数据挖掘互评作业 1：数据探索性分析与数据预处理

姓名：曹健　　学号：3120190978

一．数据可视化和摘要

1. 数据集选择：Wine Reviews 和 Consumer & Visitor Insights For Neighborhoods
2. 数据集描述：

数据集 Wine Reviews：

winemag-data-130k-v2.csv 包含 13 列和 130k 行葡萄酒评论

序号	属性	含义
1	Country	葡萄酒产地
2	Description	介绍
3	Designation	所属葡萄园
4	Points	WineEnthusiast 对葡萄酒的评分
5	Price	单瓶葡萄酒成本
6	Province	葡萄酒来源的省或州
7	Region_1	一个省或州的葡萄酒种植区
8	Region_2	有时在葡萄酒种植区域中指定了更多特定区域，但此值有时可能为空
9	Taster_name	
10	Taster_twitter_handle	
11	Title	评论的标题
12	Variety	葡萄种类
13	Winery	所属酿酒厂

标称属性：country，designation，province，region_1，region_2，taster_name，taster_twitter_handle，variety，winery；
数值属性：points，price

winemag-data_first150k.csv 包含 10 列和 150k 行葡萄酒评论
与上一个数据集相比缺少 taster_name，taster_twitter_handle，title 三个属性。
标称属性：country，designation，province，region_1，region_2，variety，winery；
数值属性：points，price

数据集 Consumer & Visitor Insights For Neighborhoods：

cbg_patterns.csv 文件，包含 13 列 22 万条数据

序号	属性	含义
1	census_block_group	人口普查区组唯一的 12 位 FIPS 码
2	date_range_start	测量周期的开始时间作为时间戳 (UTC 秒)
3	date_range_end	测量周期的时间结束作为时间戳 (UTC 秒)

4	raw_visit_count	在此日期范围内，我们的专家组对该 CBG 的访问次数
5	raw_visitor_count	在日期范围内，我们的小组访问 POI 的独立访问人数
6	visitor_home_cbgs	此列列出了在 census_block_group 列（目标 CBG）中列出的 CBG 中访问过目的地的设备的所有原始归属 CBG
7	visitor_work_cbgs	此列列出了在 census_block_group 列（目标 CBG）中列出的 CBG 中访问过目标的设备的所有工作位置 CBG。
8	distance_from_home	访客（我们确定住所的访客）到 CBG 的平均距离（以米为单位）
9	related_same_day_brand	这些 CBG 的访问者在访问 CBG 的同一天访问的品牌
10	related_same_month_brand	访问 CBG 的客户在访问 CBG 的同一个月访问的品牌
11	top_brands	在此期间，访问过 CBG 的顶级品牌列表。 仅限十大品牌。
12	popularity_by_hour	在当地时间范围内，每天的小时数与每小时的访问量之间的映射
13	popularity_by_day	在日期范围内，将星期几映射为每天的访问量(当地时间)。

标称属性：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；
数值属性：raw_visit_count，raw_visitor_count，distance_from_home

3. 数据集摘要（**以数据集 Wine Reviews 为例**）
针对标称属性，给出每个可能取值的频数
使用 pandas 包的 read_csv()方法将 csv 文件输入并存为 dataframe 格式。

```
df1 = pd.read_csv(data_file1)
```

按列分别取出各标称属性：

```
def Nominal(df1, filename):
    resdic = {}
    col1 = df1.country.unique()      #country
    col6 = df1.province.unique()     #province
    col7 = df1.region_1.unique()     #region_1
    col8 = df1.region_2.unique()     #region_2
    col9 = df1.variety.unique()       #variety
    col10 = df1.winery.unique()       #winery
```

然后通过 list 类型统计频数：

```
for each in col1:
    resdic["country"][each] = list(df1.country).count(each)
for each in col6:
    resdic["province"][each] = list(df1.province).count(each)
for each in col7:
    resdic["region_1"][each] = list(df1.region_1).count(each)
for each in col8:
    resdic["region_2"][each] = list(df1.region_2).count(each)
for each in col9:
    resdic["variety"][each] = list(df1.variety).count(each)
for each in col10:
    resdic["winery"][each] = list(df1.winery).count(each)
```

