## **数据挖掘大作业一：数据探索性分析与数据预处理**

姓名：相铮 学号：2620170056

**1. 问题描述**

本次作业中，将对2个数据集进行探索性分析与预处理。

**2. 数据说明**

* 数据集1: NFL Play-by-Play 2009-2017
* 数据集2: San Francisco Building Permits

下载数据: 地址

数据集中属性解释：

* 数据集1: 参考
* 数据集2: 见下载地址中DataDictionaryBuildingPermit.xlsx

**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，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**

观察数据集中缺失数据，分析其缺失的原因。

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

**处理过程：**

# \*\*Step1. 读取数据\*\*

# - 读取csv文件，生成data frame

#DataFile = open("E:/Building\_Permits.csv",encoding='gb18030',errors='ignore')

DataFile = open("E:/NFL Play by Play 2009-2017 (v4).csv",encoding='gb18030',errors='ignore')

DataTable = pandas.read\_csv(DataFile,low\_memory=False);

DataTable = DataTable.dropna(axis=1, how='all');

# 定义两类数据：标称型和数值型

#数据集1

name\_category = ['GameID', 'time', 'SideofField', 'FirstDown', 'posteam', 'DefensiveTeam', 'desc',

'PlayAttempted', 'Yards.Gained', 'sp', 'Touchdown', 'ExPointResult', 'TwoPointConv', 'DefTwoPoint',

'Onsidekick', 'Safety', 'PuntResult', 'PlayType', 'Passer', 'Passer\_ID', 'PassAttempt',

'PassOutcome', 'PassLength', 'QBHit', 'PassLocation', 'InterceptionThrown', 'Interceptor', 'Rusher',

'Rusher\_ID', 'RushAttempt', 'RunLocation', 'RunGap', 'Receiver', 'Receiver\_ID', 'Reception',

'ReturnResult', 'Returner', 'BlockingPlayer', 'Tackler1', 'Tackler2', 'FieldGoalResult', 'Fumble',

'RecFumbTeam', 'RecFumbPlayer', 'Sack', 'Challenge.Replay', 'ChalReplayResult', 'Accepted.Penalty',

'PenalizedTeam', 'PenaltyType', 'PenalizedPlayer', 'HomeTeam', 'AwayTeam', 'Timeout\_Indicator',

'Timeout\_Team', 'Season', 'posteam\_timeouts\_pre', 'HomeTimeouts\_Remaining\_Pre',

'AwayTimeouts\_Remaining\_Pre', 'HomeTimeouts\_Remaining\_Post', 'AwayTimeouts\_Remaining\_Post']

name\_value = ['Drive', 'qtr', 'down', 'TimeUnder', 'TimeSecs', 'PlayTimeDiff', 'yrdln', 'yrdline100', 'ydstogo',

'ydsnet', 'GoalToGo', 'AirYards', 'YardsAfterCatch', 'FieldGoalDistance', 'Penalty.Yards',

'PosTeamScore', 'DefTeamScore', 'ScoreDiff', 'AbsScoreDiff', 'No\_Score\_Prob', 'Opp\_Field\_Goal\_Prob',

'Opp\_Safety\_Prob', 'Opp\_Touchdown\_Prob', 'Field\_Goal\_Prob', 'Safety\_Prob', 'Touchdown\_Prob',

'ExPoint\_Prob', 'TwoPoint\_Prob', 'ExpPts', 'EPA', 'airEPA', 'yacEPA', 'Home\_WP\_pre', 'Away\_WP\_pre',

'Home\_WP\_post', 'Away\_WP\_post', 'Win\_Prob', 'WPA', 'airWPA', 'yacWPA']

'''

#数据集2

name\_category = ['Permit Number', 'Permit Type', 'Permit Type Definition', 'Permit Creation Date', 'Block', 'Lot',

'Street Number', 'Street Number Suffix', 'Street Name', 'Street Suffix', 'Unit Suffix',

'Description', 'Current Status', 'Current Status Date', 'Filed Date', 'Issued Date',

'Completed Date', 'First Construction Document Date', 'Structural Notification', 'Fire Only Permit',

'Permit Expiration Date', 'Existing Use', 'Proposed Use', 'Plansets', 'TIDF Compliance',

