oakland-crime-statistics

2021 年 3 月 28 日

1 数据挖掘作业一

1.1 姓名：余晓学号：3220200998

1.2 数据集：Oakland Crime Statistics 2011 to 2016

[1]:
```
# 导入必要的包
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from scipy import stats
from collections import Counter
from math import isnan
import math
```

1.3 查看数据集并对数据集进行了解

[2]:
```
# 查看当前文件夹下有哪些数据集以及数据集所处的路径
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# 数据集文件解释，这里我们拿 records-for-2011.csv 做数据分析
# 其他文件均为 2011-2016 的犯罪数据
```

```
/kaggle/input/oakland-crime-statistics-2011-to-2016/records-for-2016.csv
/kaggle/input/oakland-crime-statistics-2011-to-2016/socrata_metadata_records-
for-2013.json
/kaggle/input/oakland-crime-statistics-2011-to-2016/socrata_metadata_records-
```

```
for-2012.json
/kaggle/input/oakland-crime-statistics-2011-to-2016/records-for-2013.csv
/kaggle/input/oakland-crime-statistics-2011-to-2016/records-for-2014.csv
/kaggle/input/oakland-crime-statistics-2011-to-2016/socrata_metadata_records-
for-2011.json
/kaggle/input/oakland-crime-statistics-2011-to-2016/records-for-2015.csv
/kaggle/input/oakland-crime-statistics-2011-to-2016/records-for-2011.csv
/kaggle/input/oakland-crime-statistics-2011-to-2016/records-for-2012.csv
/kaggle/input/oakland-crime-statistics-2011-to-2016/socrata_metadata_records-
for-2016.json
/kaggle/input/oakland-crime-statistics-2011-to-2016/socrata_metadata_records-
for-2015.json
/kaggle/input/oakland-crime-statistics-2011-to-2016/socrata_metadata_records-
for-2014.json
```

[3]:
```
# 读取数据集
path = '/kaggle/input/oakland-crime-statistics-2011-to-2016/'
data = pd.read_csv(path+'records-for-2011.csv',index_col=0)
data.head()# 默认展示前五行数据
```

[3]:

Agency	Create Time	Location	Area Id	Beat	Priority	\
OP	2011-01-01T00:00:00.000	ST&SAN PABLO AV	1.0	06X	1.0	
OP	2011-01-01T00:01:11.000	ST&HANNAH ST	1.0	07X	1.0	
OP	2011-01-01T00:01:25.000	ST&MARKET ST	1.0	10Y	2.0	
OP	2011-01-01T00:01:35.000	PRENTISS ST	2.0	21Y	2.0	
OP	2011-01-01T00:02:10.000	AV&FOOTHILL BLVD	2.0	20X	1.0	

Agency	Incident Type Id	Incident Type Description	Event Number	\
OP	PDOA	POSSIBLE DEAD PERSON	LOP110101000001	
OP	415GS	415 GUNSHOTS	LOP110101000002	
OP	415GS	415 GUNSHOTS	LOP110101000003	
OP	415GS	415 GUNSHOTS	LOP110101000005	
OP	415GS	415 GUNSHOTS	LOP110101000004	

Agency	Closed Time

```
Agency
OP      2011-01-01T00:28:17.000
OP      2011-01-01T01:12:56.000
OP      2011-01-01T00:07:20.000
OP      2011-01-01T00:02:28.000
OP      2011-01-01T00:50:04.000
```

[4]:
```
data.dtypes # 每列数据的数据类型
```

[4]:
```
Create Time                   object
Location                      object
Area Id                      float64
Beat                          object
Priority                     float64
Incident Type Id              object
Incident Type Description     object
Event Number                  object
Closed Time                   object
dtype: object
```

[5]:
```
data.shape # 数据集的大小
```

[5]:
```
(180016, 9)
```

2 数据分析要求

2.1 数据可视化和摘要

2.1.1 数据摘要

（1）标称属性，给出每个可能聚会的频数

[6]:
```
# 由上面对数据集各列进行分析得知，该数据集的标称属性有'Create␣
↪Time','Location','Beat','Incident Type Id',
#'Incident Type Description','Event Number','Closed Time' 七个标称属性
# 下面给出每个属性取值的频数
#(1)Create Time
pd.value_counts(data['Create Time'])
```

[6]:
```
2011-06-02T00:00:00.000    4
2011-03-27T00:22:41.000    3
2011-09-21T14:05:59.000    3
2011-07-31T21:12:03.000    2
2011-08-23T22:14:40.000    2
                          ..
2011-06-12T15:12:23.000    1
2011-09-04T03:02:43.000    1
2011-02-02T13:44:12.000    1
2011-10-14T02:26:46.000    1
2011-10-28T12:34:25.000    1
Name: Create Time, Length: 179451, dtype: int64
```

[7]:
```
#(2) Location
pd.value_counts(data['Location'])
```

[7]:
```
 INTERNATIONAL BLVD        3866
 MACARTHUR BLVD            3129
 AV&INTERNATIONAL BLVD     3067
 BROADWAY                  2132
 FOOTHILL BLVD             1791
                           …
NB SUNNYSIDE ST               1
SB RITCHIE ST                 1
46TH AV&TELEGRAPH AV          1
COOLIDGE BALBOA DR            1
12TH 50TH AV                  1
Name: Location, Length: 32505, dtype: int64
```

[8]:
```
#(3) Beat
pd.value_counts(data['Beat'])
```

[8]:
```
04X     7410
08X     6885
26Y     5478
30Y     5295
06X     5119
23X     5051
```

30X	4956
19X	4955
34X	4673
29X	4483
20X	4287
27Y	4159
07X	4134
31Y	4082
25X	4022
35X	3880
33X	3849
03X	3819
32X	3711
27X	3703
09X	3630
21Y	3435
32Y	3125
22X	3061
26X	2978
02Y	2970
10X	2967
14X	2733
03Y	2726
22Y	2664
12Y	2651
05X	2633
02X	2614
31X	2603
21X	2593
17Y	2582
24Y	2575
13Z	2546
15X	2509
24X	2459
12X	2422
10Y	2383

```
01X     2210
28X     2191
17X     2133
11X     2087
13Y     2017
35Y     1956
31Z     1870
18Y     1778
16Y     1561
14Y     1492
25Y     1482
13X     1122
18X     1063
16X      994
05Y      710
PDT2      20
Name: Beat, dtype: int64
```

[9]:
```
#(4)Incident Type Id
pd.value_counts(data['Incident Type Id'])
```

[9]:
```
933R      17348
911H      12817
SECCK     11393
415       10752
10851      7180
          …
243B          1
970A          1
148           1
YELALT        1
MTHLAB        1
Name: Incident Type Id, Length: 263, dtype: int64
```

[10]:
```
#(5)Incident Type Description
pd.value_counts(data['Incident Type Description'])
```

[10]:
```
ALARM-RINGER            17348
911 HANG-UP             12817
SECURITY CHECK          11393
STOLEN VEHICLE           7180
415 UNKNOWN              6624
                        …
CHOP SHOP OWNERSHIP/        1
ASSAULT ON A POLICE         1
THREATEN WITNESS/VIC        1
YELLOW ALERT AT THE         1
EXTORTION                   1
Name: Incident Type Description, Length: 265, dtype: int64
```

[11]:
```
#(6)Event Number
pd.value_counts(data['Event Number'])
```

[11]:
```
LOP110702000977    1
LOP111026000263    1
LOP110211000709    1
LOP111218000728    1
LOP110911000007    1
                  ..
LOP110729000748    1
LOP110416000414    1
LOP110730000423    1
LOP110622000796    1
LOP110127001173    1
Name: Event Number, Length: 180015, dtype: int64
```

[12]:
```
#(7)Closed Time
pd.value_counts(data['Closed Time'])
```

[12]:
```
2011-08-27T12:38:28.000    2
2011-07-05T17:52:50.000    2
2011-06-22T20:06:47.000    2
2011-08-02T01:52:05.000    2
2011-07-22T01:08:17.000    2
                          ..
```

```
2011-10-22T03:54:01.000    1
2011-08-21T12:05:02.000    1
2011-03-14T18:12:53.000    1
2011-07-17T09:44:56.000    1
2011-01-16T23:23:41.000    1
Name: Closed Time, Length: 179506, dtype: int64
```

(2) 数值属性，给出 5 数概括及缺失值的个数

[13]:
```
# 这里的数值属性包括 points 和 price
# 用 describe 函数对数据的 5 数进行概括
digital_data = ['Area Id','Priority']
data[digital_data].describe()
```

[13]:

	Area Id	Priority
count	179112.000000	180015.000000
mean	1.740648	1.796111
std	0.746468	0.402916
min	1.000000	0.000000
25%	1.000000	2.000000
50%	2.000000	2.000000
75%	2.000000	2.000000
max	3.000000	2.000000