针对标称属性，给出 5 数概括及缺失值的个数（最大值、最小值、均值、中位数、四分位数）

选择需要处理的数值属性的两列构建新的 dataframe，调用 describe()方法得到对数值属性列的重要指标描述。

```
def Numeric(Ndf, filename):
    resdic = {}
    describe = Ndf.describe()
```

```
Numeric(pd.DataFrame(dataframe, columns=["points", "price"]), "result-Numeric-{}.json".format(id))
```

得到的指标值如下所示：

	points	price
count	150930.000000	137235.000000
mean	87.888418	33.131482
std	3.222392	36.322536
min	80.000000	4.000000
25%	86.000000	16.000000
50%	88.000000	24.000000
75%	90.000000	40.000000
max	100.000000	2300.000000

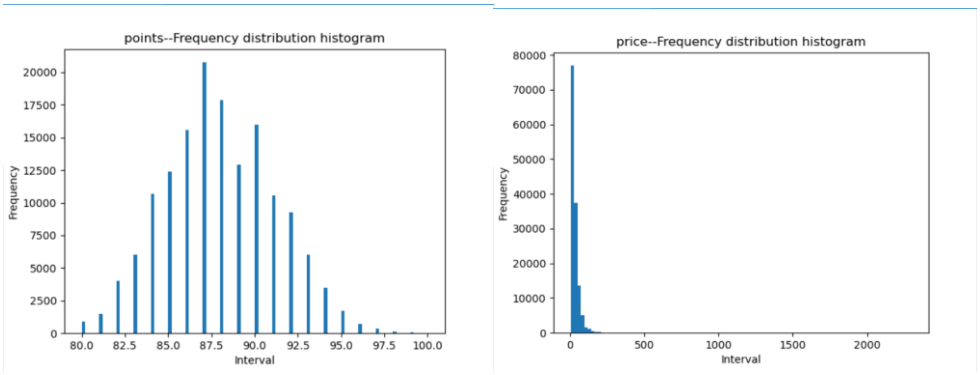
同时通过对每一个数值属性列判断空值并统计个数，得到缺失值的个数：

```
resdic["col4"]["Max"] = describe.points[7]
resdic["col4"]["Min"] = describe.points[3]
resdic["col4"]["Mean"] = describe.points[1]
resdic["col4"]["Mid"] = describe.points[5]
resdic["col4"]["25"] = describe.points[4]
resdic["col4"]["75"] = describe.points[6]
resdic["col4"]["Nan"] = Ndf.points.isna().sum()
print(resdic["col4"]["Nan"])

resdic["col5"]["Max"] = describe.price[7]
resdic["col5"]["Min"] = describe.price[3]
resdic["col5"]["Mean"] = describe.price[1]
resdic["col5"]["Mid"] = describe.price[5]
resdic["col5"]["25"] = describe.price[4]
resdic["col5"]["75"] = describe.price[6]
resdic["col5"]["Nan"] = Ndf.price.isna().sum()
print(resdic["col5"]["Nan"])
```

4. 数据可视化（**以数据集 Wine Reviews 为例**）

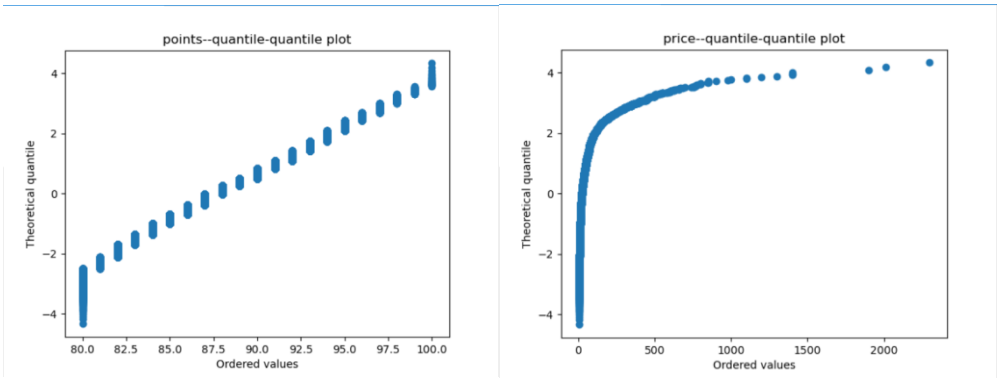
绘制直方图：

```
plt.hist(Ndf["points"], bins=100)
plt.xlabel("Interval")
plt.ylabel("Frequency")
plt.title("points--Frequency distribution histogram")
plt.show()


#去空值
print(len(Ndf))
Ndf = Ndf.dropna(subset=["price"])
print(len(Ndf))
plt.hist(Ndf["price"], bins=100)
plt.xlabel("Interval")
plt.ylabel("Frequency")
plt.title("price--Frequency distribution histogram")
plt.show()
```