'Existing Construction Type', 'Existing Construction Type Description',

'Proposed Construction Type', 'Proposed Construction Type Description','Site Permit',

'Neighborhoods - Analysis Boundaries', 'Zipcode', 'Location'];

name\_value = ['Unit', 'Number of Existing Stories', 'Number of Proposed Stories', 'Estimated Cost',

'Revised Cost', 'Existing Units', 'Proposed Units', 'Supervisor District'];

'''

# \*\*Step 2. 数据摘要\*\*

#

# - 对标称属性，给出每个可能取值的频数

# 使用value\_counts函数统计每个标称属性的取值频数

for item in name\_category:

print (item, '的频数为：\n', pd.value\_counts(DataTable[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 = DataTable[name\_value].max(), columns = ['max'])

# 最小值

data\_show['min'] = DataTable[name\_value].min()

# 均值

data\_show['mean'] = DataTable[name\_value].mean()

# 中位数

data\_show['median'] = DataTable[name\_value].median()

# 四分位数

#data\_show['quartile'] = DataTable[name\_value].describe().loc['quartile']

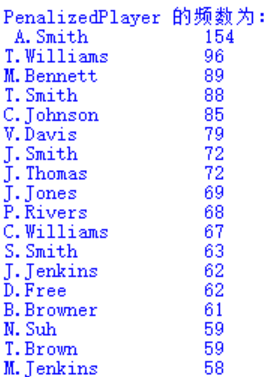
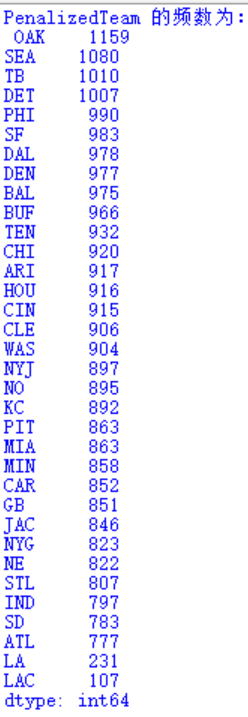
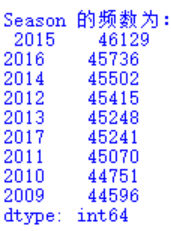
#data\_show['quartile']=stats.quantile(DataTable[name\_value],p=0.25)

# 缺失值个数

data\_show['missing'] = DataTable[name\_value].describe().loc['count'].apply(lambda x : 200-x)

print (data\_show)

部分结果展示：



# \*\*Step 3. 数据可视化 \*\*

#

# - 针对数值属性：

# 直方图

fig = plt.figure(figsize = (20,15))

i = 1

for item in name\_value:

ax = fig.add\_subplot(6, 7, i)

DataTable[item].plot(kind = 'hist', title = item, ax = ax)

i += 1

plt.subplots\_adjust(wspace = 0.3, hspace = 0.3)

fig.savefig('./image/histogram.png')

print ('histogram saved at ./image/histogram.png')

# qq图

fig = plt.figure(figsize = (20,15))

i = 1

for item in name\_value:ax = fig.add\_subplot(6, 7, i)

sm.qqplot(DataTable[item], ax = ax)

ax.set\_title(item)

i += 1

plt.subplots\_adjust(wspace = 0.3, hspace = 0.3)

fig.savefig('./image/qqplot.png')

print ('qqplot saved at ./image/qqplot.png')

# - 绘制盒图，对离群值进行识别。

# 盒图

fig = plt.figure(figsize = (20,15))

i = 1

for item in name\_value:

ax = fig.add\_subplot(6, 7, i)

DataTable[item].plot(kind = 'box')

i += 1

fig.savefig('./image/boxplot.png')

print ('boxplot saved at ./image/boxplot.png')

程序运行结果会存放于同一路径下image文件夹中，提交的结果文件夹中imageN与imageB分别存放两个数据集的运行结果。

# \*\*Step 4. 数据缺失的处理\*\*

# 找出含有缺失值的数据条目索引值

nan\_list = pd.isnull(DataTable).any(1).nonzero()[0]

# 4.1 将缺失部分剔除

data\_filtrated = DataTable

#DataTable=DataFile;

