DataMining_homework1 实验报告

姓名：张映雪

学号：2120171101

小组：第 6 组

目录

1. 实验要求

1.1　实验所用数据集

- NFL Play by Play 2009-2017 (v4).csv
- Building_Permits.csv

1.2 数据可视化和摘要

数据摘要

- 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

1.3 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。

2. 实验环境

- python2.7
- 分析数据所用的包： numpy、pandas、scipy
- 绘图所用的包：matplotlib

3. 实验结果分析

3.1 数据摘要

数据摘要可分为两部分，一部分针对标称属性，标称型目标变量的结果只在有限目标集中取值，如真与假。对这一部分属性，对其计算频数并输出，部分代码如下：

```
def nominal_statistic(csv_file, numeric_attr, name):
    result_dict = {}
    for column in csv_file.columns:
        if column not in numeric_attr:
            result_dict[column] = csv_file[column].value_counts().to_dict()
    json.dump(str(result_dict), open(r'process_result/'+name+'_frequency.json', 'w'))
```

另一部分针对数值属性，数值型目标变量则可以从无限的数值集合中取值，如 0.100，42.001 等，主要用于回归分析，对其计算最大、最小、均值、中位数、四分位数及缺失值的个数，部分代码如下：

```
def numeric_statistic(csv_file, numeric_attr, name):
    result_dict = {}
    for column in numeric_attr:
        column_series = copy.copy(csv_file[column])
        clean_series = column_series.dropna()

        num_of_NaN = column_series.__len__() - clean_series.__len__()

        clean_list = clean_series.values.tolist()

        clean_list.sort()
        len = clean_list.__len__()
        max_value = clean_list[-1]
        min_value = clean_list[0]
        sum_value = sum(clean_list)
        mean_value = sum_value / clean_list.__len__()

        Q1 = clean_list[int((len + 1) * 0.25)]
        Q2 = clean_list[int((len + 1) * 0.5)]
        Q3 = clean_list[int((len + 1) * 0.75)]

        result = [max_value, min_value, mean_value, Q2, [Q1, Q2, Q3], num_of_NaN]
        result_dict[column] = result
    json.dump(result_dict, open('process_result/'+name+'_numeric_result.json', 'w'))
```

3.1.1 数据集 NFL Play by Play 2009-2017 (v4).csv

标称属性：Date、GameID、Time、SideofField、GoalToGo、FirstDown、Posteam、Defensive Team、PlayAttempted、Sp、Touchdown、ExPointResult、TwoPointConv、DefTwoPoint、Safety、Onsidekick、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、Trackler1、Trackler2、FieldGoalResult、Fumble、RecFumbTeam RecFumbPlayer、Sack、Challenge.Replay、ChalReplayResult、Accepted.Renalty PenalizedTeam、PenaltyType、PenalizedPlayer、HomeTeam、AwayTeam、Timeout_Indicator、Timeout_Team、Season

数据处理结果存放在 NFL Play by Play 2009-2017 (v4)_frequency.json 文件中，文件内结构为{'Data'：frequency；'GameID'：frequency；……}

数值属性：Drive、Qtr、Down、TimeUnder、TimeSecs、PlayTimeDiff、Yrdln、Yrdline100、Ydstogo、Ydsnet、Yards.Gained、AirYards、YardsAfterCatch、FieldGoalDistance、Penalty.Yards、PosTeamScore、DefTeamScore、ScoreDiff、AbsScoreDiff、Posteam_timeouts_pre、HomeTimeouts_Remaining_Pre、AwayTimeouts_Remaining_Pre、HomeTimeouts_Remaining_Post、AwayTimeouts_Remaining_Post、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

数值属性的数据处理结果存放在 NFL Play by Play 2009-2017 (v4)_numeric_result.json 文件中，json 文件中结构{'Drive'：[max，min、mean，median，quarters，miss_num]；'Qtr'：[…….]；……}

3.1.2 数据集 Building_Permits.csv

标称属性：Permit Number、Permit Type、Permit Type Definition、Permit Creation Date、Block、Lot、Street Number、Street Number Suffix、Street Name、Street Name Suffix、Unit、Unit suffix、Description、Current Status、Current Status Date、

Filed Date、Issued Date、Completed Date、First Construction Document Date、Structural Notification、Voluntary Soft-Story 、Fire Only Permit、Permit Expiration Date 、 Plansets 、 TIDF Compliance 、 Existing Construction Type 、 Proposed Construction Type 、 Proposed Construction Type Description 、 Site Permit 、 Supervisor District、Neighborhoods - Analysis Boundaries、Zipcode、Location、Record ID

标称属性的数据处理结果存放在 Building_Permits_frequency.json 文件中，文件内结构为{'Permit Number'：frequency；'Permit Type'：frequency；……}

数值属性 :Existing Units、Number of Existing Stories、Number of Proposed Stories、Estimated Cost、Revised Cost、proposed Units

数值属性的数据处理结果存放在 Building_Permits_numerical_result.json 文件中，json 文件中结构{'Existing Units'：[max, min、mean, median, quarters, miss_num]；'Number of Existing Stories'：[…….]；……}

3.2 数据可视化

人类利用视觉获取的信息量，远远超出其他器官，大量视觉信息在潜意识阶段就被处理完成，人类对图像的处理速度比文本快 6 万倍。数据可视化正是利用人类天生技能来增强数据处理和组织效率。本次试验中，我们针对数值属性做可视化处理：绘制直方图，用 qq 图检验其分布是否为正态分布，部分代码如下：

```
def draw_hist(column, vc, loc):
    plt.figure(figsize=(2.8, 2))
    plt.title("Hist:" + str(column))
    plt.hist(vc, bins=20)
    plt.savefig(loc+'hist_'+column+'.png')
    plt.close()
    pass


def draw_qq_norm(column, vc, loc):
    plt.figure(figsize=(2.8, 2))
    stats.probplot(vc, dist="norm", plot=plt)
    plt.title("Q-Q:" + str(column))
    plt.savefig(loc+'qq_'+column+'.png')
    # plt.show()
    plt.close()
    pass
```

绘制盒图，对离群值进行识别，部分代码如下：

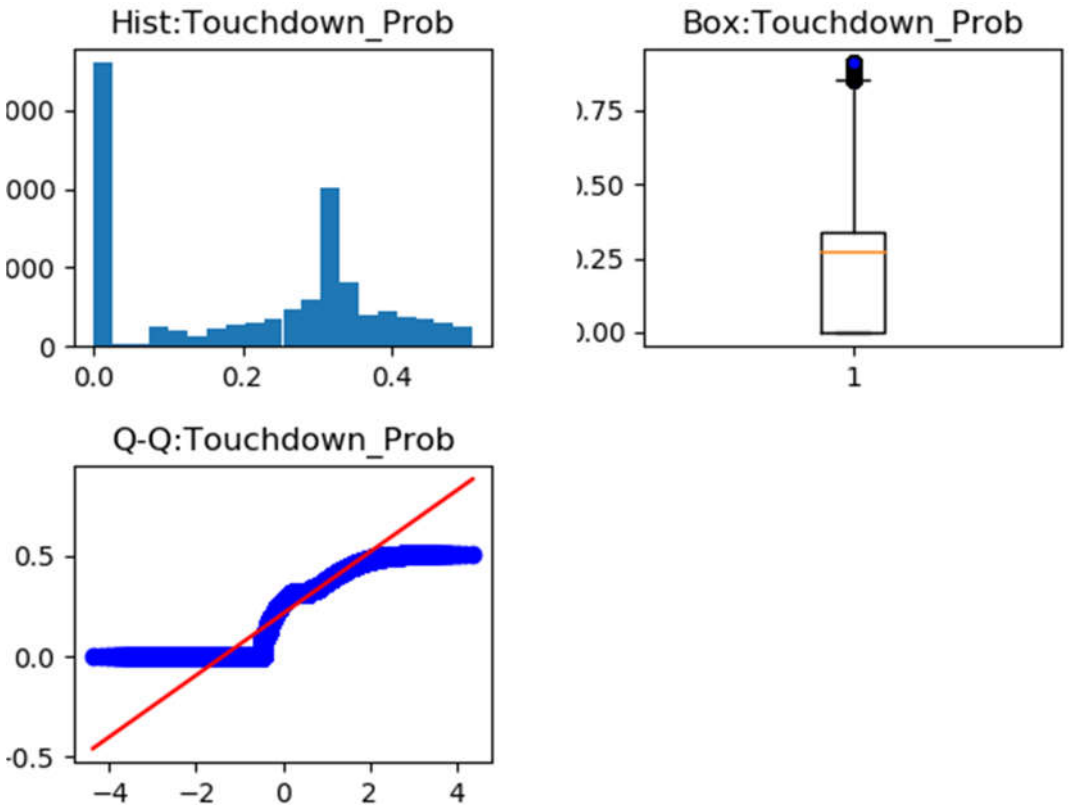
```python
def draw_box(column, values_clean, loc):
    plt.figure(figsize=(2.8,2))
    # 离散点（图标样式,图标颜色,大小,..）
    fp = {'marker': "o", 'markerfacecolor': 'blue', 'markersize': 5, 'linestyle': 'none'}
    plt.title('Box:' + str(column))
    plt.boxplot(values_clean, flierprops=fp)
    plt.savefig(loc+'box_'+column+'.png')
    # plt.show()
    plt.close()
    pass
```