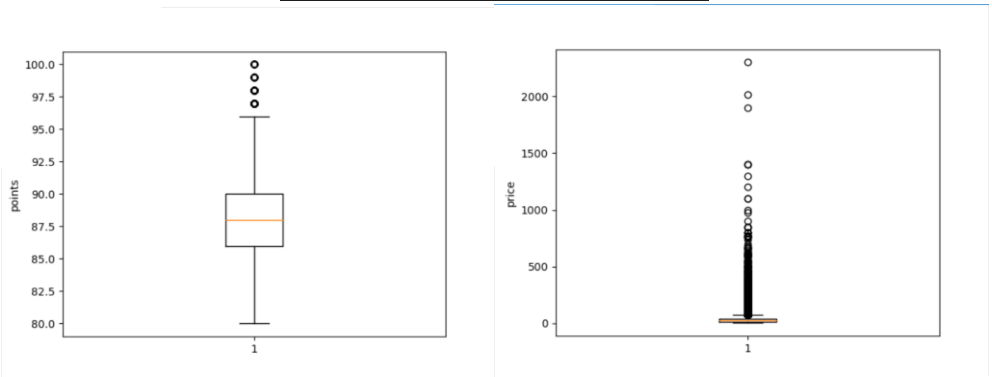
绘制 QQ 图检验数据分布是否为正态分布：

```
# qq图
points = Ndf["points"]
price = Ndf["price"]
sort_points = np.sort(points)
sort_price = np.sort(price)
y_points = np.arange(len(sort_points))/float(len(sort_points))
y_price = np.arange(len(sort_price))/float(len(sort_price))
trans_y_points = stats.norm.ppf(y_points)
trans_y_price = stats.norm.ppf(y_price)
plt.scatter(sort_points, trans_y_points)
plt.xlabel("Ordered values")
plt.ylabel("Theoretical quantile")
plt.title("points--quantile-quantile plot")
plt.show()
plt.scatter(sort_price, trans_y_price)
plt.xlabel("Ordered values")
plt.ylabel("Theoretical quantile")
plt.title("points--quantile-quantile plot")
plt.show()
```



绘制盒图，对离群值进行识别：

```
# 盒图
plt.boxplot(Ndf["points"])
plt.ylabel("points")
plt.show()
plt.boxplot(Ndf["price"])
plt.ylabel("price")
plt.show()
```