# 绘制可视化图

fig = plt.figure(figsize = (20,15))

i = 6

# 对数值属性，绘制直方图

for item in name\_value:

ax = fig.add\_subplot(8, 9, i)

#ax = fig.add\_subplot(4, 5, i)

data\_filtrated[item]=data\_filtrated[item].dropna()

ax.set\_title(item)

DataTable[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(DataTable[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.png')

print ('filted\_missing\_data1 saved at ./image/missing\_data\_delete.png')

# 4.2 最高频率值来填补缺失值

# 建立原始数据的拷贝

data\_filtrated = DataTable.copy()

# 对每一列数据，分别进行处理

for item in name\_category+name\_value:

# 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 = (20,15))

i = 1

# 对标称属性，绘制折线图

for item in name\_category:

ax = fig.add\_subplot(8, 9, i)

ax.set\_title(item)

pd.value\_counts(DataTable[item].values).plot(ax = ax, marker = '^', label = 'origin', legend = True)

pd.value\_counts(data\_filtrated[item].values).plot(ax = ax, marker = 'o', label = 'filtrated', legend = True)

i += 1

i = 6

# 对数值属性，绘制直方图

for item in name\_value:

ax = fig.add\_subplot(8, 9, i)

ax.set\_title(item)

DataTable[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(DataTable[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.png')

print ('filted\_missing\_data2 saved at ./image/missing\_data\_most.png')

# 4.3 通过属性的相关关系来填补缺失值

# 建立原始数据的拷贝

data\_filtrated = DataTable.copy()

# 对数值型属性的每一列，进行插值运算

for item in name\_value:

data\_filtrated[item].interpolate(inplace = True)

fig = plt.figure(figsize = (20,15))

i = 1

# 对标称属性，绘制折线图

for item in name\_category:

ax = fig.add\_subplot(8, 9, i)

ax.set\_title(item)

pd.value\_counts(DataTable[item].values).plot(ax = ax, marker = '^', label = 'origin', legend = True)

pd.value\_counts(data\_filtrated[item].values).plot(ax = ax, marker = 'o', label = 'filtrated', legend = True)

i += 1

i = 6

# 对数值属性，绘制直方图

for item in name\_value:

ax = fig.add\_subplot(8, 9, i)

ax.set\_title(item)

DataTable[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(DataTable[item].mean(), color = 'r')

ax.axvline(data\_filtrated[item].mean(), color = 'b')

i += 1

plt.subplots\_adjust(wspace = 0.3, hspace = 0.3)

fig.savefig('./image/missing\_data\_corelation.png')

print ('filted\_missing\_data3 saved at ./image/missing\_data\_corelation.png')

'''

# 4.4 通过数据对象之间的相似性来填补缺失值

data\_norm = DataTable.copy()

data\_norm[name\_value] = data\_norm[name\_value].fillna(0)

data\_norm[name\_value] = data\_norm[name\_value].apply(lambda x : (x - np.mean(x)) / (np.max(x) - np.min(x)))

# 构造分数表

score = {}

range\_length = len(DataTable)

for i in range(0, range\_length):

score[i] = {}

for j in range(0, range\_length):

score[i][j] = 0

for i in range(0, range\_length):

for j in range(i, range\_length):

for item in name\_category:

if data\_norm.iloc[i][item] != data\_norm.iloc[j][item]:

score[i][j] += 1

for item in name\_value:

temp = abs(data\_norm.iloc[i][item] - data\_norm.iloc[j][item])

score[i][j] += temp

score[j][i] = score[i][j]

data\_filtrated = DataTable.copy()

for index in nan\_list:

best\_friend = sorted(score[index].items(), key=operator.itemgetter(1), reverse = False)[1][0]

for item in name\_value:

if pd.isnull(data\_filtrated.iloc[index][item]):

if pd.isnull(DataTable.iloc[best\_friend][item]):

data\_filtrated.ix[index, item] = DataTable[item].value\_counts().idxmax()

else:

data\_filtrated.ix[index, item] = DataTable.iloc[best\_friend][item]

fig = plt.figure(figsize = (20,15))

i = 1

# 对标称属性，绘制折线图

for item in name\_category:

ax = fig.add\_subplot(4, 5, i)

ax.set\_title(item)

pd.value\_counts(DataTable[item].values).plot(ax = ax, marker = '^', label = 'origin', legend = True)

pd.value\_counts(data\_filtrated[item].values).plot(ax = ax, marker = 'o', label = 'filtrated', legend = True)

i += 1

i = 6

# 对数值属性，绘制直方图

for item in name\_value:

ax = fig.add\_subplot(4, 5, i)

ax.set\_title(item)