Area Id: 最大值 3，最小值 1，均值 1.74，中位数 2，四分位数 [1,2,2], 缺失值个数为 904

Priority: 最大值 2，最小值 0，均值 1.80，中位数 2，四分位数 [2,2,2], 缺失值个数为 1

[14]:
```
# 给出 points 和 price 缺失值个数
print("The Null num of 'Area Id' is:",data['Area Id'].isnull().sum())
```

```
The Null num of 'Area Id' is: 904
```

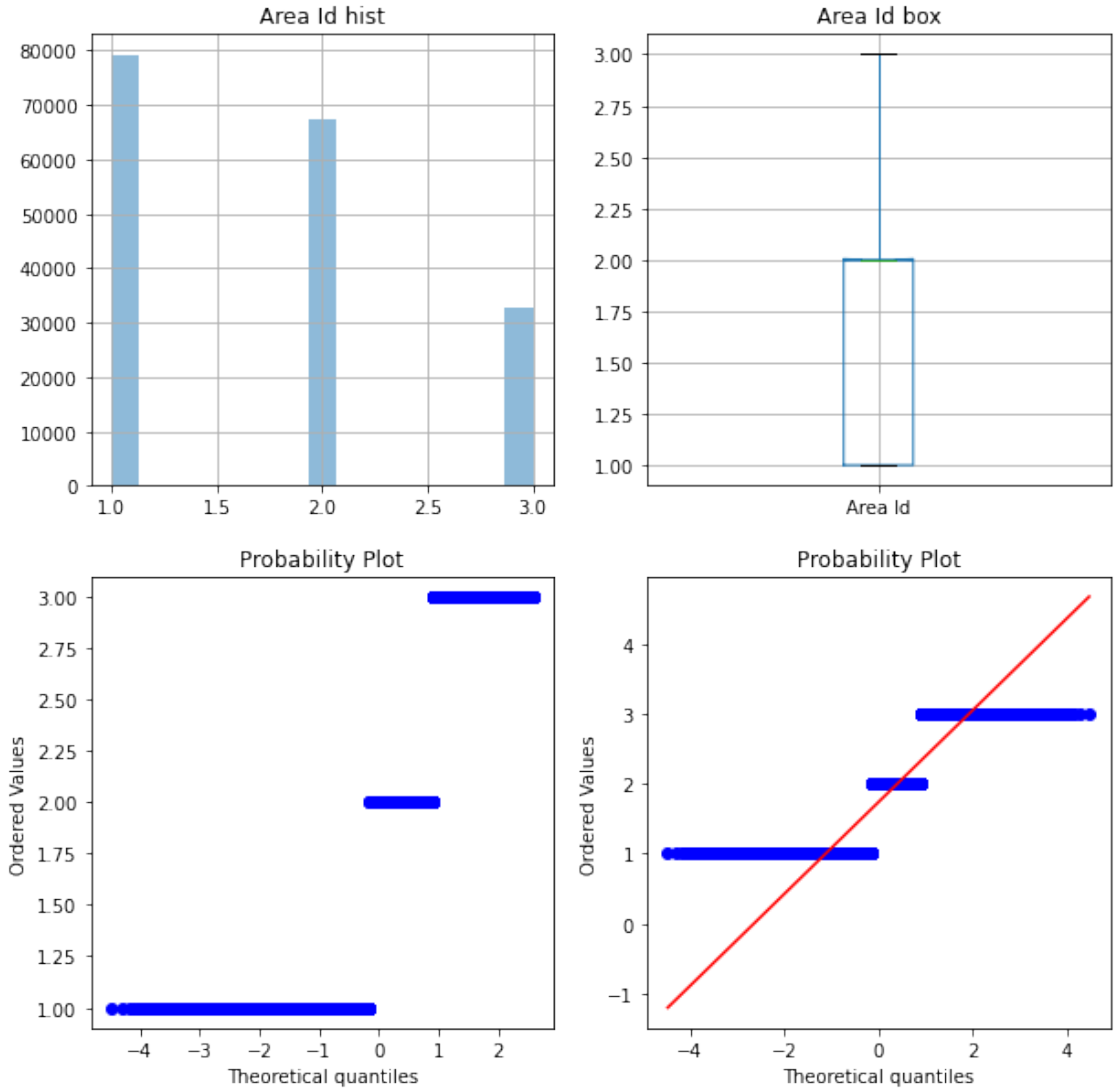
[15]:
```
print("The Null num of 'Priority' is:",data['Priority'].isnull().sum())
```

```
The Null num of 'Priority' is: 1
```

2.1.2 数据可视化

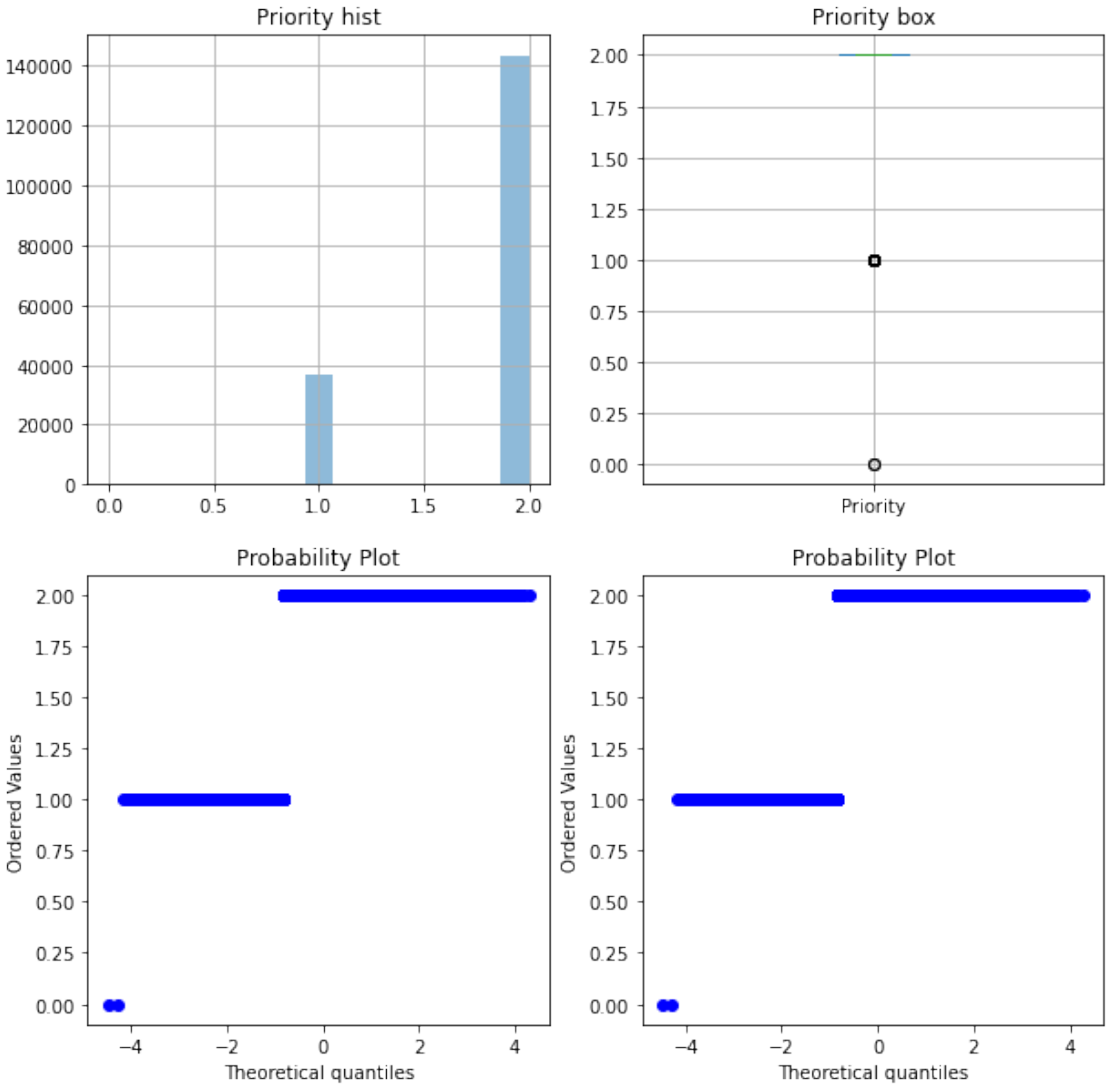
（1）绘制 Area Id 的直方图、盒图、qq 图（此处只针对数值类型的数据）