3.2.1 数据集 NFL Play by Play 2009-2017 (v4).csv

由于该数据集数值属性数量过多，选取其中几个进行可视化展示

Opp_Field_Goal_Prob 属性：

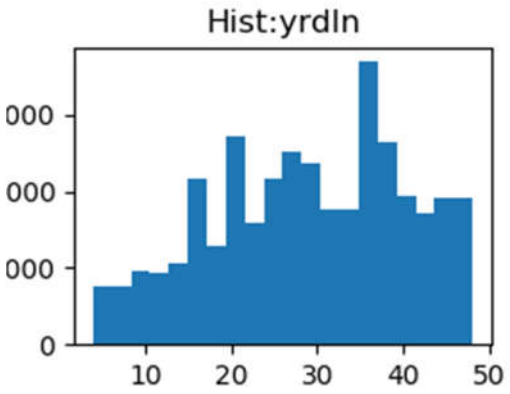
直方图、盒图以及 qq 图如下图

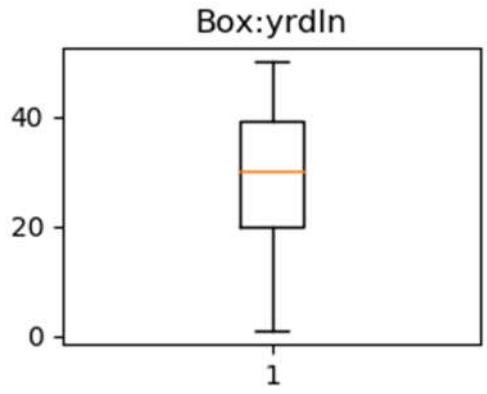


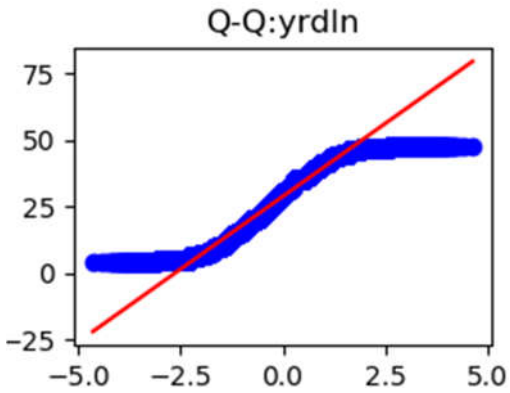
由上图我们可分析出：该属性的离散点很少，不满足正态分布；

'yrdln'属性：

直方图、盒图以及 qq 图如下图：



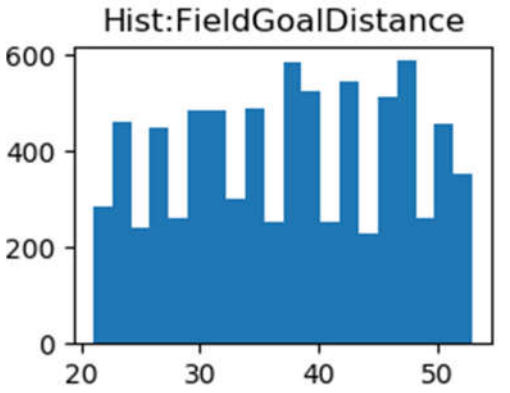


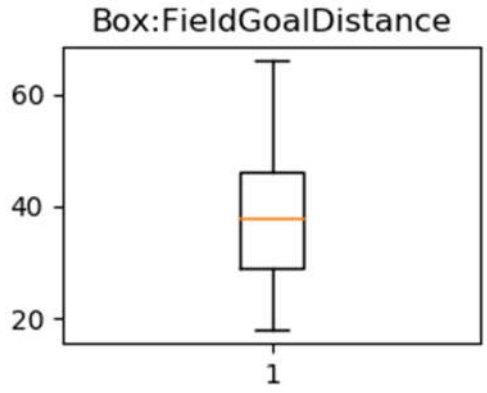


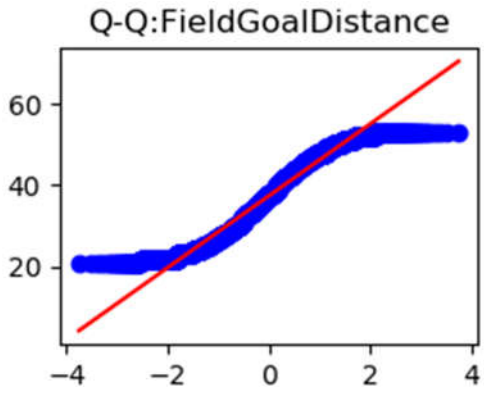
由图可知该属性离散点少，不满足正态分布；

FieldGoalDistance 属性：

直方图、盒图以及 qq 图如下图所示



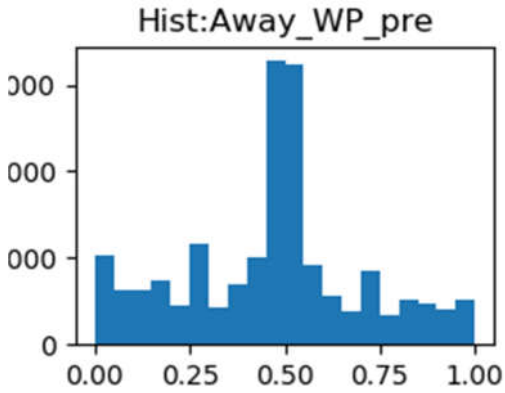


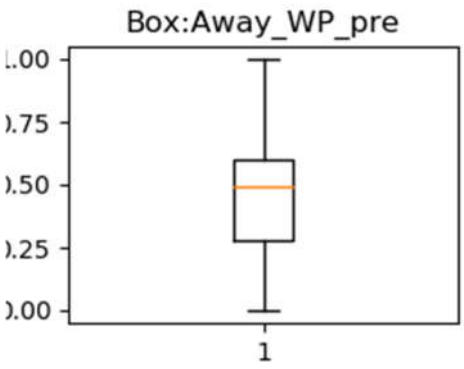


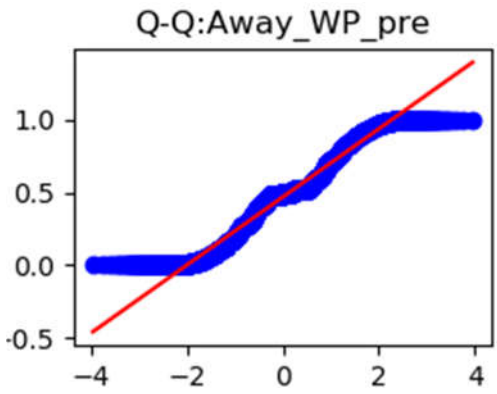
由图可知该属性离散点少，不满足正态分布；

Away_WP_pre 属性

直方图、盒图以及 qq 图如下所示，