DataTable[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(DataTable[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.png')

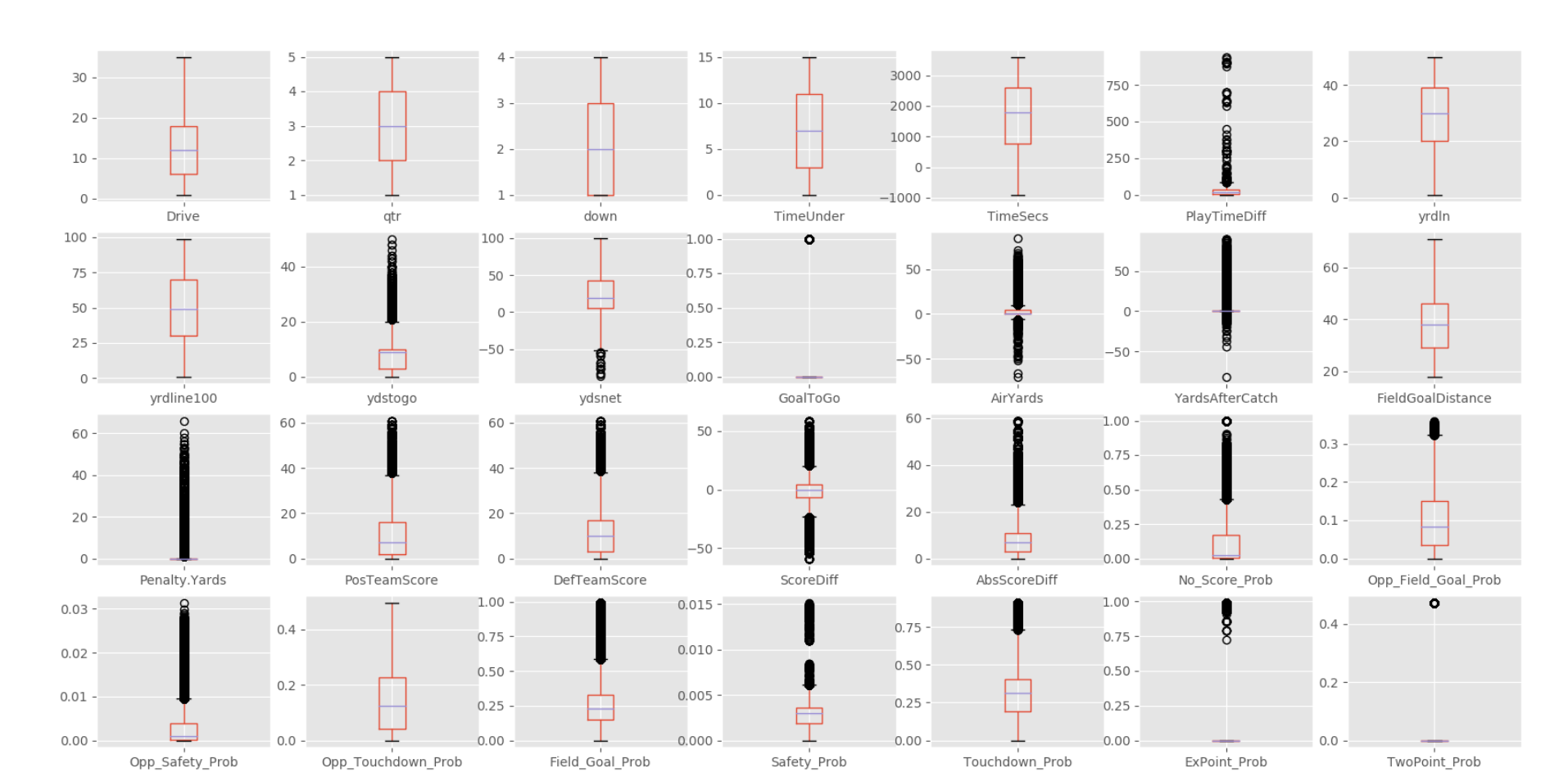
print ('filted\_missing\_data4 saved at ./image/filted\_missing\_data4.png')