[17]:
```
# coding=utf-8
plt.figure(figsize = (10,10))

# 直方图
plt.subplot(2,2,1)
plt.title("Area Id hist")
data['Area Id'].hist(alpha=0.5,bins=15) #alpha 透明度, bins 竖条数

# 盒图
plt.subplot(2,2,2)
plt.title("Area Id box")
data['Area Id'].plot(kind='box',notch=True,grid=True)

#q-q 图
plt.subplot(2,2,3)
stats.probplot(data['Area Id'],dist="norm",plot=plt)

# 去除缺失值再绘制 q-q 图
plt.subplot(2,2,4)
data_drop=pd.DataFrame(data['Area Id'].copy(deep=True))
data_drop = data_drop.dropna()
stats.probplot(data_drop['Area Id'], dist="norm", plot=plt)

plt.show()
```



[19]:
```
# 绘制 price 的直方图、盒图、qq 图
plt.figure(figsize = (10,10))

# 直方图
plt.subplot(2,2,1)
plt.title("Priority hist")
data['Priority'].hist(alpha=0.5,bins=15) #alpha 透明度, bins 竖条数

# 盒图
plt.subplot(2,2,2)
```

```
plt.title("Priority box")
data['Priority'].plot(kind='box',notch=True,grid=True)

#q-q 图
plt.subplot(2,2,3)
stats.probplot(data['Priority'],dist="norm",plot=plt)

# 去除缺失值再绘制 q-q 图
plt.subplot(2,2,4)
pricewithooutnulldata = pd.DataFrame(data['Priority'])
pricewithooutnulldata = pricewithooutnulldata.dropna()
stats.probplot(data['Priority'],dist="norm",plot=plt)

plt.show()
```



2.1.3 由上图可以得出结论：

Area Id 属性、Priority 属性分布均不符合正态分布。

3 数据缺失处理

[20]:
```
# 绘制表格查看数据缺失值并检验四种方案填充后是否还有缺失值
def missing_data(datatodel):
    missing_num = datatodel.isnull().sum()
    missing_percent = missing_num/datatodel.shape[0]*100
```

```
    concat_data = pd.
↪concat([missing_num,missing_percent],axis=1,keys=['missing_num','missing_percent'])
    concat_data['Types'] = datatodel.dtypes
    return concat_data
```

由上表可以看出，数值型数据 price 存在缺失值

标称型数据 country，designation,province,region_1,region_2 存在缺失值

这里缺失的原因可能是由于未完全记录、遗漏或无法获取

3.1 方案一缺失值剔除

[23]:
```
# 未处理前的原始数据
missing_data(data)
```

[23]:

	missing_num	missing_percent	Types
Create Time	1	0.000556	object
Location	0	0.000000	object
Area Id	904	0.502178	float64
Beat	520	0.288863	object
Priority	1	0.000556	float64
Incident Type Id	1	0.000556	object
Incident Type Description	1	0.000556	object
Event Number	1	0.000556	object
Closed Time	7	0.003889	object

[21]:
```
del_null_data = data.copy(deep=True)
del_null_data = del_null_data.dropna()
```

[22]:
```
# 处理缺失数据后的数据展示
missing_data(del_null_data)
```

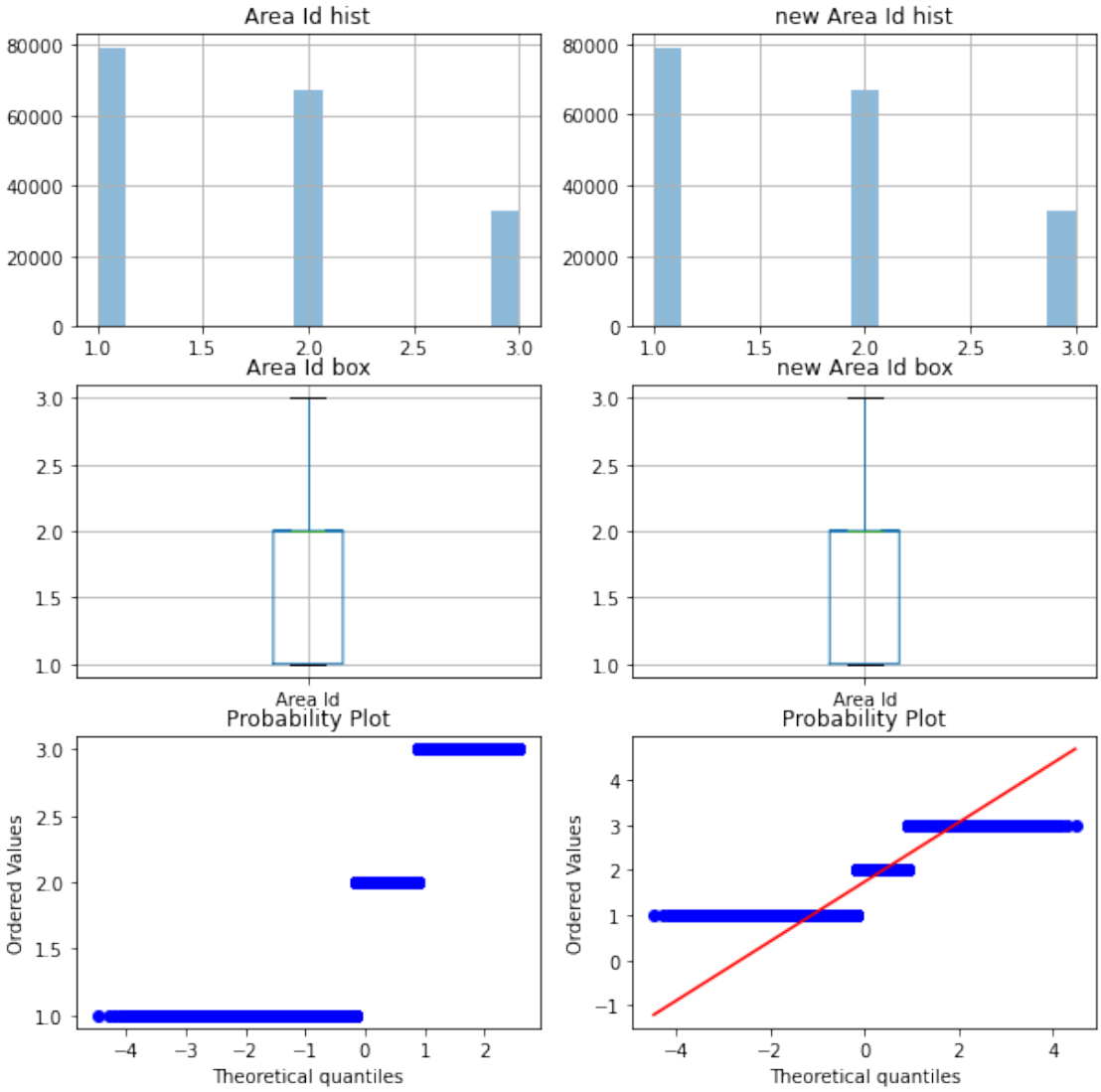
[22]:

	missing_num	missing_percent	Types
Create Time	0	0.0	object
Location	0	0.0	object
Area Id	0	0.0	float64
Beat	0	0.0	object
Priority	0	0.0	float64

```
Incident Type Id                    0              0.0   object
Incident Type Description           0              0.0   object
Event Number                        0              0.0   object
Closed Time                         0              0.0   object
```

