iloc[[k]]
                    s=0
                    for l in v_title:
                        if(not(np.isnan(a[l].values[0]) or np.isnan(b[l].
↪values[0]))):
                            s=s+math.pow(a[l].values[0]-b[l].values[0],2)
                    tt=s
                    if(tt<mins):
                        mins=tt
                        re=temp.iloc[[1]][i].values[0]
                if(not(re==-1)):
                    nan_data[i][j:j+1]=re
                    dic[(a1,a2,a3)]=re
    print("属性"+i+"填充完毕")
print("共填充"+str(len(nan_data))+"个数据")
print("填充前后各属性盒图对比:")
val_title=val_data.columns.values.tolist()
for i in val_title:
    t_data=val_data[i]
    d_val_data=t_data.dropna()
    d_val_data=pd.concat([d_val_data, nan_data[i]])
    contrast=pd.DataFrame(list(itertools.
↪zip_longest(d_val_data,t_data)),columns=['new_'+i, 'old_'+i])
    contrast = (contrast - np.mean(contrast,axis=0)) / np.std(contrast,axis=0)
    boxplot=contrast.boxplot()
    plt.show()
```

4.通过数据对象之间的相似性来填补缺失值

首先剔除掉全空数据，然后找出与缺失数据其余属性间距离最近的项，按照该项属性值进行缺失值填充。

开始填充

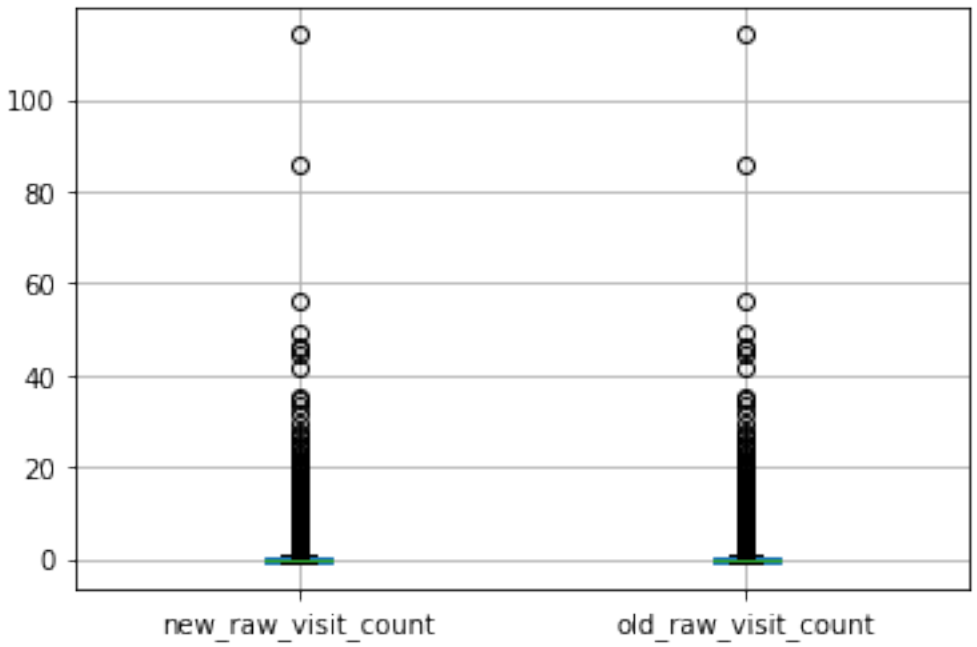
属性 raw_visit_count 填充完毕

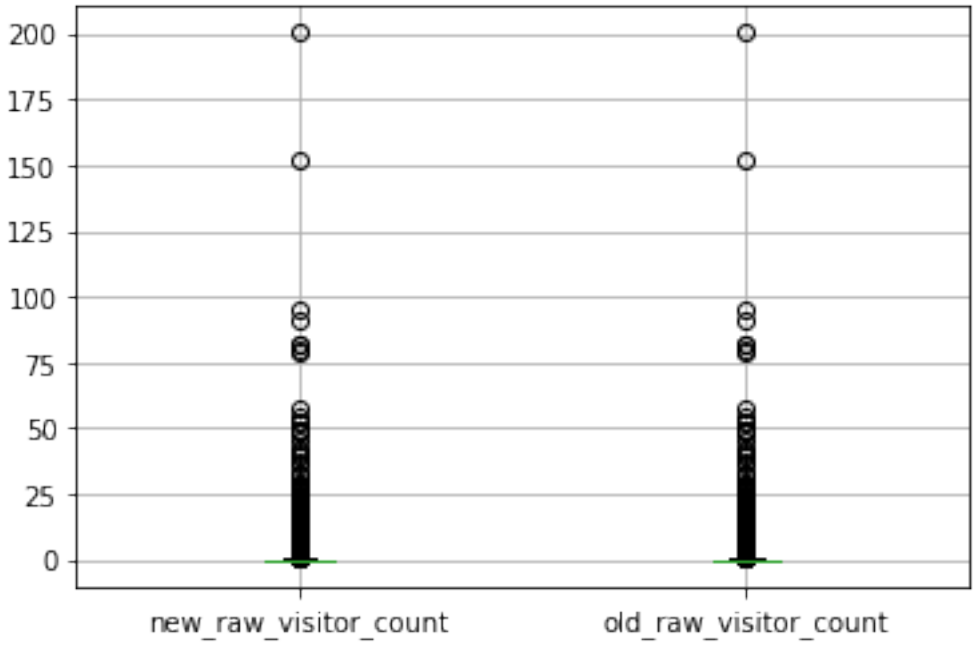
属性 raw_visitor_count 填充完毕

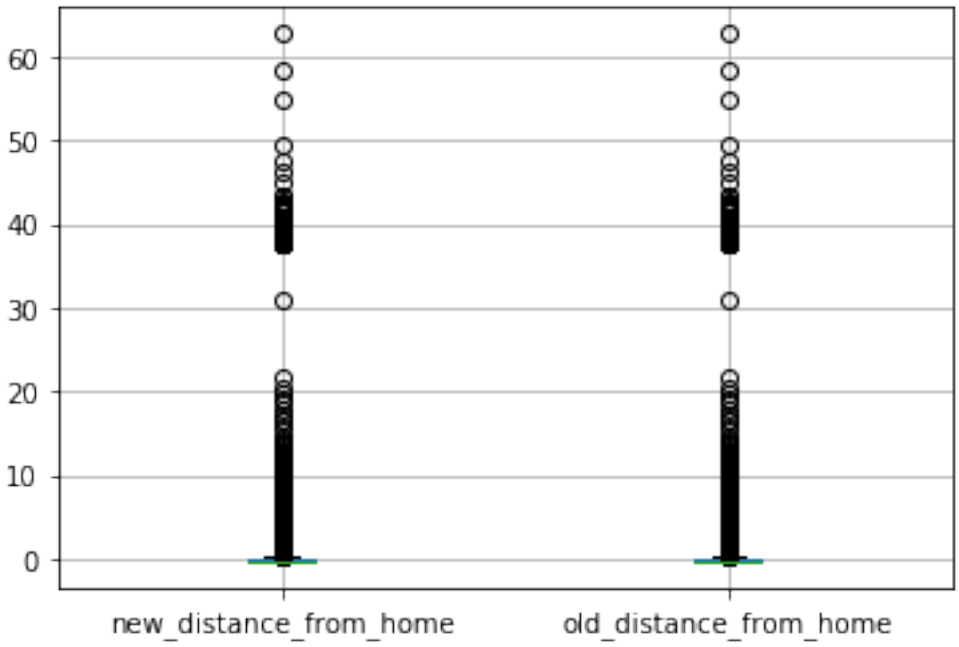
属性 distance_from_home 填充完毕

共填充 111 个数据

填充前后各属性盒图对比:






[15]:
```
print("由于缺失数据项相比于全部数据项较少，所以处理前后数据盒图变化不明显。")
```

由于缺失数据项相比于全部数据项较少，所以处理前后数据盒图变化不明显。