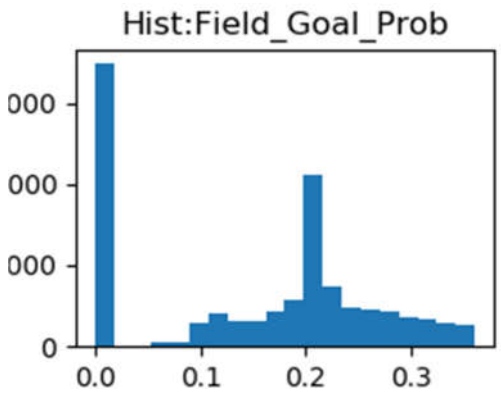


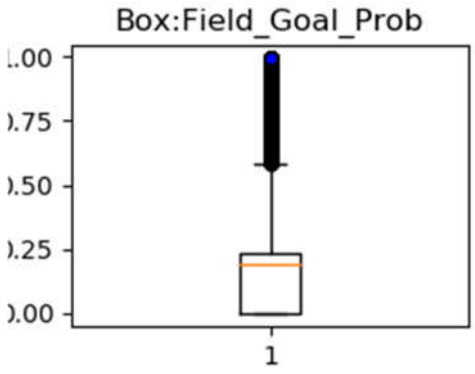


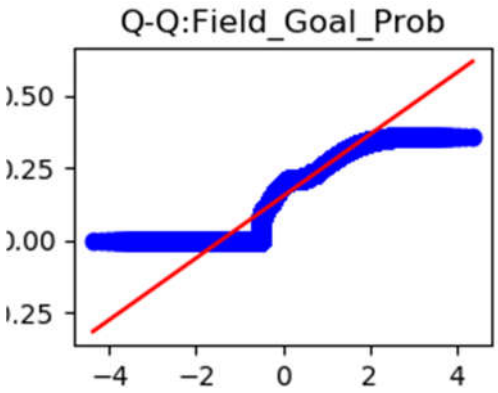
该属性离散点少，不满足正态分布；

Field_Goal_Prob 属性：

直方图、盒图以及 qq 图如下图所示



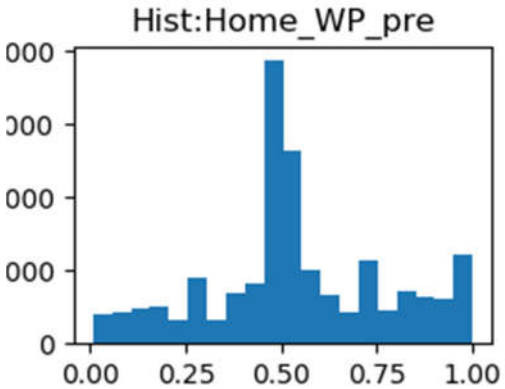


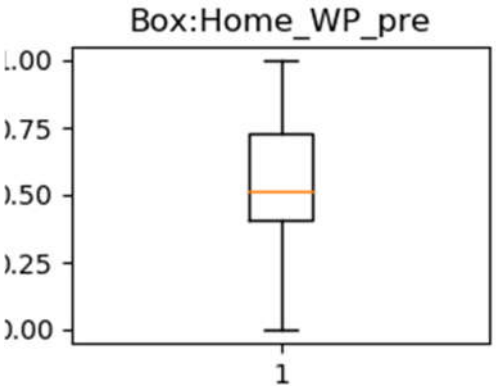


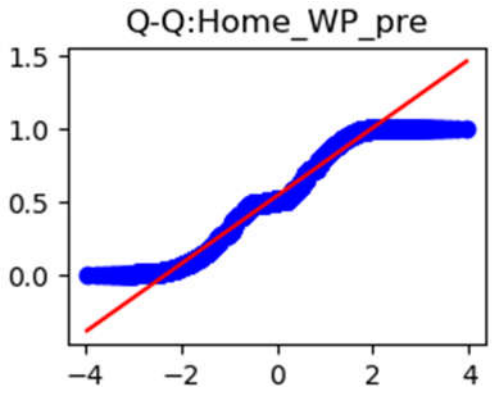
由图可知该属性离散点多，不满足正态分布；

Home_WP_pre 属性：

直方图、盒图以及 qq 图，







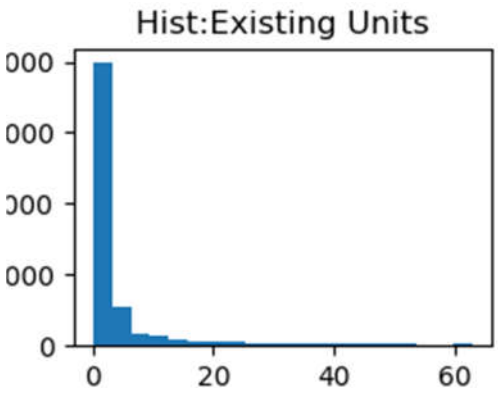
该属性离散点少，近似一个正态分布；

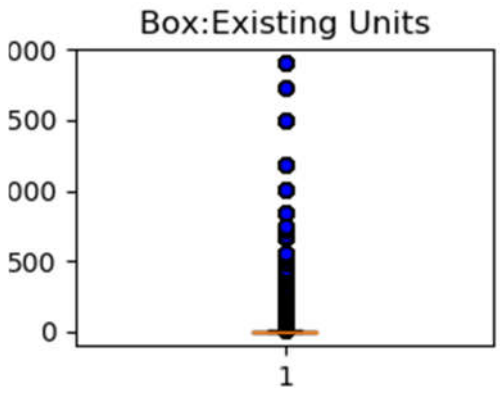
3.2.2 数据集 Building_Permits.csv

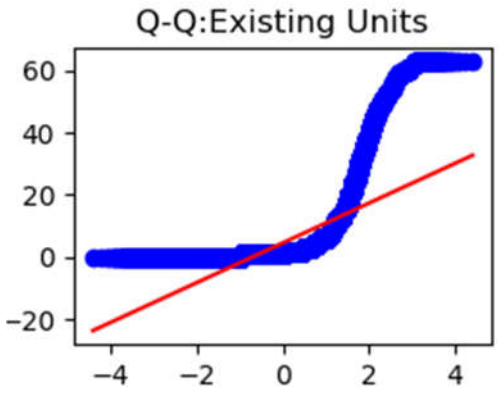
由于该数据集数值属性并不很多，我们依次展示各个属性：Existing Units、Number of Existing Stories、Number of Proposed Stories、Estimated Cost、Revised Cost、proposed Units