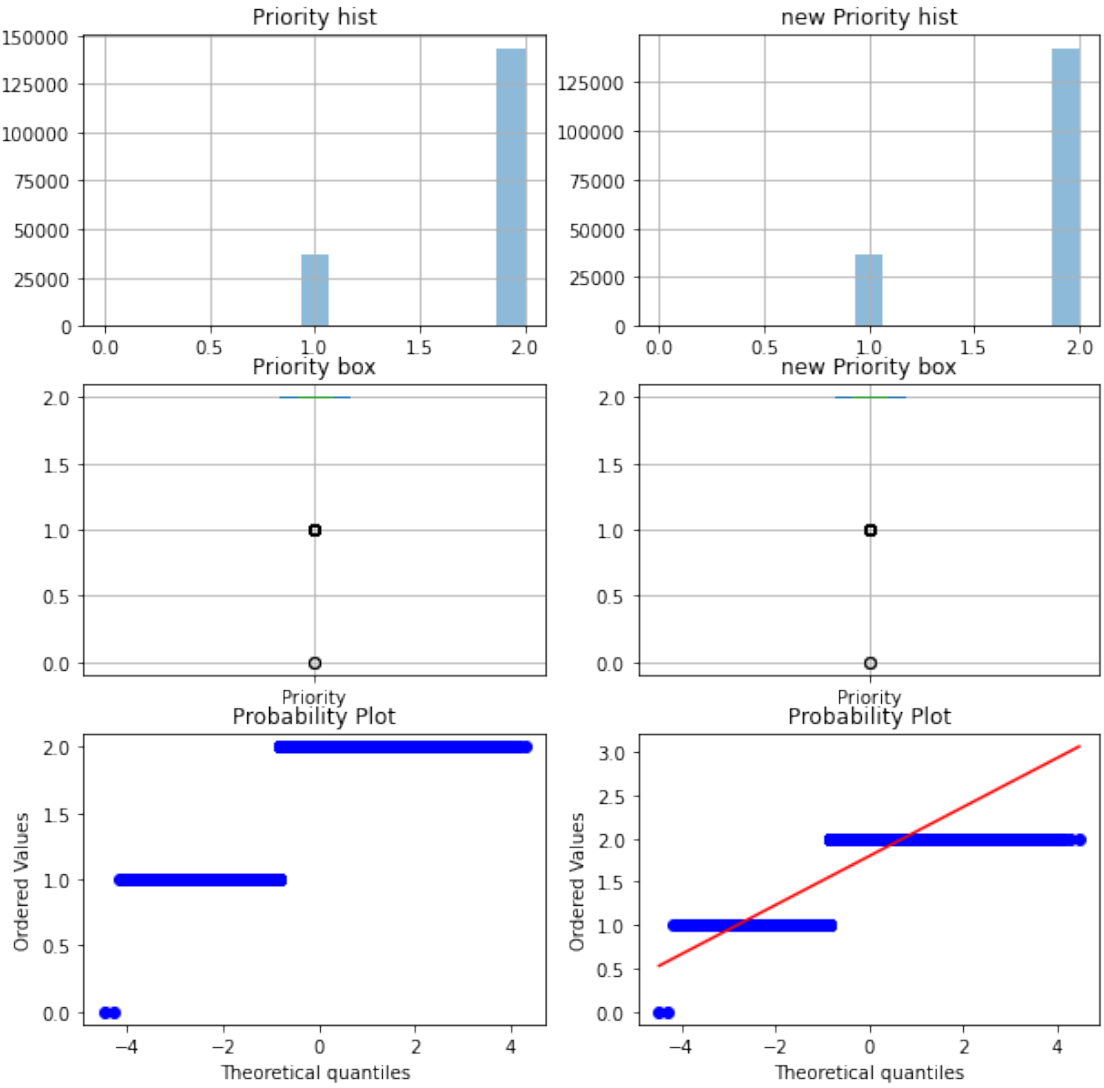
[25]:
```
# Area Id 可视化对比新旧数据

plt.figure(figsize = (10,10))

# 直方图
plt.subplot(3,2,1)
plt.title("Area Id hist")
data['Area Id'].hist(alpha=0.5,bins=15) #alpha 透明度, bins 竖条数

# 直方图
plt.subplot(3,2,2)
plt.title("new Area Id hist")
del_null_data['Area Id'].hist(alpha=0.5,bins=15) #alpha 透明度, bins 竖条数



# 盒图
plt.subplot(3,2,3)
plt.title("Area Id box")
data['Area Id'].plot(kind='box',notch=True,grid=True)

# 盒图
plt.subplot(3,2,4)
plt.title("new Area Id box")
del_null_data['Area Id'].plot(kind='box',notch=True,grid=True)

#q-q 图
plt.subplot(3,2,5)
stats.probplot(data['Area Id'],dist="norm",plot=plt)

plt.subplot(3,2,6)
stats.probplot(del_null_data['Area Id'],dist="norm",plot=plt)
```

```
plt.show()
```



[27]:
```
# Priority 可视化对比新旧数据

plt.figure(figsize = (10,10))

# 直方图
plt.subplot(3,2,1)
plt.title("Priority hist")
```

```
data['Priority'].hist(alpha=0.5,bins=15) #alpha 透明度, bins 竖条数

# 直方图
plt.subplot(3,2,2)
plt.title("new Priority hist")
del_null_data['Priority'].hist(alpha=0.5,bins=15) #alpha 透明度, bins 竖条数


# 盒图
plt.subplot(3,2,3)
plt.title("Priority box")
data['Priority'].plot(kind='box',notch=True,grid=True)

# 盒图
plt.subplot(3,2,4)
plt.title("new Priority box")
del_null_data['Priority'].plot(kind='box',notch=True,grid=True)

#q-q 图
plt.subplot(3,2,5)
stats.probplot(data['Priority'],dist="norm",plot=plt)

plt.subplot(3,2,6)
stats.probplot(del_null_data['Priority'],dist="norm",plot=plt)

plt.show()
```



[28]:
```
del_null_data[['Area Id','Priority']].describe()  # 缺失部分剔除后数据的 5 数概况
```

[28]:

	Area Id	Priority
count	178771.000000	178771.000000
mean	1.740898	1.795252
std	0.746487	0.403546
min	1.000000	0.000000
25%	1.000000	2.000000
50%	2.000000	2.000000
75%	2.000000	2.000000

```
max          3.000000        2.000000
```

缺失部分剔除后

Area Id: 最大值 3，最小值 1，均值 1.74，中位数 2，四分位数 [1,2,2], 缺失值个数为 0

Priority: 最大值 2，最小值 0，均值 1.80，中位数 2，四分位数 [2,2,2], 缺失值个数为 0

3.2 用最高频率值来填补缺失值

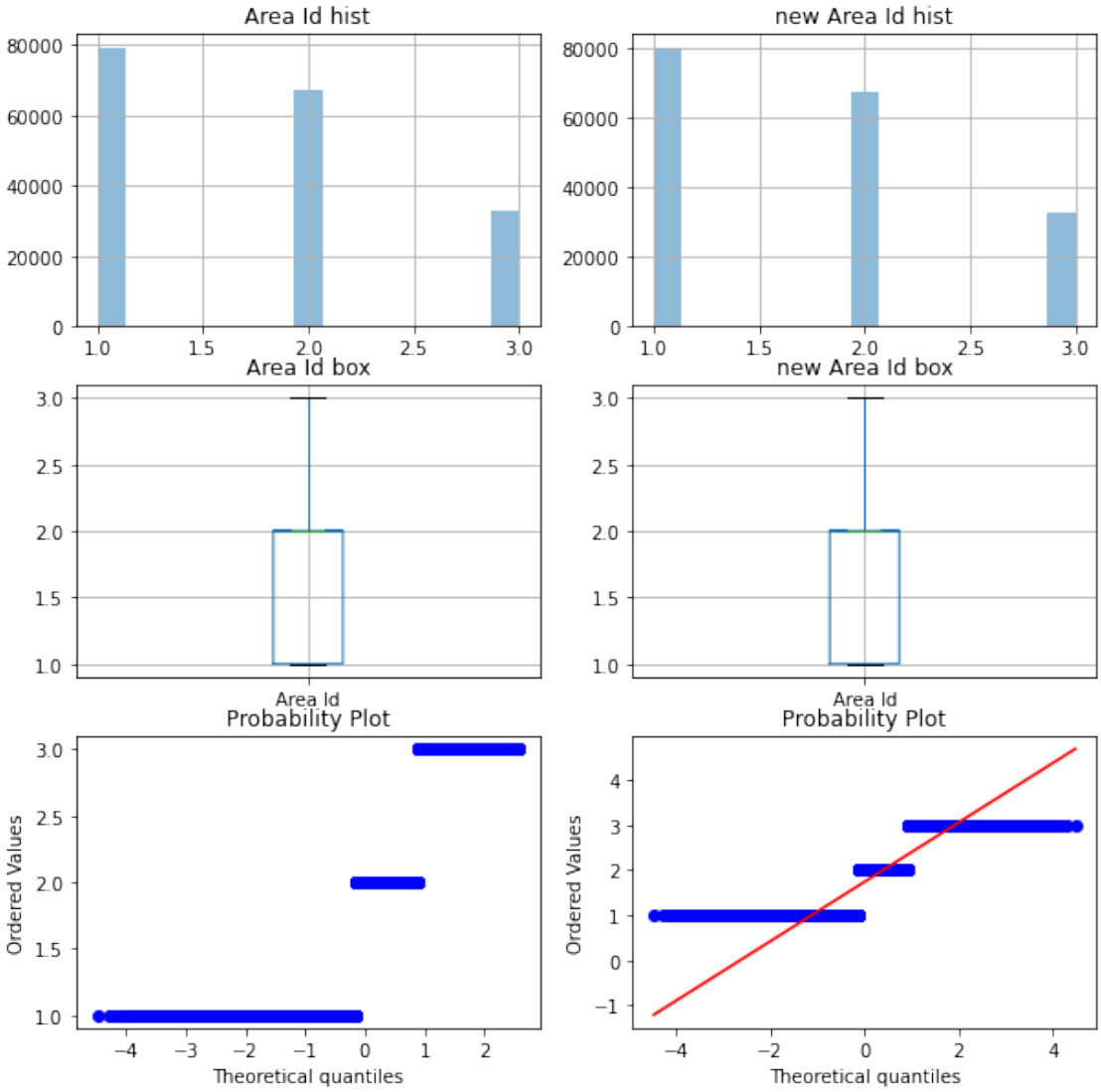
[29]:
```
# 用最高频率来填补缺失值--此处使用深拷贝，否则会改变原值
fill_data_with_most_frequency = data.copy(deep=True)
# 对'Area Id'进行最高频率值填补缺失值
word_counts = Counter(fill_data_with_most_frequency['Area Id'])
top = word_counts.most_common(1)[0][0]
fill_data_with_most_frequency['Area Id'] = fill_data_with_most_frequency['Area␣
↪Id'].fillna(top)



# 对'Area Id'进行最高频率值填补缺失值
word_counts = Counter(fill_data_with_most_frequency['Priority'])
top = word_counts.most_common(1)[0][0]
fill_data_with_most_frequency['Priority'] =␣
↪fill_data_with_most_frequency['Priority'].fillna(top)
```