二．数据缺失的处理（**以数据集 Wine Reviews 为例**）

文件 1：winemag-data_first150k.csv

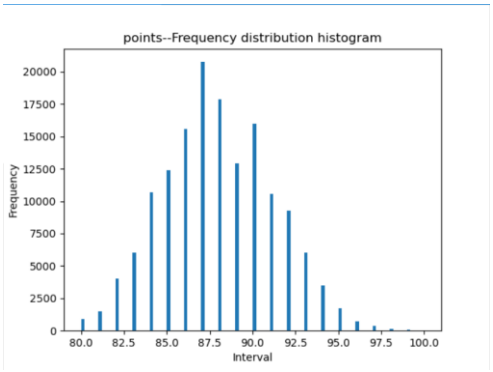
1. 将缺失部分剔除

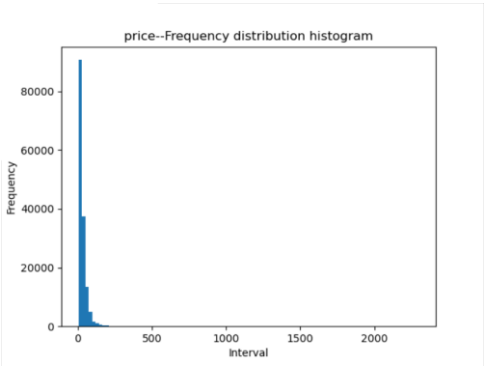
   上文中使用的过程就是剔除缺失值的方法，将空值皆去除掉。

2. 用最高频率值来填补缺失值

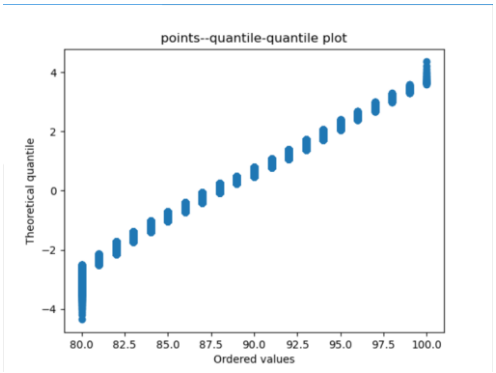
```
# 用最高频率值来填补缺失值
def high_feq_process(df1):
    Ndf = pd.DataFrame(df1, columns=["points", "price"])

    # 用value_counts()方法计算指定数值属性不同取值的频率
    feq_points = Ndf["points"].value_counts()
    feq_price = Ndf["price"].value_counts()

    # 通过下标得到频率最高的取值
    fill_value = {
        "points": list(dict(feq_points))[0],
        "price": list(dict(feq_price))[0]
    }
    print(fill_value)
    # 用fillna()方法填补对应列的缺失值，并调用之前的可视化函数
    Ndf = Ndf.fillna(value=fill_value)
    print(Ndf)
    Numeric(Ndf)
```