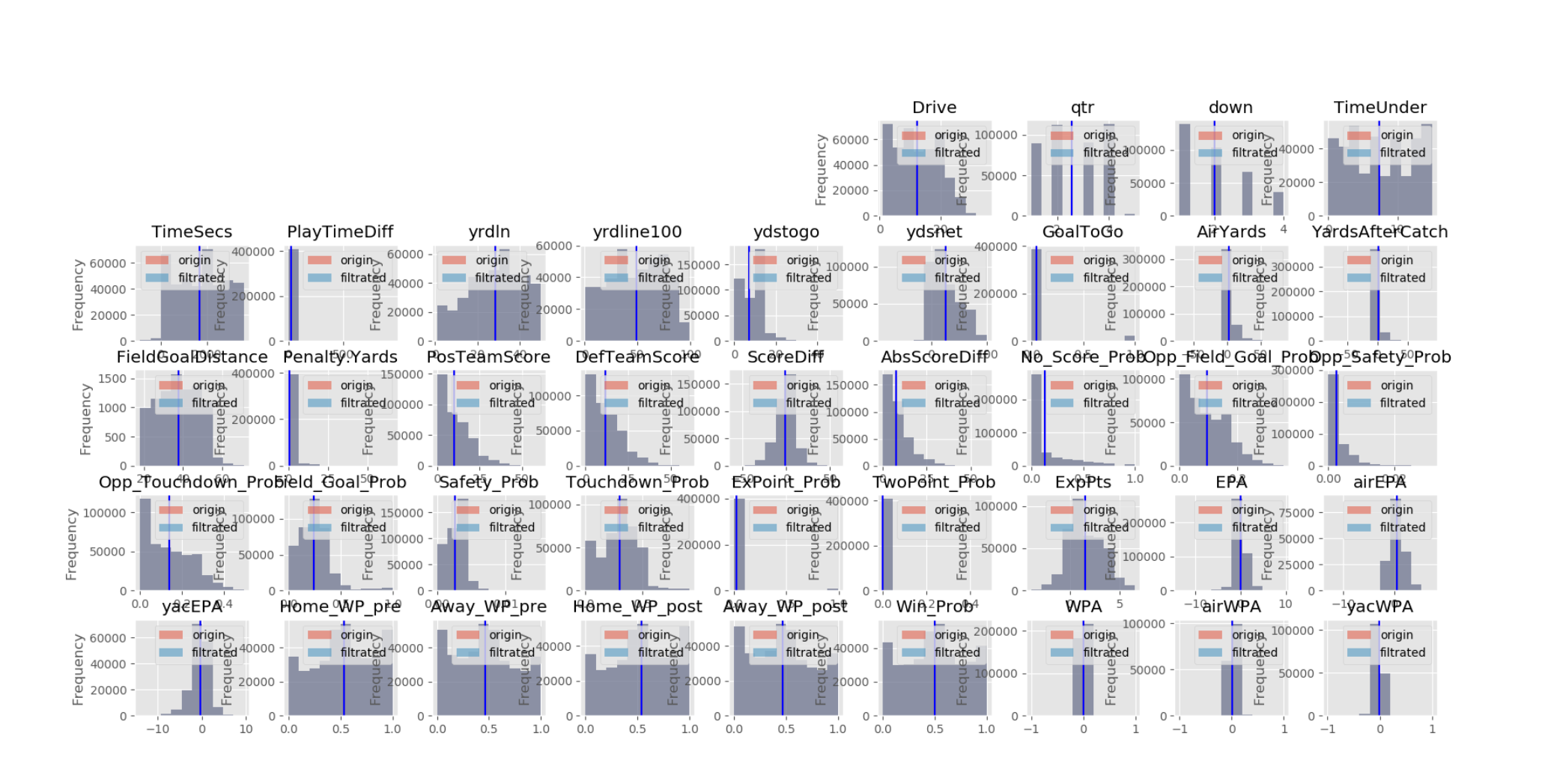
'''