[30]:
```
# 查看填充后是否还有数据缺失
missing_data(fill_data_with_most_frequency)
```

[30]:

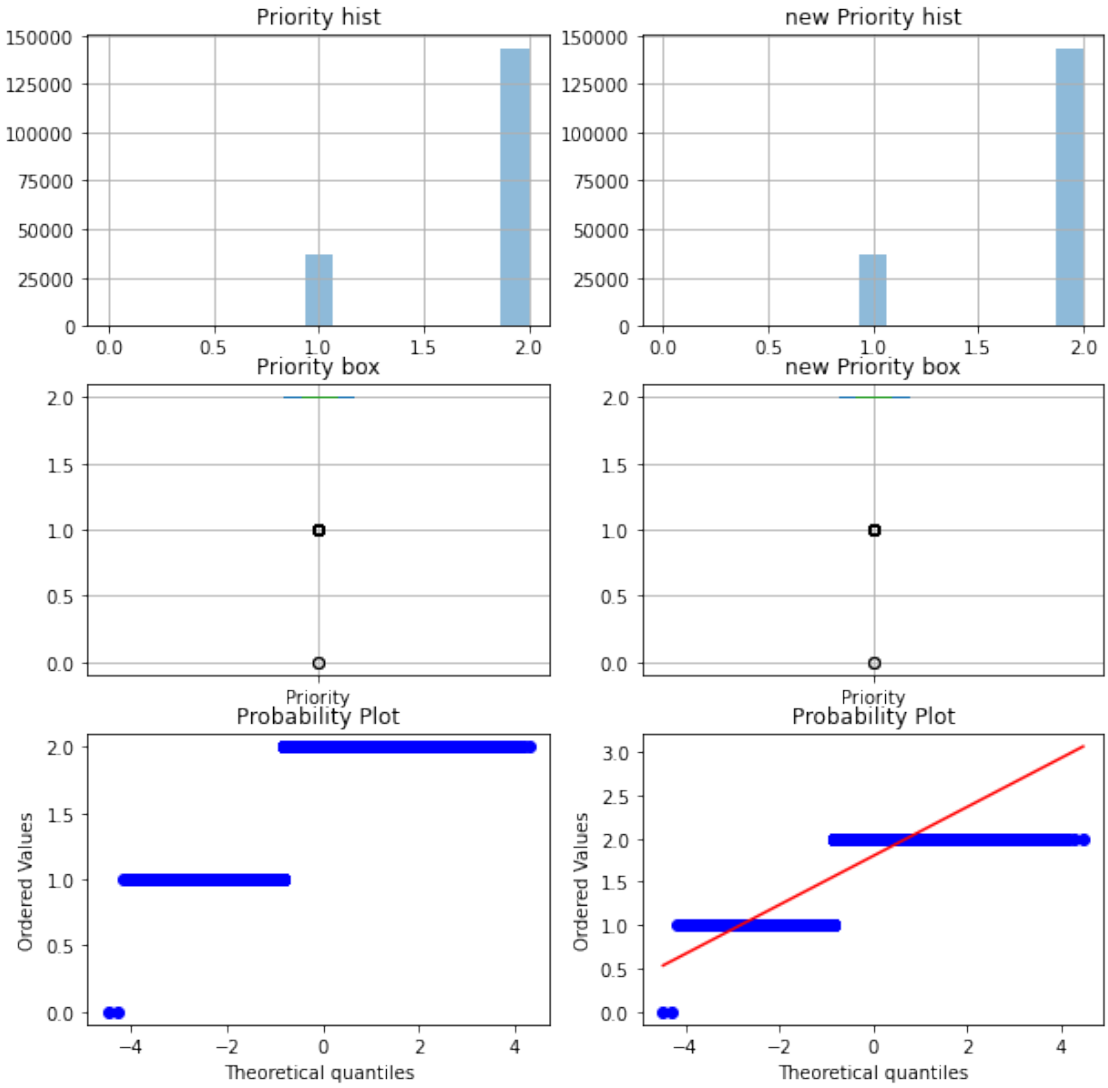
	missing_num	missing_percent	Types
Create Time	1	0.000556	object
Location	0	0.000000	object
Area Id	0	0.000000	float64
Beat	520	0.288863	object
Priority	0	0.000000	float64
Incident Type Id	1	0.000556	object
Incident Type Description	1	0.000556	object
Event Number	1	0.000556	object
Closed Time	7	0.003889	object

[31]:
```
#Area Id 可视化对比新旧数据

plt.figure(figsize = (10,10))

# 直方图
plt.subplot(3,2,1)
plt.title("Area Id hist")
data['Area Id'].hist(alpha=0.5,bins=15) #alpha 透明度, bins 竖条数

# 直方图
plt.subplot(3,2,2)
plt.title("new Area Id hist")
fill_data_with_most_frequency['Area Id'].hist(alpha=0.5,bins=15) #alpha 透明度,
bins 竖条数



# 盒图
plt.subplot(3,2,3)
plt.title("Area Id box")
data['Area Id'].plot(kind='box',notch=True,grid=True)

# 盒图
plt.subplot(3,2,4)
plt.title("new Area Id box")
fill_data_with_most_frequency['Area Id'].plot(kind='box',notch=True,grid=True)

#q-q 图
plt.subplot(3,2,5)
stats.probplot(data['Area Id'],dist="norm",plot=plt)

plt.subplot(3,2,6)
stats.probplot(fill_data_with_most_frequency['Area Id'],dist="norm",plot=plt)

plt.show()
```



[32]:
```
#Priority 可视化对比新旧数据

plt.figure(figsize = (10,10))

# 直方图
plt.subplot(3,2,1)
plt.title("Priority hist")
data['Priority'].hist(alpha=0.5,bins=15) #alpha 透明度, bins 竖条数
```

```
# 直方图
plt.subplot(3,2,2)
plt.title("new Priority hist")
fill_data_with_most_frequency['Priority'].hist(alpha=0.5,bins=15) #alpha 透明度,
bins 竖条数



# 盒图
plt.subplot(3,2,3)
plt.title("Priority box")
data['Priority'].plot(kind='box',notch=True,grid=True)

# 盒图
plt.subplot(3,2,4)
plt.title("new Priority box")
fill_data_with_most_frequency['Priority'].plot(kind='box',notch=True,grid=True)

#q-q 图
plt.subplot(3,2,5)
stats.probplot(data['Priority'],dist="norm",plot=plt)

plt.subplot(3,2,6)
stats.probplot(fill_data_with_most_frequency['Priority'],dist="norm",plot=plt)

plt.show()
```



[34]:
```
# 对填充后的新数据进行描述
fill_data_with_most_frequency[['Area Id','Priority']].describe()
```

[34]:

	Area Id	Priority
count	180016.000000	180016.000000
mean	1.736929	1.796113
std	0.746430	0.402915
min	1.000000	0.000000
25%	1.000000	2.000000
50%	2.000000	2.000000

```
75%         2.000000        2.000000
max         3.000000        2.000000
```