Existing Units 属性：

直方图、盒图以及 qq 图，



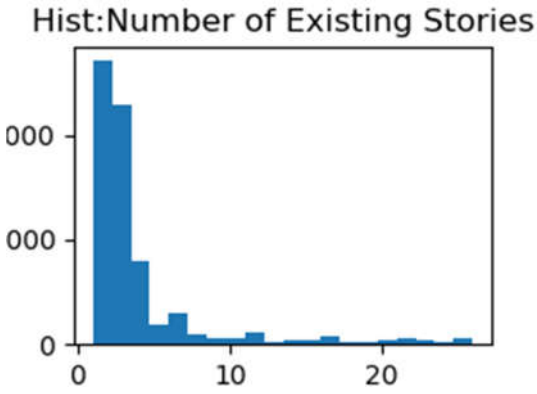


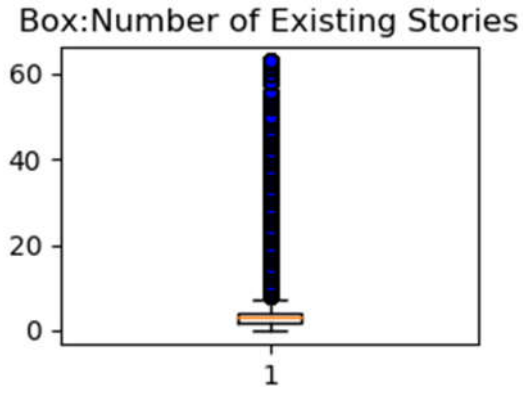


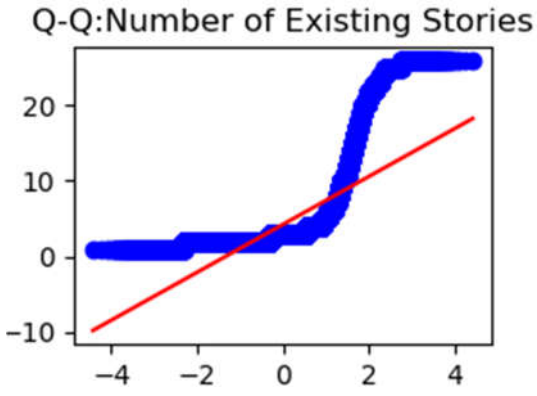
分析：该属性离散点较多，不满足正态分布；

Number of Exiting Stories 属性

直方图、盒图以及 qq 图：



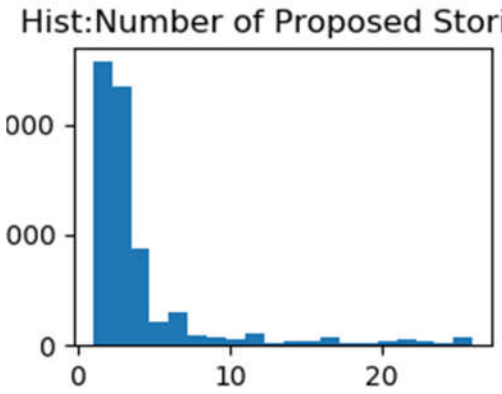


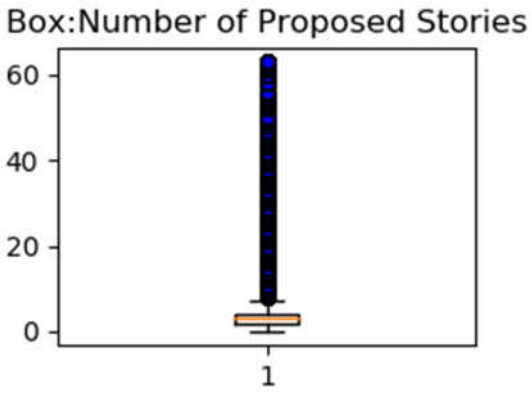


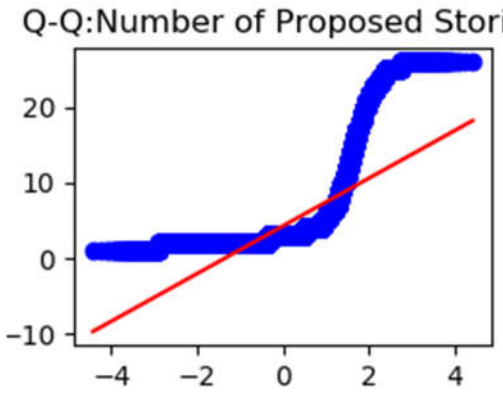
分析：该属性离散点较多，不满足正态分布；

Number of Proposed Stories 属性：

直方图、盒图以及 qq 图，



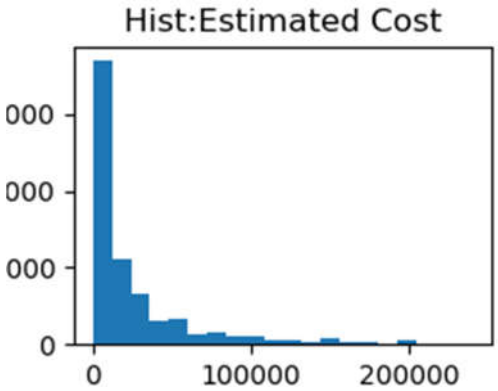


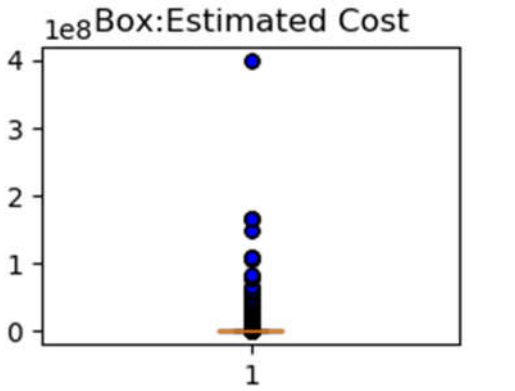


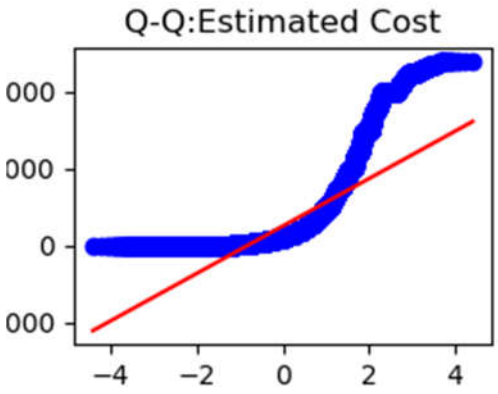
分析：该属性离散点较多，不满足正态分布；

Estimated Cost 属性:

直方图、盒图以及 qq 图，



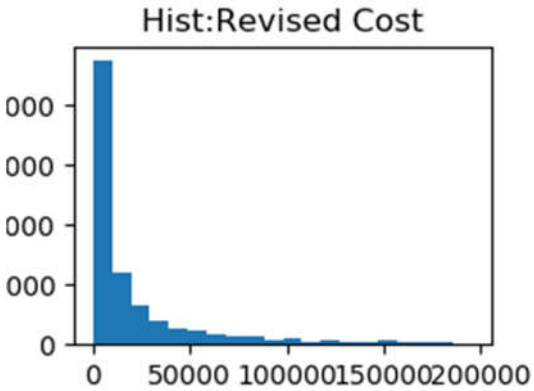


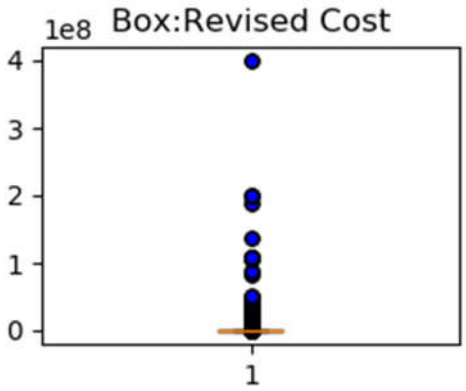


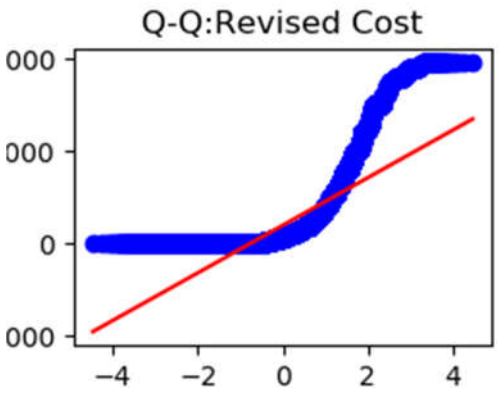
分析：该属性离散点较多，不满足正态分布；

Revised Cost 属性:

直方图、盒图以及 qq 图:



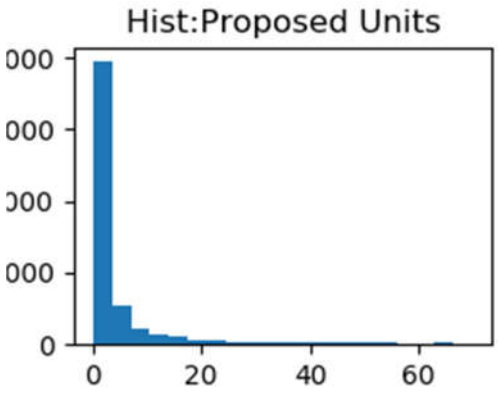


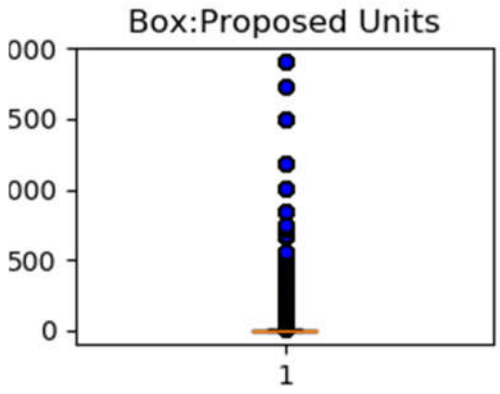


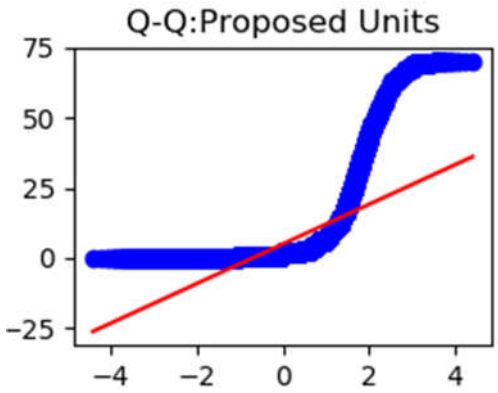
分析：该属性离散点较多，不满足正态分布；

Proposed Units 属性：

直方图、盒图以及 qq 图







分析：该属性离散点较多，不满足正态分布；

3.3 数据缺失处理

现实世界中的数据异常杂乱，属性值缺失的情况经常发全甚至是不可避免的。经过实验分析造成数据缺失的原因是多方面的：比如信息暂时无法获取。也可能是因为输入时认为不重要、忘记填写了或对数据理解错误而遗漏，也可能是由于数据采集设备的故障、存储介质的故障、传输媒体的故障、一些人为因素等原因而丢失。而且还有可能有些对象的某个或某些属性是不可用的。如一个未婚者的配偶姓名、一个儿童的固定收入状况等等。在本次实验中，我们采用四种缺失处理的方法。

将缺失部分剔除：

```
def complete_dropna(csv_file, column):
    loc = "diagram/diagram_after_completion/type1_"
    values_dropna = csv_file[column].dropna().values
    draw_hist(column, values_dropna, loc)
    draw_qq_norm(column, values_dropna, loc)
    draw_box(column, values_dropna, loc)
```

用最高频率值来填补缺失值：

```
def complete_fre_attr(csv_file, column):
    value_count = csv_file[column].dropna().value_counts()
    max_fre_value = value_count.index[0]
    data = csv_file[column]
    miss_index = data[data.isnull()].index
    complete_data = data.copy()
    for i in miss_index:
        complete_data[i] = max_fre_value

    loc = "diagram/diagram_after_completion/type2_"
    draw_hist(column, complete_data, loc)
    draw_qq_norm(column, complete_data, loc)
    draw_box(column, complete_data, loc)
```

通过属性的相关关系来填补缺失值；

```
def complete_rel_attr(csv_file, double_column):
    target_data = csv_file[double_column[0]]
    source_data = csv_file[double_column[1]]
    flag1 = target_data.isnull().values
    flag2 = source_data.isnull().values
    complete_data = target_data.copy()
    for index, value in target_data.iteritems():
        if flag1[index] == True and flag2[index] == False:

            complete_data[index] = 1 - source_data[index]

    values_clean = list(complete_data.dropna().values)
    for value, count in complete_data.value_counts().iteritems():
        if count == 1:
            values_clean.remove(value)

    loc = "diagram/diagram_after_completion/type3_"
    draw_hist(double_column[0], values_clean, loc)
    draw_qq_norm(double_column[0], values_clean, loc)
    draw_box(double_column[0], values_clean, loc)
```

通过数据对象之间的相似性来填补缺失值：

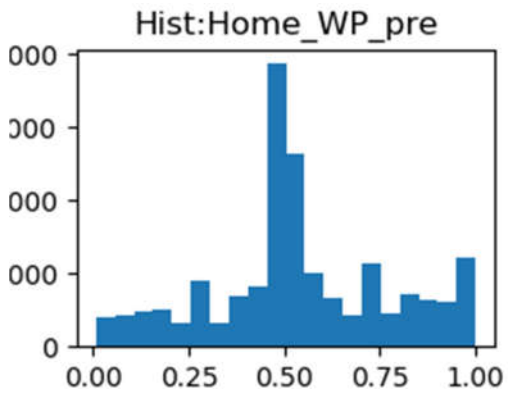
```
def complete_smi_attr(csv_file, column, numeric_attr):
    data = csv_file[column].copy()
    for index, value in data.iteritems():
        if value == np.NaN:
            data[index] = data[find_dis_value(csv_file, index, column, numeric_attr)]
    loc = "diagram/diagram_after_completion/type4_"
    draw_hist(column, data.dropna().values, loc)
    draw_qq_norm(column, data.dropna().values, loc)
    draw_box(column, data.dropna().values, loc)
```

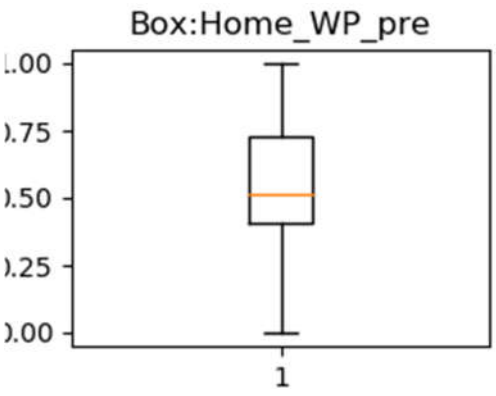
3.3.1 数据集 NFL Play by Play 2009-2017 (v4).csv

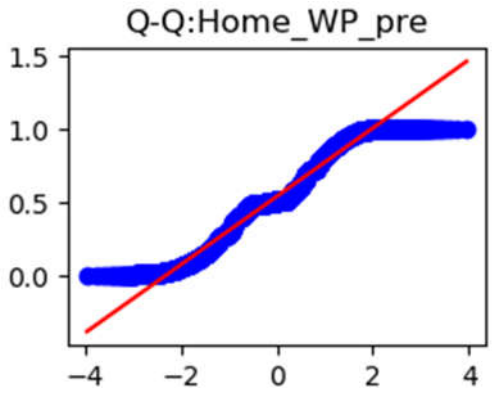
由于数值属性的数量相对较多且有多个属性存在缺失值，在这里展示 Home_WP_pre 的缺失处理过程和结果。

分别使用下列四种策略对缺失值进行处理：

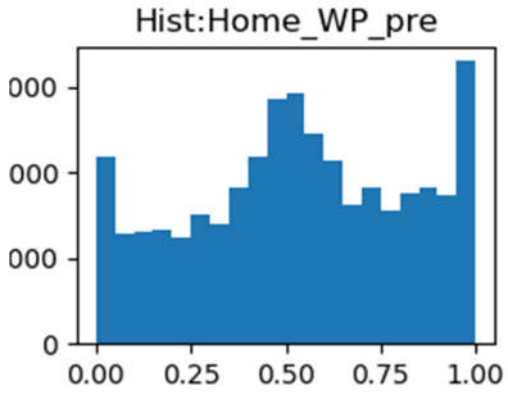
- 缺失处理之前：直方图、盒图以及 qq 图，

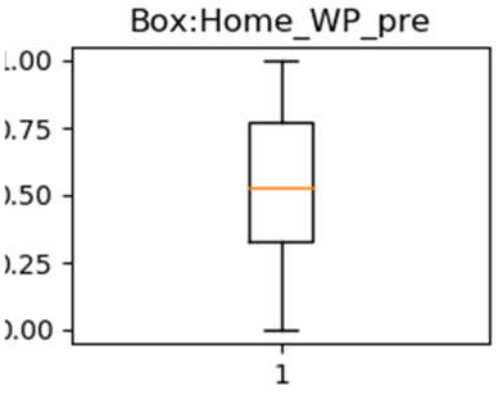


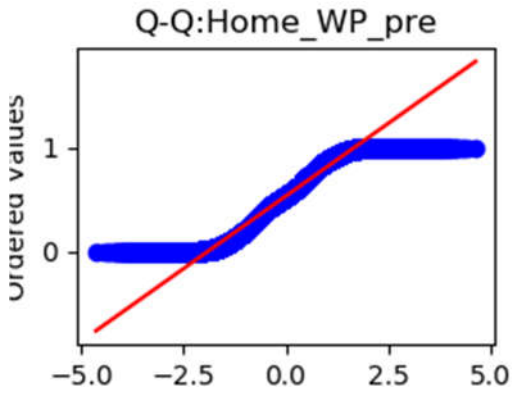




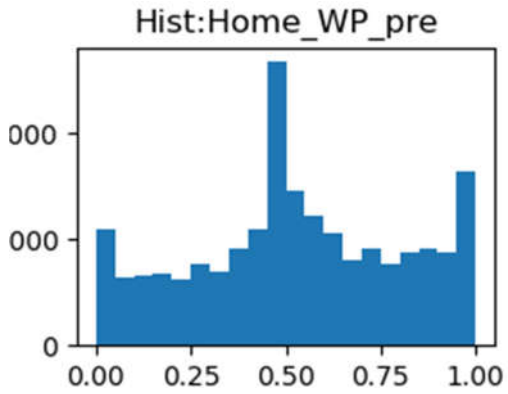
- 将缺失部分剔除，直方图、盒图、qq 图:

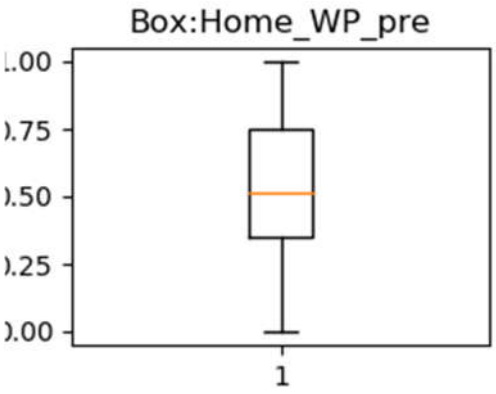


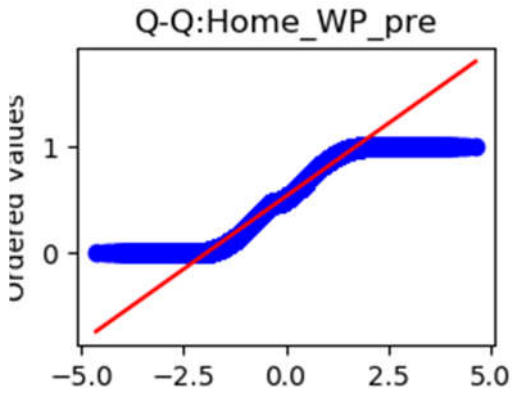




- 用最高频率值来填补缺失值，直方图、盒图、qq 图:

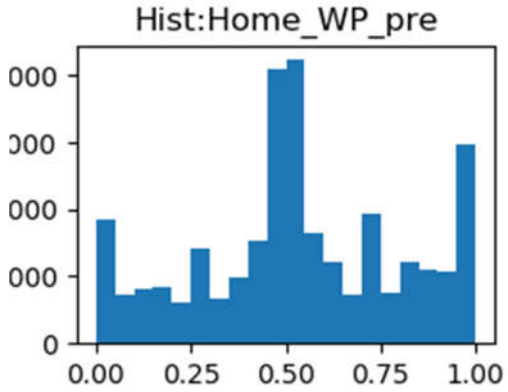


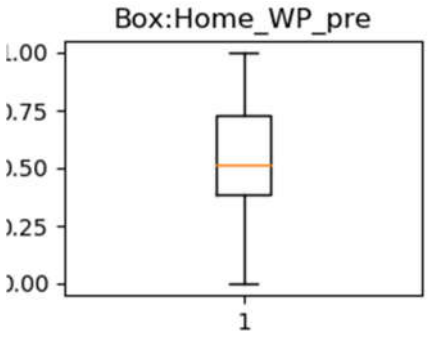


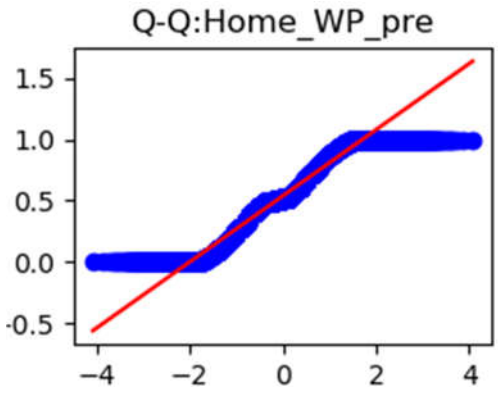


- 通过属性的相关关系来填补缺失值：

Home_WP_pre 属性直方图、盒图、qq 图如下所示:

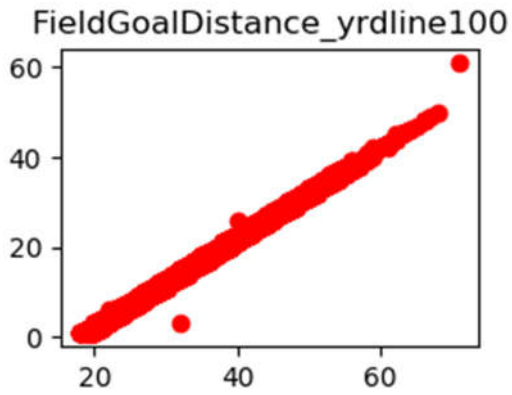






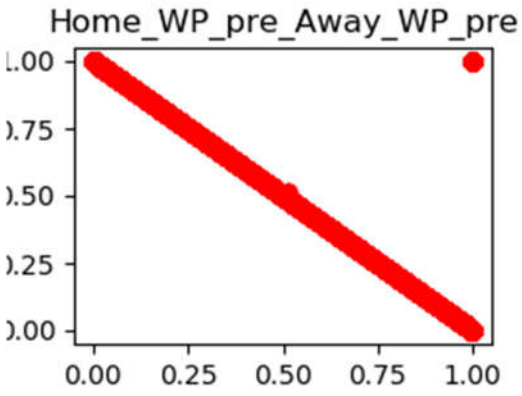
在这些数值属性中，可以分析出属性间存在或负或正的相关性。我们仅拿出其中几个进行举例：

'FieldGoalDistance'属性和'yrdline100'属性呈正相关：

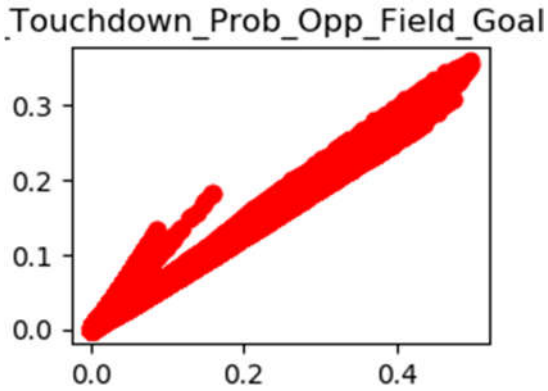


'Away_WP_pre'属性和'Home_WP_pre'属性呈负相关,且满足 value(Away_WP_pre) = 1-value(Home_WP_pre)

