数据探索性分析与数据预处理

姓名：李坤 学号：3220201059

一、 概述：

本次作业中，自行选择 2 个数据集进行探索性分析与预处理。

- 数据可视化和摘要。

  (1) 标称属性，给出每个可能取值的频数。

  (2) 数值属性，给出 5 数概括及缺失值的个数。

  (3) 使用直方图、盒图等检查数据分布及离群点。

- 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:

  (1) 将缺失部分剔除

  (2) 用最高频率值来填补缺失值

  (3) 通过属性的相关关系来填补缺失值

  (4) 通过数据对象之间的相似性来填补缺失值。

  (5) 处理后，可视化地对比新旧数据集。

二、 数据说明：

- **数据集 1：Wine Reviews**

  包括 winemag-data_first150k.csv、winemag-data-130k-v2.csv

- **数据集 2：Oakland Crime Statistics 2011 to 2016**

  包括 records-for-2011.csv、records-for-2012.csv、records-for-2013.csv、records-for-2014.csv、records-for-2015.csv、records-for-2016.csv。

三、 数据分析过程：

3.1 数据可视化和摘要

数据摘要：

通过 pandas 的 read_csv 方法将 csv 格式数据读入为 DataFrame 格式数据，通过判断数据类型为 int64 或者 float64 判断列的属性是否为数值属性。

数据集 Wine Reviews 的 winemag-data_first150k.csv 文件，其包含 8 个标称属性：'country'，'description'，'designation'，'province'，'region_1'，'region_2', 'variety', 'winery'；包含 2 个数值属性：'points', 'price'。

数据集 Wine Reviews 的 winemag-data-130k-v2.csv 文件，其包含 11 个标称属性：'country'，'description'，'designation'，'province'，'region_1'，'region_2', 'taster_name', 'taster_twitter_handle', 'title', 'variety', 'winery'；包含 2 个数值属性：'points', 'price'。

数据集 Oakland Crime Statistics 的 records-for-2011.csv 和 records-for-2013.csv 文件，包含 8 个标称属性：'Agency', 'Create Time', 'Location', 'Beat', 'Incident Type Id', 'Incident Type Description', 'Event Number', 'Closed Time'；包含 2 个数值属性：'Area Id', 'Priority'。

数据集 Oakland Crime Statistics 的 records-for-2012.csv 和 records-for-2014.csv 文件，包含 8 个标称属性： 'Agency', 'Create Time', 'Beat', 'Incident Type Id', 'Incident Type Description', 'Event Number', 'Closed Time', 'Location 1'；包含 2 个数值属性：'Area Id', 'Priority'。

数据集 Oakland Crime Statistics 的 records-for-2015.csv 和 records-for-2016.csv 文件，包含 9 个标称属性：'Agency', 'Create Time', 'Location', 'Area Id', 'Beat', 'Incident Type Id', 'Incident Type Description', 'Event Number', 'Closed Time'；包含 1 个数值属性：'Priority'。

通过在 Data 类的 process_features()函数判断列的属性后分别使用 process_nom_features()函数和 process_num_features()函数来对列的标称属性和数值属性进行处理。

```
for title in content.columns.values:
    if title == "Unnamed: 0":  # 未命名0列
        continue
    if content[title].dtypes == "int64" or content[title].dtypes == "float64":
        self.process_num_features(content, title, write_data_path)  # 处理数值属性
    else:
        self.process_nom_features(content, title, write_data_path)   # 处理标称属性
```

对于标称属性，process_nom_features()函数使用 dict 返回该属性对应数据的各个取值及其频数，获得的对应结果位于 nominal_attribute 下。

```
Feature Name: country
Value Num: 43
Italy,19540
Portugal,5691
US,54504
Spain,6645
France,22093
Germany,2165
Argentina,3800
Chile,4472
Australia,2329
Austria,3345
```

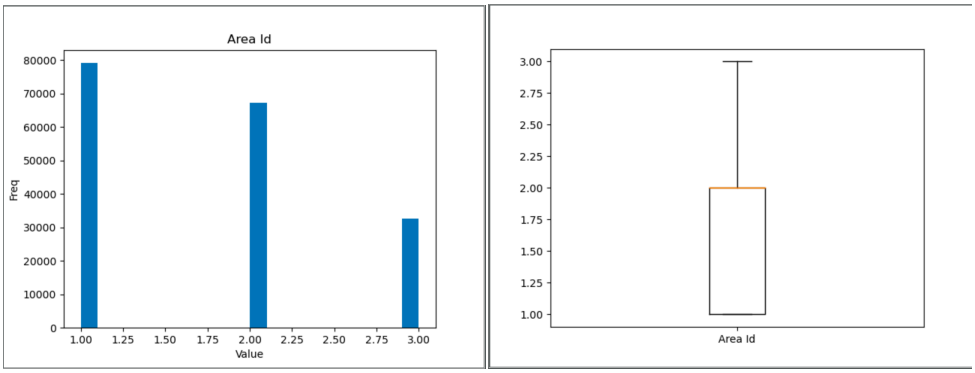
对于数值属性，process_num_features()函数使用 DataFrame 自带的函数返回该属性的最大、最小、均值、中位数、四分位数及缺失值的个数，获得的对应结果位于 numeric_attribute 下。

```
Feature Name: price
Max Num: 3300.0
Min Num: 4.0
Mean Num: 35.363389129985535
Median Num: 25.0
Quartile Num: 17.0, 42.0
Missing Num: 8996
```