3.3 通过属性的相关关系来填补缺失值

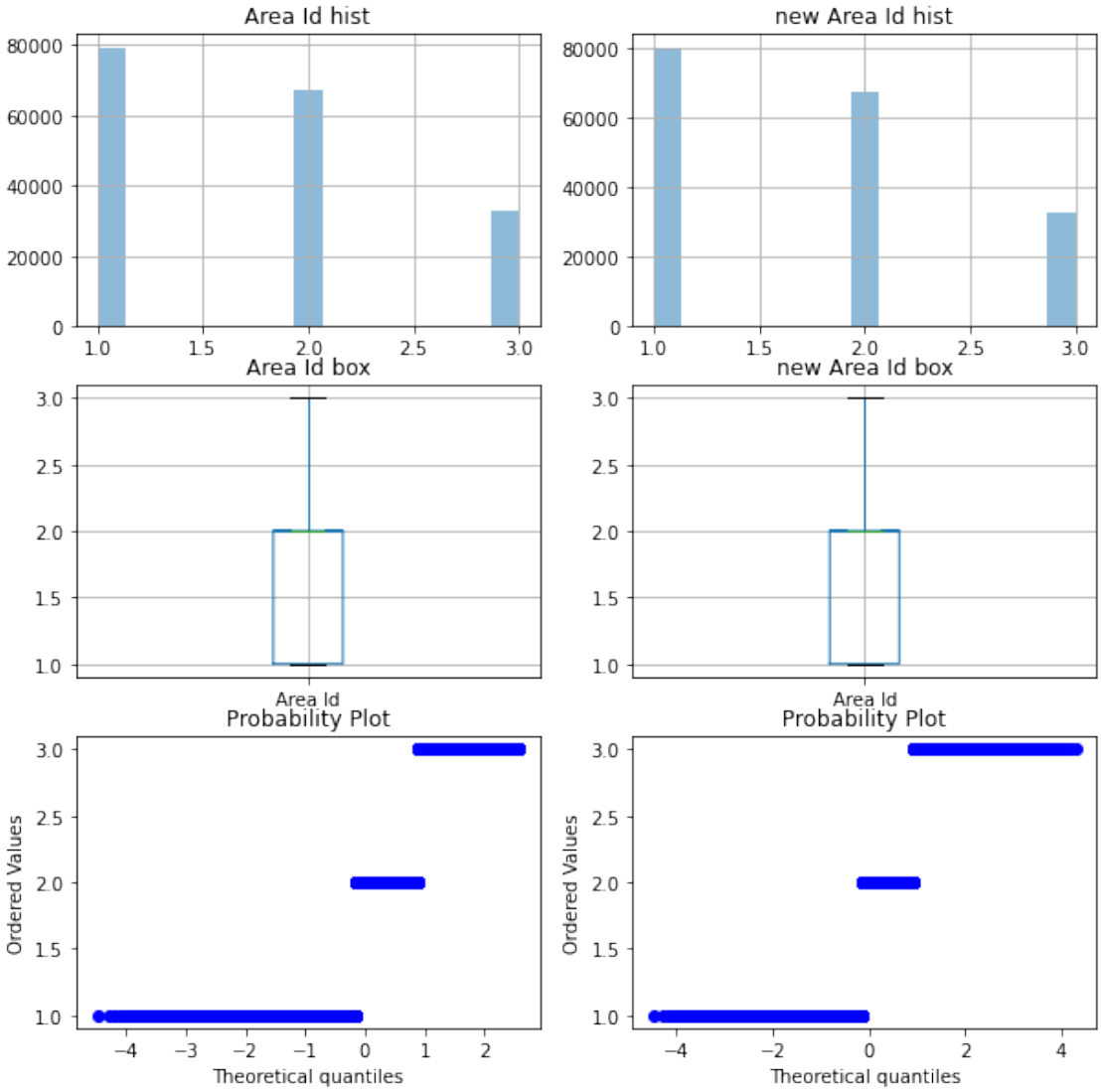
[35]:
```
# 查看相关的属性关系
data.corr()
```

[35]:
```
            Area Id  Priority
Area Id    1.000000 -0.023366
Priority -0.023366  1.000000
```

[36]:
```
# 通过属性的相关关系来填补缺失值
target_data = data['Area Id'].copy(deep=True)
source_data = data['Priority'].copy(deep=True)

flag1 = target_data.isnull().values
flag2 = source_data.isnull().values

i=0
for _,value in target_data.iteritems():
    if(flag1[i]==True) and (flag2[i]==False):
        target_data[i] = 3 - source_data[i]
    i=i+1
```

[37]:
```
#Area Id 可视化对比新旧数据

plt.figure(figsize = (10,10))

# 直方图
plt.subplot(3,2,1)
plt.title("Area Id hist")
data['Area Id'].hist(alpha=0.5,bins=15) #alpha 透明度, bins 竖条数

# 直方图
plt.subplot(3,2,2)
plt.title("new Area Id hist")
```

```
target_data.hist(alpha=0.5,bins=15) #alpha 透明度, bins 竖条数


# 盒图
plt.subplot(3,2,3)
plt.title("Area Id box")
data['Area Id'].plot(kind='box',notch=True,grid=True)

# 盒图
plt.subplot(3,2,4)
plt.title("new Area Id box")
target_data.plot(kind='box',notch=True,grid=True)

#q-q 图
plt.subplot(3,2,5)
stats.probplot(data['Area Id'],dist="norm",plot=plt)

plt.subplot(3,2,6)
stats.probplot(target_data,dist="norm",plot=plt)

plt.show()
```



[41]:
```
# 补充后的 Area Id 数据描述
target_data.describe()
```

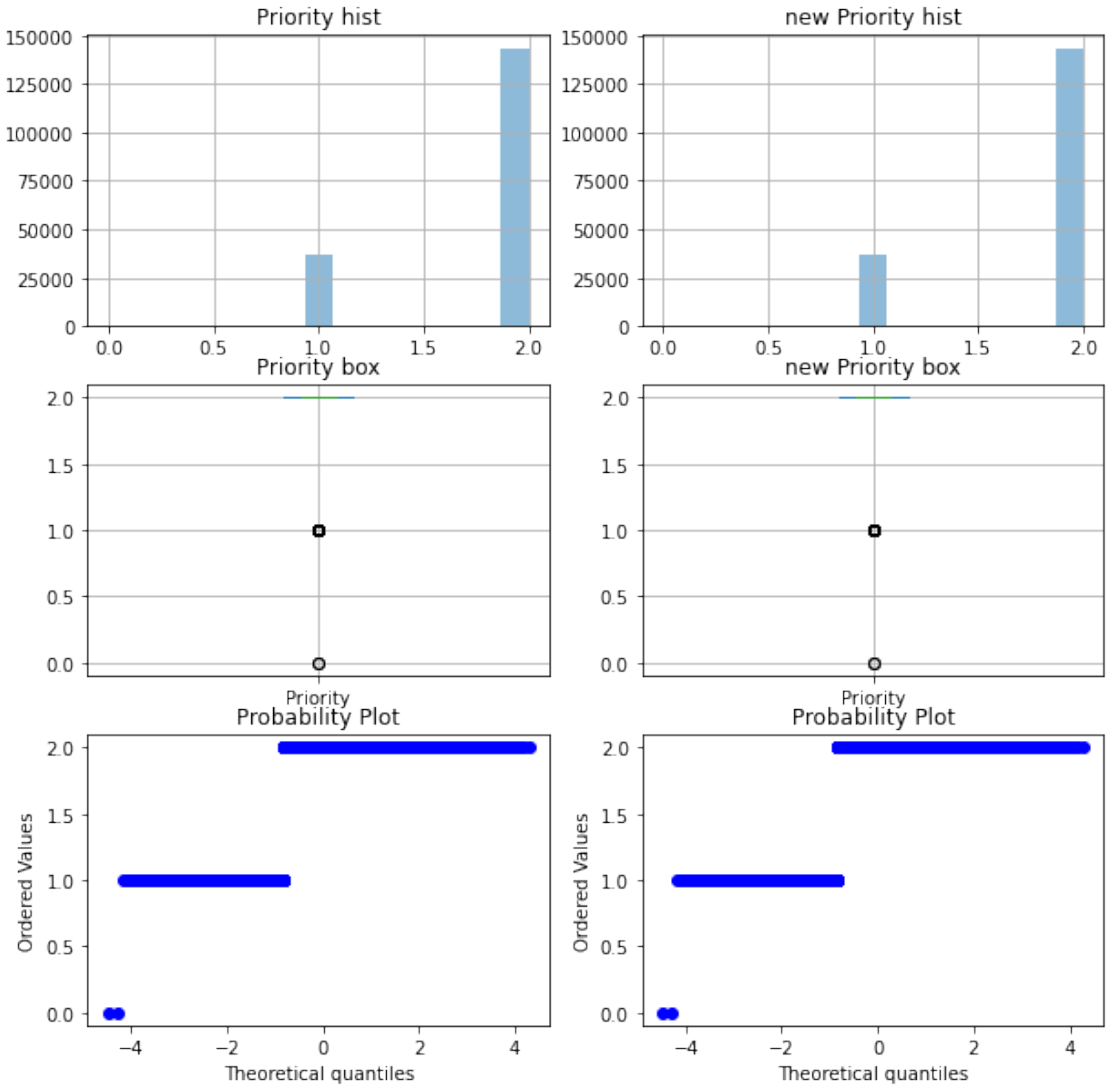
[41]:
```
count    180015.000000
mean          1.796111
std           0.402916
min           0.000000
25%           2.000000
50%           2.000000
75%           2.000000
```

```
max           2.000000
Name: Priority, dtype: float64
```