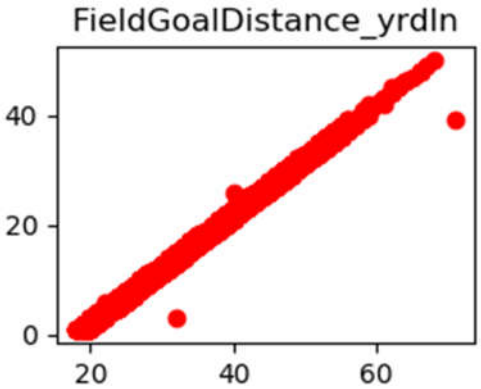
：



'Touchdown_Prob'属性'Opp_Field_Goal_Prob'属性呈正相关：

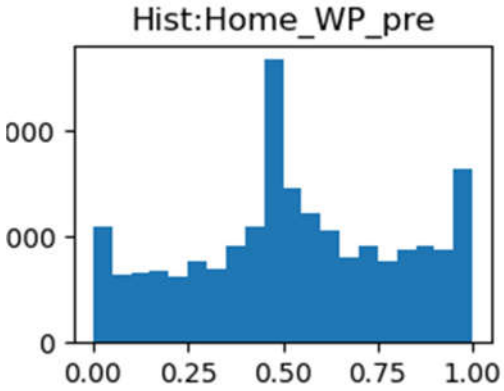


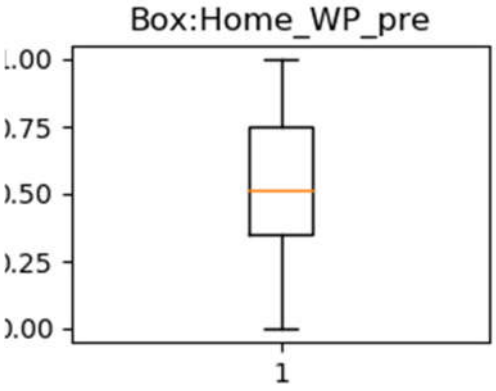
'FieldGoalDistance'属性和'yrdln'属性呈正相关。

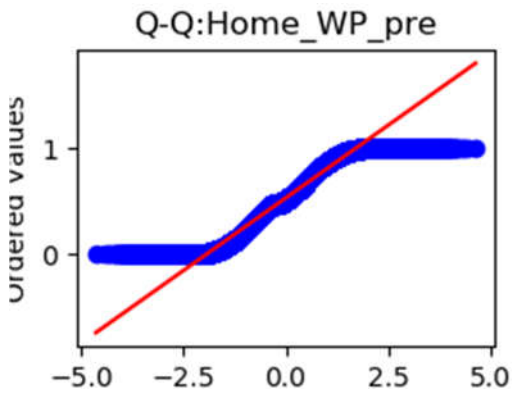


- 通过数据对象之间的相似性来填补缺失值，此处是利用混合属性的相异性度量方法来等效计算相似性的。

  直方图、盒图、qq 图如下所示:





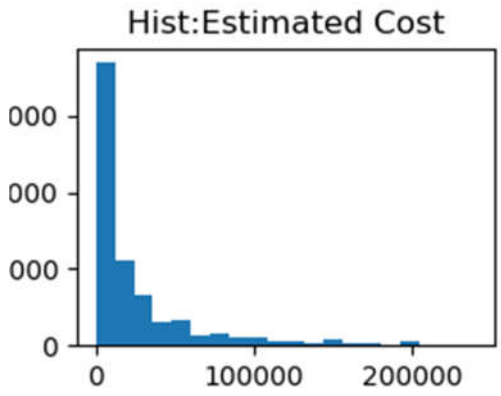


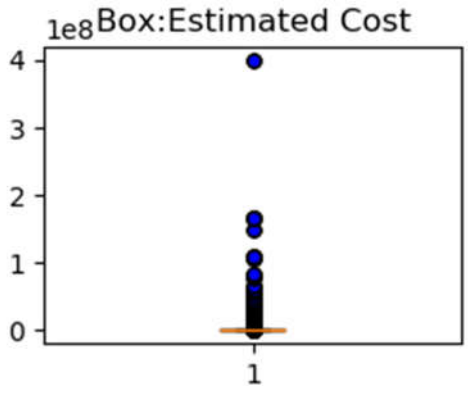
3.3.2 数据集 Building_Permits.csv

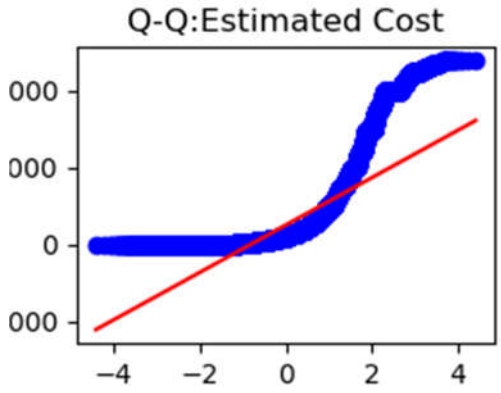
由于数值属性的数量相对较多且有多个属性存在缺失值，故只展示 Estimated Cost 这个属性的缺失处理结果。

分别使用下列四种策略对缺失值进行处理：

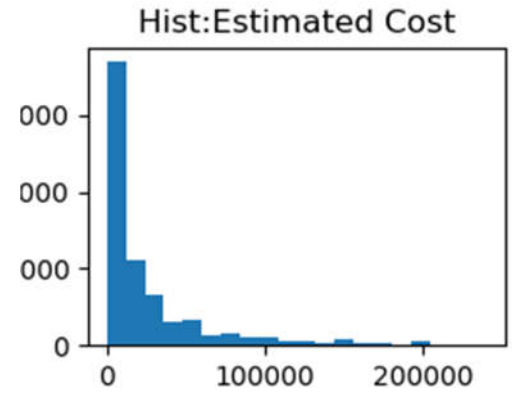
- 缺失处理之前，Estimated Cost 属性的直方图、盒图以及 qq 图：

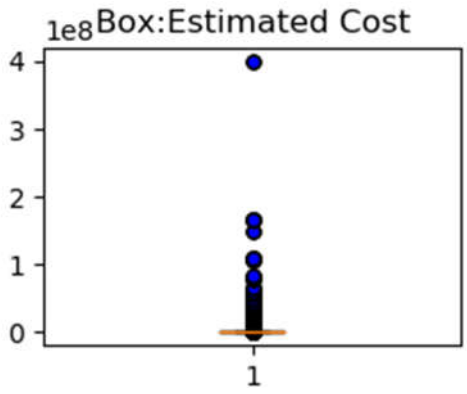


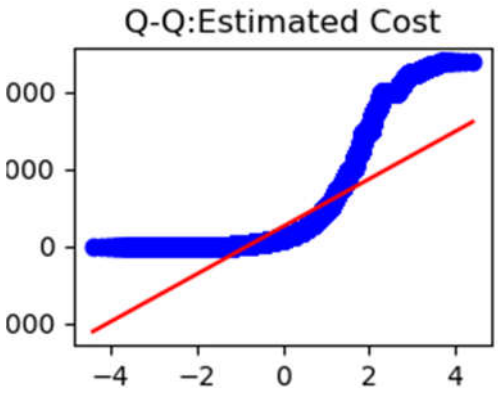




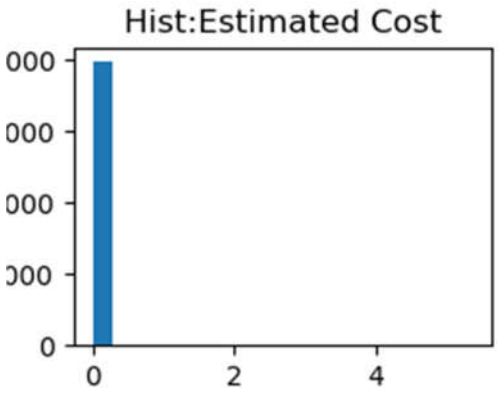
- 将缺失部分剔除，Estimated Cost 属性直方图、盒图、qq 图如下所示：