数据可视化：

对于两个数据集的全部属性，在 draw_figure()函数中分别调用 draw_histogram()和 draw_box()函数绘制直方图和盒图。

绘图结果分别位于 figure 目录，如下图所示:



3.2 数据缺失的处理

使用四种策略对数值属性的缺失值进行处理，数据缺失的原因可能为：

- 记录数据时缺失
- 有些对象的某些属性无法获取
- 某些对象的该属性本身就是 None
- 某些对象的属性在数据分析中不重要，所以没有获取

分别使用下列四种策略对缺失值进行处理：

（1）将缺失部分剔除

（2）用最高频率值来填补缺失值

（3）通过属性的相关关系来填补缺失值

（4）通过数据对象之间的相似性来填补缺失值。

3.2.1 将缺失部分剔除

使用 pandas 的 dropna()函数可直接将包含 nan 的数据删掉。

```
# 填充策略1：剔除缺失的内容
strategy_path = os.path.join(write_data_path, "strategy_1")
makedir(strategy_path)
with open(os.path.join(strategy_path, file_name), 'w', encoding='utf-8') as fp1:
    strategy1_content = content
    for title in content.columns.values:
        if title == "Unnamed: 0":
            strategy1_content = strategy1_content.drop(columns=[title])
        elif strategy1_content[title].dtypes == "int64" or strategy1_content[title].dtypes == "float64":
            strategy1_content = strategy1_content.dropna(subset=[title])
            self.draw_figure(strategy1_content, title, strategy_path)
```

3.2.2 用最高频率值来填补缺失值

得到该属性所有可能的取值及其频数，取其最大值后使用 pandas 的 fillna()函数来填补缺失值，具体实现如下：

```
# 填充策略2：用最高频率值来填补缺失值
strategy_path = os.path.join(write_data_path, "strategy_2")
makedir(strategy_path)
with open(os.path.join(strategy_path, file_name), 'w', encoding='utf-8') as fp2:
    strategy2_content = content
    for title in content.columns.values:
        if title == "Unnamed: 0":
            strategy2_content = strategy2_content.drop(columns=[title])
        elif strategy2_content[title].dtypes == "int64" or strategy2_content[title].dtypes == "float64":
            value_dict = self.get_feature_value(strategy2_content, title)
            filling_data = max(value_dict, key=value_dict.get)
            strategy2_content = strategy2_content.fillna({title:filling_data})
            self.draw_figure(strategy2_content, title, strategy_path)
```

3.2.3 通过属性的相关关系来填补缺失值

使用属性的相关关系来填补缺失值，使用 pandas 的 interpolate()函数来填补缺失值，具体实现如下：

```
# 通过属性的相关关系来填补缺失值
strategy_path = os.path.join(write_data_path, "strategy_3")
makedir(strategy_path)
with open(os.path.join(strategy_path, file_name), 'w', encoding='utf-8') as fp3:
    strategy3_content = content
    strategy3_content = strategy3_content.interpolate(kind='nearest')
    for title in content.columns.values:
        if title == "Unnamed: 0":
            continue
        elif strategy3_content[title].dtypes == "int64" or strategy3_content[title].dtypes == "float64":
            self.draw_figure(strategy3_content, title, strategy_path)
```

3.2.4 通过数据对象之间的相似性来填补缺失值

使用数据对象之间的相似性来填补缺失值，考虑的是数据对象之间的相关性，使用 sklearn 的 KNeighborsClassifier 和 KNeighborsRegressor 函数来实现，具体实现如下：

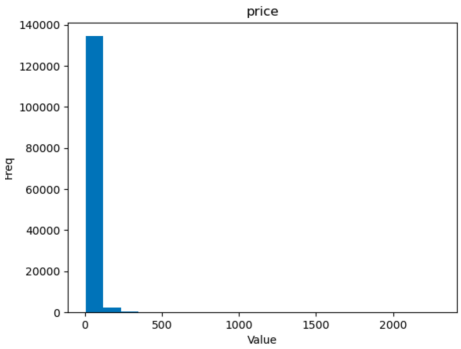
```
# 使用k近邻算法，计算填充内容
def knn_missing_filled(self, x_train, y_train, test, k=3, dispersed=True):
    if dispersed:
        clf = KNeighborsClassifier(n_neighbors=k, weights="distance")
    else:
        clf = KNeighborsRegressor(n_neighbors=k, weights="distance")

    clf.fit(x_train, y_train)
    return test.index, clf.predict(test)
```