[38]:
```
# 通过属性的相关关系来填补 Priority 缺失值
target_data = data['Priority'].copy(deep=True)
source_data = data['Area Id'].copy(deep=True)

flag1 = target_data.isnull().values
flag2 = source_data.isnull().values

i=0
for _,value in target_data.iteritems():
    if(flag1[i]==True) and (flag2[i]==False):
        target_data[i] = 3 - source_data[i]
    i=i+1
```

[39]:
```
#Priority 可视化对比新旧数据

plt.figure(figsize = (10,10))

# 直方图
plt.subplot(3,2,1)
plt.title("Priority hist")
data['Priority'].hist(alpha=0.5,bins=15) #alpha 透明度, bins 竖条数

# 直方图
plt.subplot(3,2,2)
plt.title("new Priority hist")
target_data.hist(alpha=0.5,bins=15) #alpha 透明度, bins 竖条数


# 盒图
plt.subplot(3,2,3)
plt.title("Priority box")
data['Priority'].plot(kind='box',notch=True,grid=True)

# 盒图
```

```
plt.subplot(3,2,4)
plt.title("new Priority box")
target_data.plot(kind='box',notch=True,grid=True)

#q-q 图
plt.subplot(3,2,5)
stats.probplot(data['Priority'],dist="norm",plot=plt)

plt.subplot(3,2,6)
stats.probplot(target_data,dist="norm",plot=plt)

plt.show()
```



[40]:
```
# 补充后的 Priority 数据描述
target_data.describe()
```

[40]:
```
count    180015.000000
mean          1.796111
std           0.402916
min           0.000000
25%           2.000000
50%           2.000000
75%           2.000000
max           2.000000
Name: Priority, dtype: float64
```

3.4 通过对象的相似性填补缺失值

[43]:
```
numeric_attr = ['Area Id','Priority']
# 查找两个对象间的相似性
# 如果通过暴力法求解耗时耗力
# 所以选择通过二分法查找的方法进行相似性选择

def find_dis_value(dataset, pos, numeric_attr):
    def dis_objs(tar_obj_index, sou_obj_index):
        tar_obj = dataset.iloc[tar_obj_index]
        sou_obj = dataset.iloc[sou_obj_index]
        dis_value = 0
        for column in tar_obj.index:
            if column == 'Priority':
                if (not math.isnan(tar_obj[column])) and (not math.
 ↪isnan(sou_obj[column])):
                    dis_value += sou_obj[column] - tar_obj[column]
                else:
                    dis_value += 9998
        return dis_value
```

```
    mindis = 9999
    result_pos = -1
    leftindex = 0;
    rightindex = dataset.shape[0]-1
    # 二分查找返回最近距离的一个 result_pos
    while leftindex<=rightindex:
        midindex = int((leftindex+rightindex)/2)
        tmpdis = dis_objs(pos,midindex)
        if(tmpdis>0):
            rightindex = midindex-1
        elif(tmpdis == 0):
            result_pos = midindex
            break;
        else:
            leftindex = midindex+1
        if(tmpdis<mindis):
            result_pos = midindex
    return result_pos




# 通过数据对象之间的相似性来填补缺失值
numical_datasets = pd.DataFrame(data[numeric_attr].copy(deep=True))

# 对 numical_datasets 排序
numical_datasets.sort_values("Priority",inplace=True)
data_area_id = numical_datasets['Area Id'].copy(deep=True)

print('空数据数量为:',data_area_id.isnull().sum())
length = numical_datasets.shape[0]
count=1;
for i in range(length):
    if math.isnan(numical_datasets['Area Id'].iloc[i]):
#         print(' 当前处理第'+str(count)+" 个")
#         print(i,numical_datasets.iloc[i])
```

```
        result_pos = find_dis_value(numical_datasets, i, numeric_attr)
#         print(result_pos,numical_datasets.iloc[result_pos])
        data_area_id.iloc[i] = data_area_id.iloc[result_pos]
#         print(i,data_area_id.iloc[i])
        count+=1
```

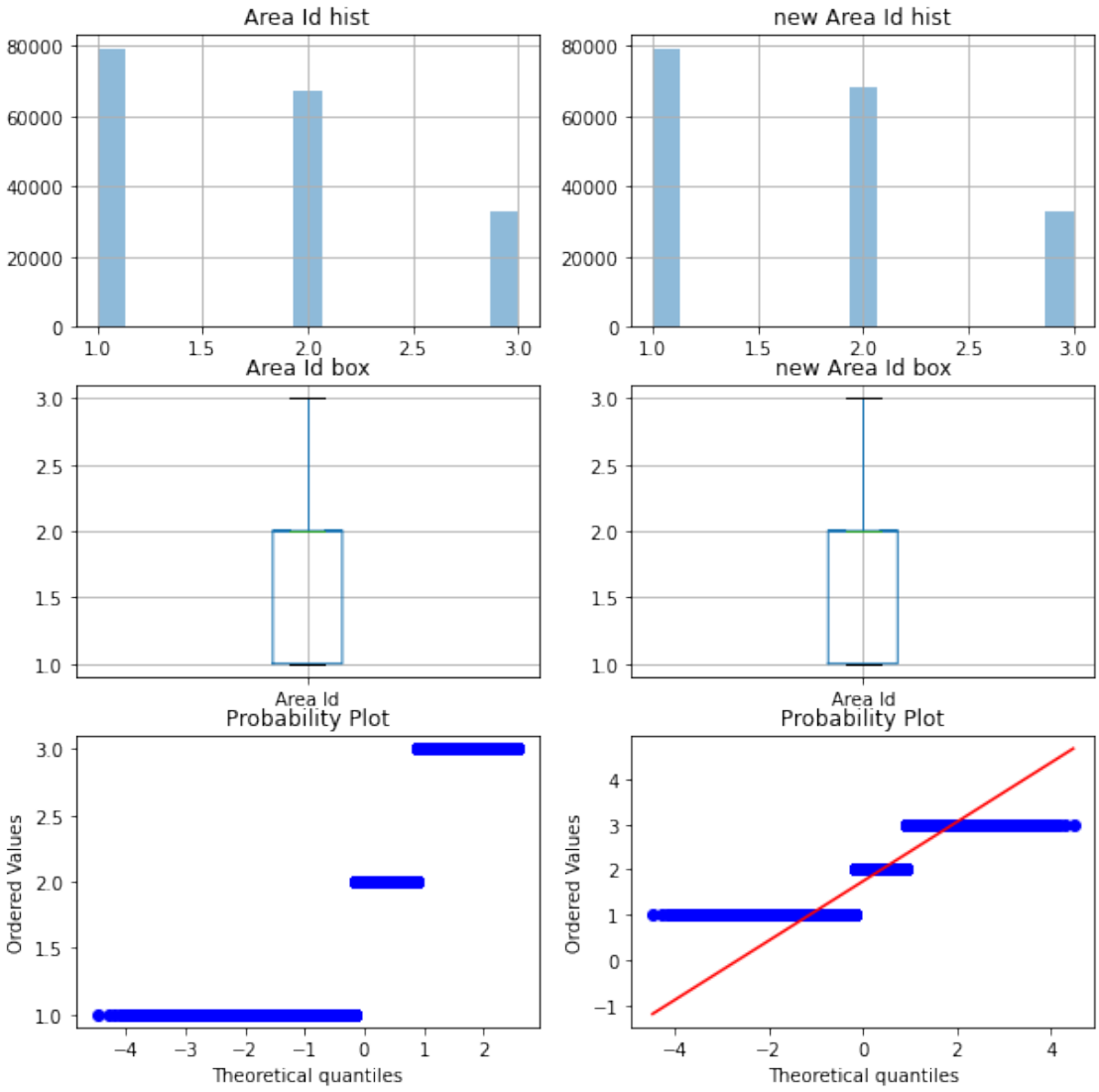
空数据数量为：904

[44]:
```
# 填充后的空数据数量
print(data_area_id.isnull().sum())
```

0

[46]:
```
#Area Id 可视化对比新旧数据

plt.figure(figsize = (10,10))

# 直方图
plt.subplot(3,2,1)
plt.title("Area Id hist")
data['Area Id'].hist(alpha=0.5,bins=15) #alpha 透明度, bins 竖条数

# 直方图
plt.subplot(3,2,2)
plt.title("new Area Id hist")
data_area_id.hist(alpha=0.5,bins=15) #alpha 透明度, bins 竖条数


# 盒图
plt.subplot(3,2,3)
plt.title("Area Id box")
data['Area Id'].plot(kind='box',notch=True,grid=True)

# 盒图
plt.subplot(3,2,4)
plt.title("new Area Id box")
data_area_id.plot(kind='box',notch=True,grid=True)
```

```
#q-q 图
plt.subplot(3,2,5)
stats.probplot(data['Area Id'],dist="norm",plot=plt)

plt.subplot(3,2,6)
stats.probplot(data_area_id,dist="norm",plot=plt)

plt.show()
```