下面进行部分的结果展示：

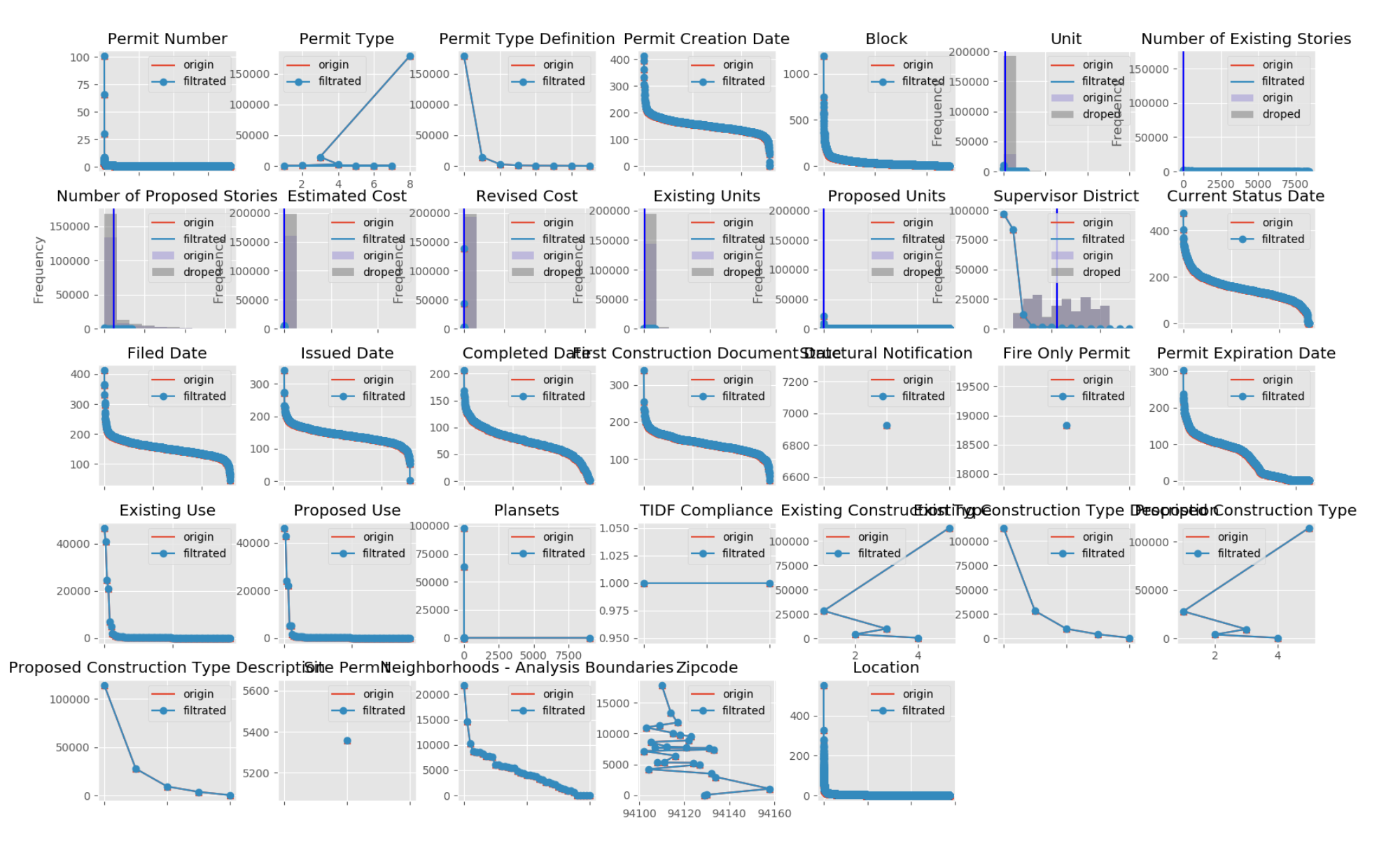
数据集一：

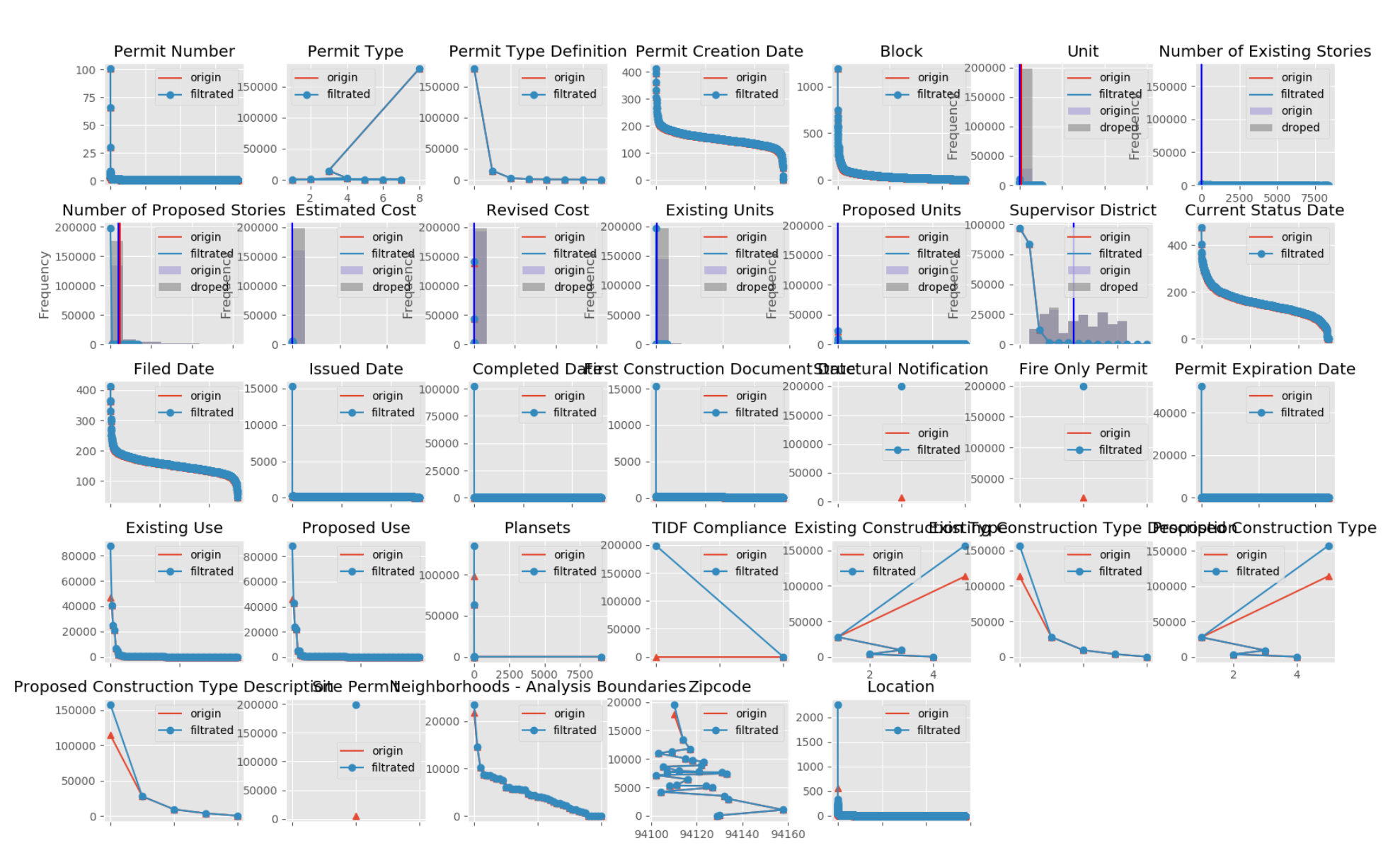
直方图与盒图



缺失数据剔除

数据集二：

通过相关关系填补缺失数据



用最高频率值来填补缺失

实验感想：

第一次接触如此庞大规模数据的处理，对我来说是一个挑战，python我也是通过此次实践才开始学习，慢慢摸索，一点点配置环境、模块，一个个bug的调试，终于小有雏形，在这个过程中我学到了很多，获益良多，但是与其他同学的差距还是显而易见的，有些功能还没有完全调试好，有时候还会有bug出现，结果显示也不够简洁美观，这也激励着我不断学习，不断完善自己！很期待后续内容的学习与实践。