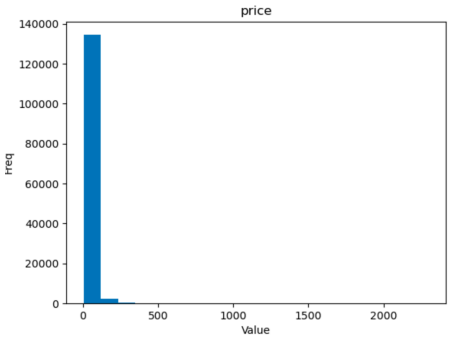
3.2.5 可视化对比

经过填补的数据分别绘制直方图和盒图，并将绘图结果与原数据进行对比。在 draw_figure()函数中分别调用 draw_histogram()和 draw_box()函数绘制直方图和盒图。

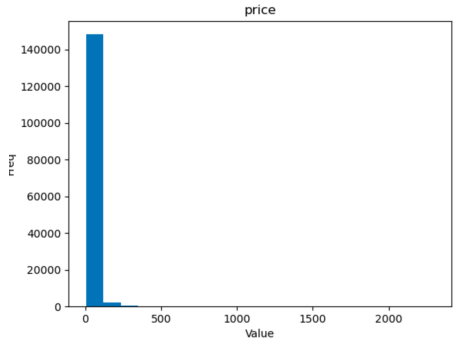
数据集 wine-review 的 winemag-data_first150k.csv 文件的’ price’ 属性可视化对比结果为:



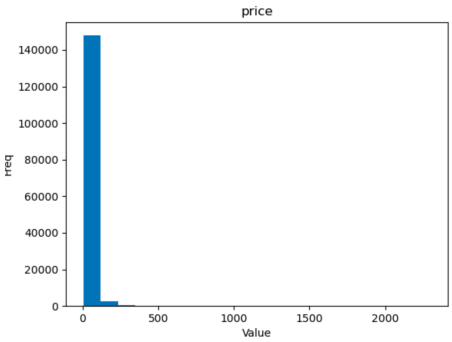
1.1 原始数据



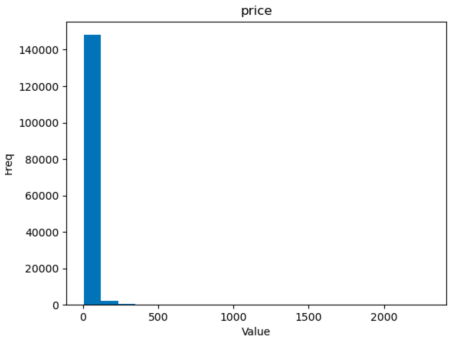
1.2 剔除 nan 数据



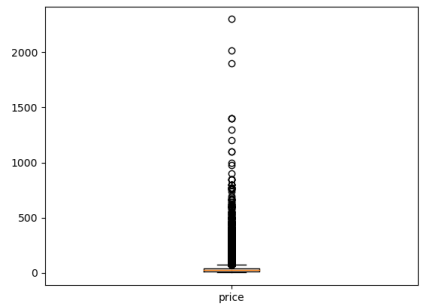
1.3 使用最高频率数值进行填充



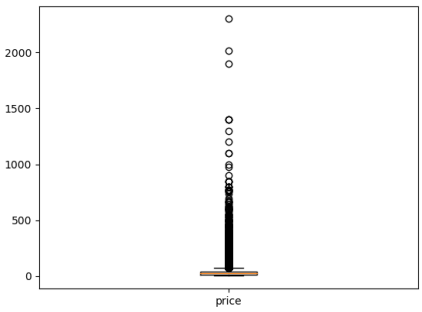
1.4 使用属性的相关关系填充



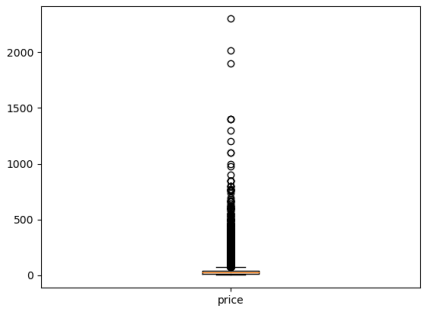
1.5 数据对象之间的相似性来填补缺失值



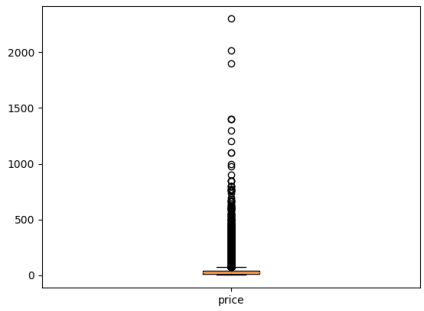
2.1 原始数据



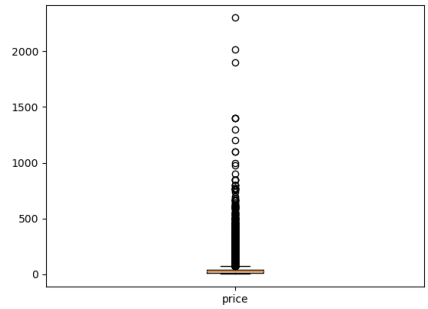
2.2 剔除 nan 数据



2.3 使用最高频率数值进行填充



2.4 使用属性的相关关系填充



2.5 数据对象之间的相似性来填补缺失值