[47]:
```
#area id 的数据描述
data_area_id.describe()  # # 通过数据对象之间的相似性来填补后数据的 5 数概况
```

[47]:
```
count    180016.000000
mean          1.741956
std           0.744823
min           1.000000
25%           1.000000
50%           2.000000
75%           2.000000
max           3.000000
Name: Area Id, dtype: float64
```

通过相似性填补后

Area Id: 最大值 3，最小值 1，均值 1.74，中位数 2，四分位数 [1,2,2], 缺失值个数为 0

[48]:
```
# 使用 id 对 priority 进行填补
numeric_attr = ['Area Id','Priority']
# 查找两个对象间的相似性
# 如果通过暴力法求解耗时耗力
# 所以选择通过二分法查找的方法进行相似性选择

def find_dis_value(dataset, pos, numeric_attr):
    def dis_objs(tar_obj_index, sou_obj_index):
        tar_obj = dataset.iloc[tar_obj_index]
        sou_obj = dataset.iloc[sou_obj_index]
        dis_value = 0
        for column in tar_obj.index:
            if column == 'Area Id':
                if (not math.isnan(tar_obj[column])) and (not math.
 ↪isnan(sou_obj[column])):
                    dis_value += sou_obj[column] - tar_obj[column]
                else:
                    dis_value += 9998
        return dis_value
```

```
    mindis = 9999
    result_pos = -1
    leftindex = 0;
    rightindex = dataset.shape[0]-1
    # 二分查找返回最近距离的一个 result_pos
    while leftindex<=rightindex:
        midindex = int((leftindex+rightindex)/2)
        tmpdis = dis_objs(pos,midindex)
        if(tmpdis>0):
            rightindex = midindex-1
        elif(tmpdis == 0):
            result_pos = midindex
            break;
        else:
            leftindex = midindex+1
        if(tmpdis<mindis):
            result_pos = midindex
    return result_pos




# 通过数据对象之间的相似性来填补缺失值
numical_datasets = pd.DataFrame(data[numeric_attr].copy(deep=True))

# 对 numical_datasets 排序
numical_datasets.sort_values("Area Id",inplace=True)
data_Priority = numical_datasets['Priority'].copy(deep=True)

print('空数据数量为:',data_Priority.isnull().sum())
length = numical_datasets.shape[0]
count=1;
for i in range(length):
    if math.isnan(numical_datasets['Priority'].iloc[i]):
#         print(' 当前处理第'+str(count)+" 个")
#         print(i,numical_datasets.iloc[i])
```

```
        result_pos = find_dis_value(numical_datasets, i, numeric_attr)
#         print(result_pos,numical_datasets.iloc[result_pos])
        data_Priority.iloc[i] = data_Priority.iloc[result_pos]
#         print(i,data_area_id.iloc[i])
        count+=1
```

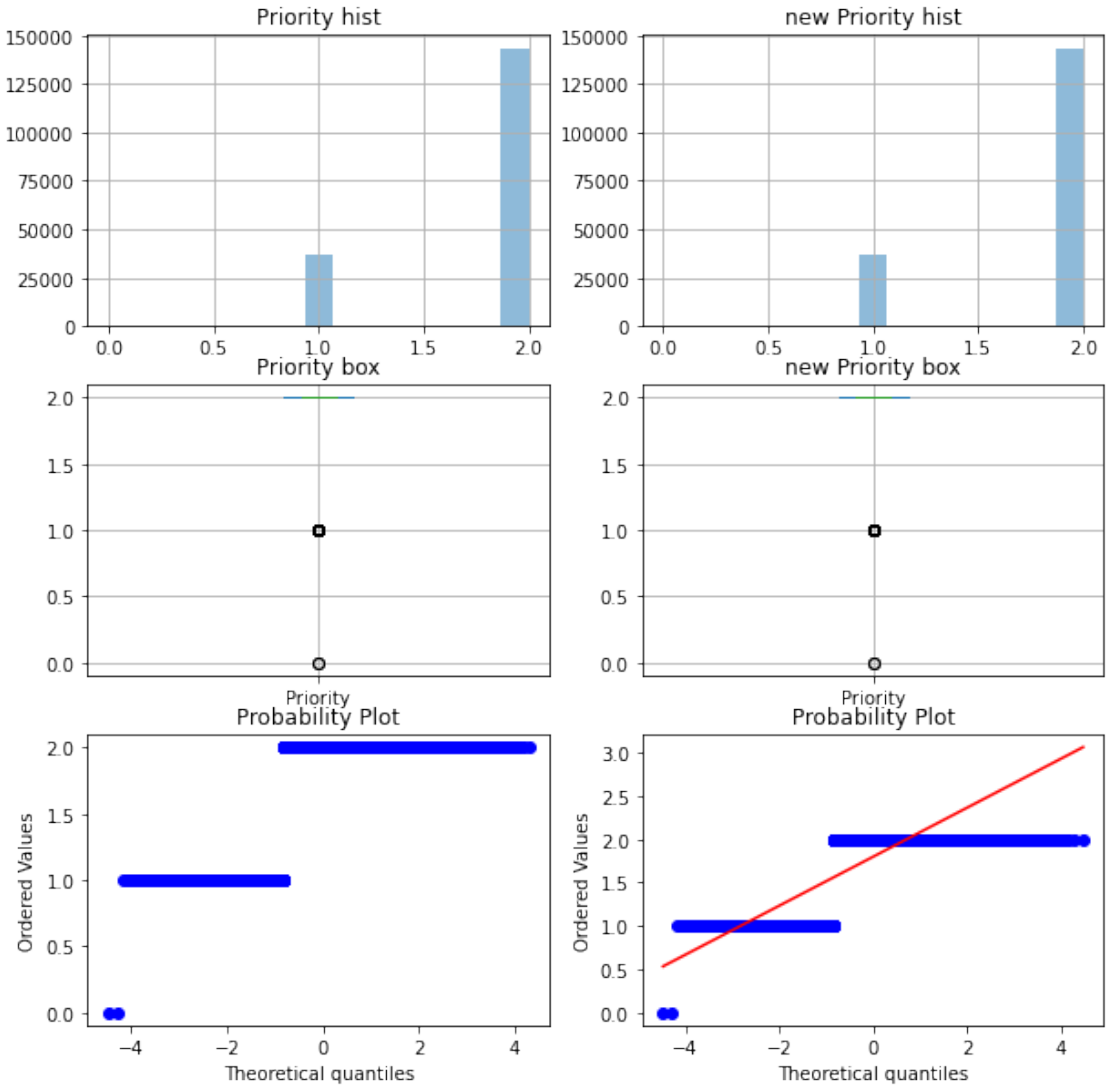
空数据数量为: 1

[49]:
```
# 补充后的 Priority 的空数据数量
print(data_Priority.isnull().sum())
```

0

[50]:
```
#Priority 可视化对比新旧数据

plt.figure(figsize = (10,10))

# 直方图
plt.subplot(3,2,1)
plt.title("Priority hist")
data['Priority'].hist(alpha=0.5,bins=15) #alpha 透明度, bins 竖条数

# 直方图
plt.subplot(3,2,2)
plt.title("new Priority hist")
data_Priority.hist(alpha=0.5,bins=15) #alpha 透明度, bins 竖条数


# 盒图
plt.subplot(3,2,3)
plt.title("Priority box")
data['Priority'].plot(kind='box',notch=True,grid=True)

# 盒图
plt.subplot(3,2,4)
plt.title("new Priority box")
data_Priority.plot(kind='box',notch=True,grid=True)
```

```
#q-q 图
plt.subplot(3,2,5)
stats.probplot(data['Priority'],dist="norm",plot=plt)

plt.subplot(3,2,6)
stats.probplot(data_Priority,dist="norm",plot=plt)

plt.show()
```



[51]:
```
#Priority 的数据描述
data_Priority.describe()
```

[51]:
```
count    180016.000000
mean          1.796107
std           0.402919
min           0.000000
25%           2.000000
50%           2.000000
75%           2.000000
max           2.000000
Name: Priority, dtype: float64
```