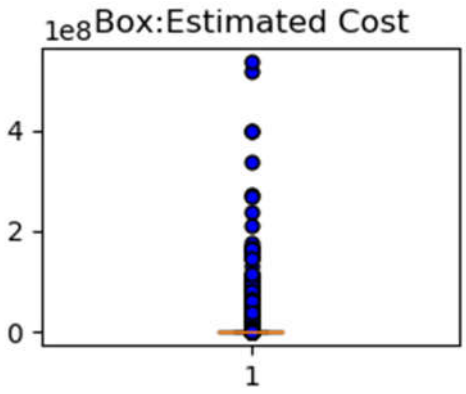


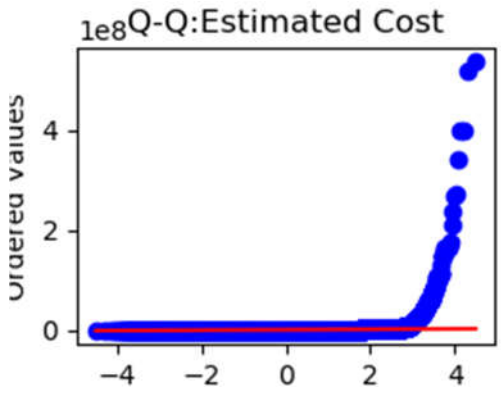




- 用最高频率值来填补缺失值：

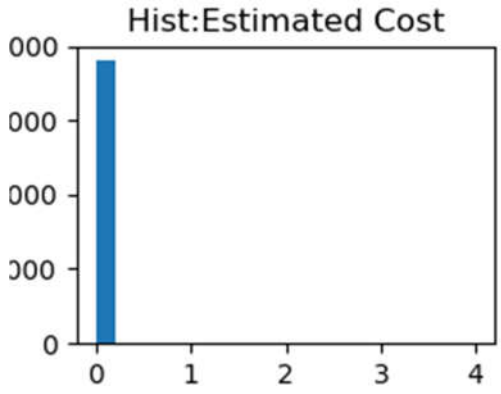


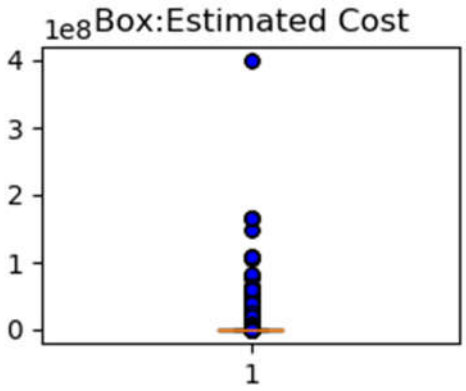


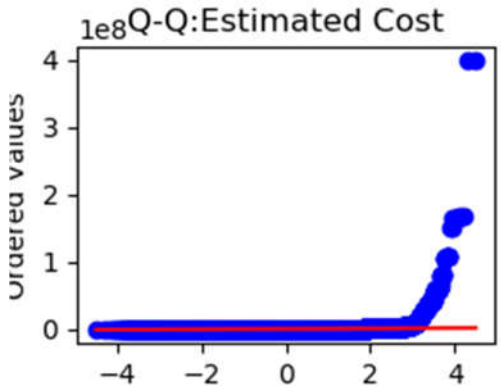


因为缺失值很多，全部填补之后会使得众数更多，使得直方图中其他数值显得很小。

- 通过属性的相关关系来填补缺失值：

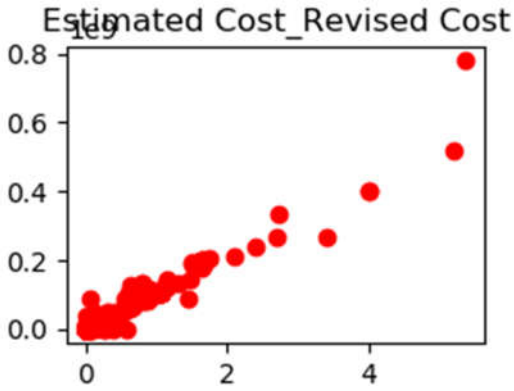




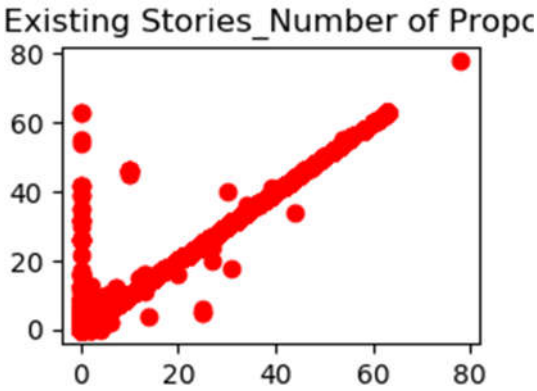


在这些数值属性中，可以分析出属性间存在或负或正的相关性。我们仅拿出其中几个进行举例：

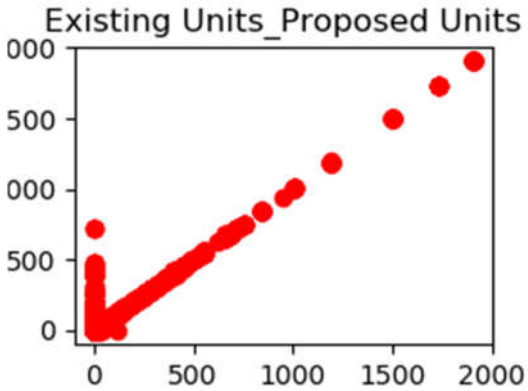
'Estimated Cost'属性和'Revised Cost'属性呈正相关，且成 1:1 正比，即 value(Estimated Cost) = value(Revised Cost)。则可以按此比例来填补 Estimated Cost 属性的缺失值。



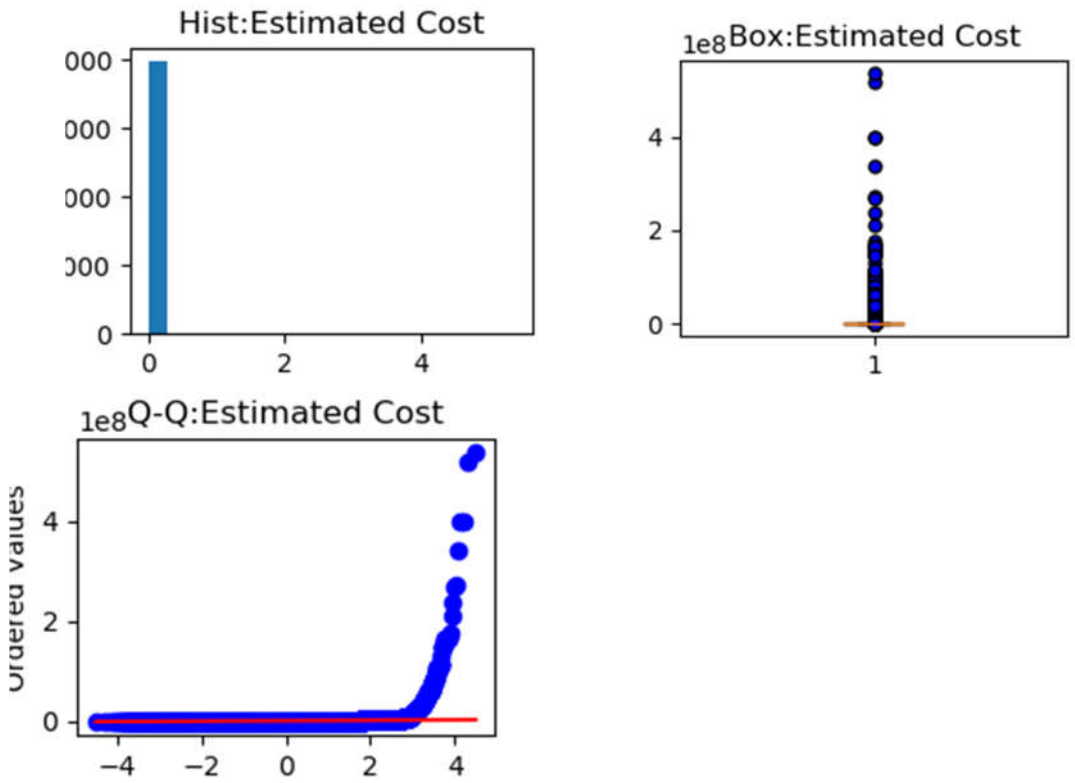
'Number of Existing Stories'属性'Number of Proposed Stories'属性呈正相关。



'Existing Units'属性和'Proposed Units'属性呈正相关：



- 通过数据对象之间的相似性来填补缺失值，此处是利用混合属性的相异性度量方法来等效计算相似性的。



4. 实验总结

经过本次实验，我熟悉了进行数据摘要、数据可视化处理、数据缺失处理的一般流程。更加深刻的体会到了数据可视化处理和缺失处理的必要性。数据可视化正是利用人类天生技能来增强数据处理和组织效率，而数据缺失在许多研究领域都是一个复杂的问题。对数据挖掘来说，缺省值的存在，会造成以下影响： 系统丢失大量的有用信息；系统中所表现出的不确定性更加显著，系统中蕴涵的确定性成分更难把握；包含空值的数据会使挖掘过程陷入混乱，导致不可靠的输出。数据挖掘算法本身更致力于避免数据过分拟合所建的模型，这一特性使得它难以通过自身的算法去很好地处理不完整数据。因此，缺省值需要通过专门的方法进行推导、填充等，以减少数据挖掘算法与实际应用之间的差距。同时我也意识到自己还有更多关于数据挖掘的算法需要去探索和学习，更加深入才能真正体会到数据处理的魅力。