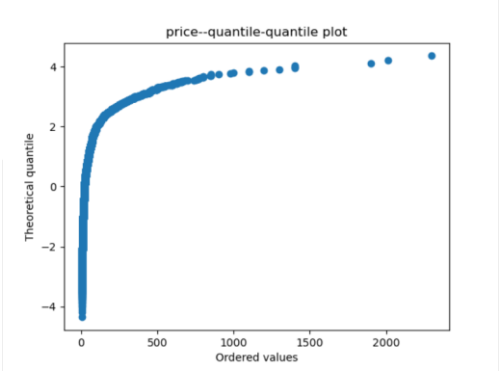
直方图：



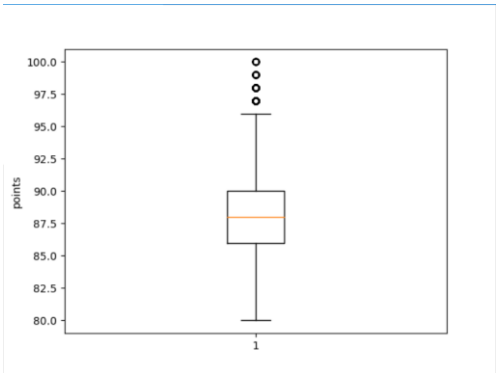


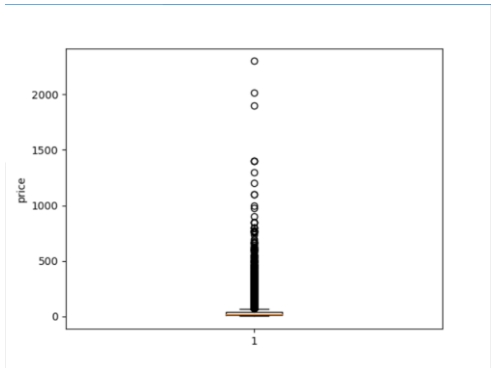
QQ 图





盒图：

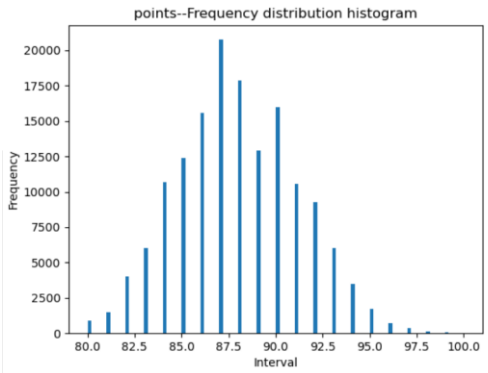


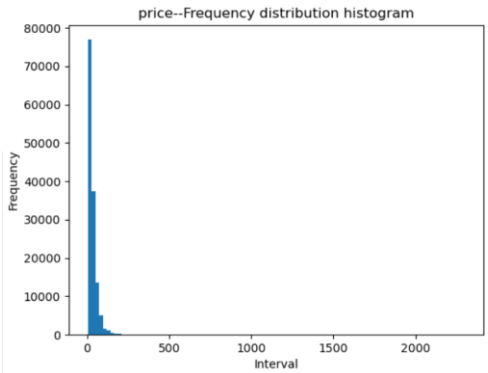


3. 通过属性的相关关系来填补缺失值

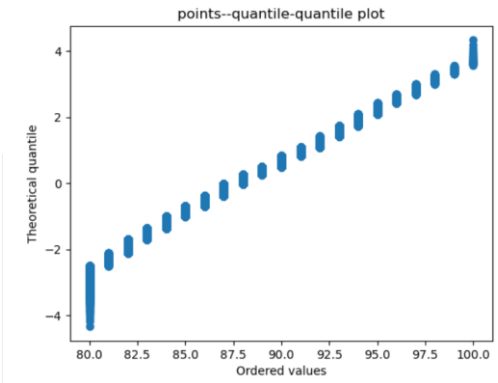
```
# 通过属性的相关关系来填补缺失值
def relation_process(df1):
    Ndf = pd.DataFrame(df1, columns=["points", "price"])
    Ndf.interpolate(method="values")
    Numeric(Ndf)
```

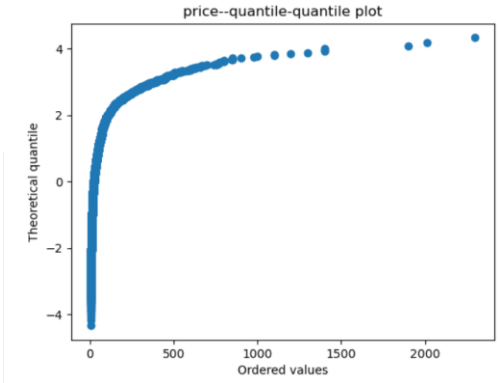
直方图：



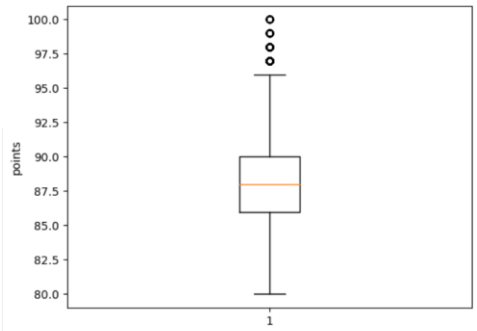


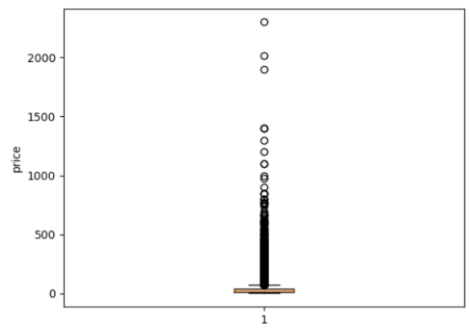
QQ 图：





盒图：

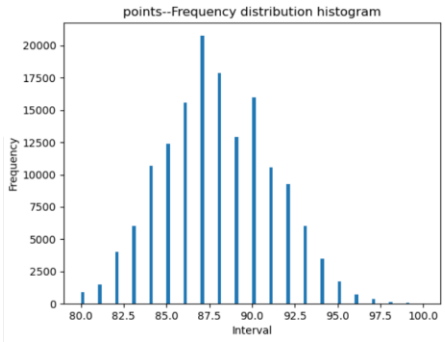


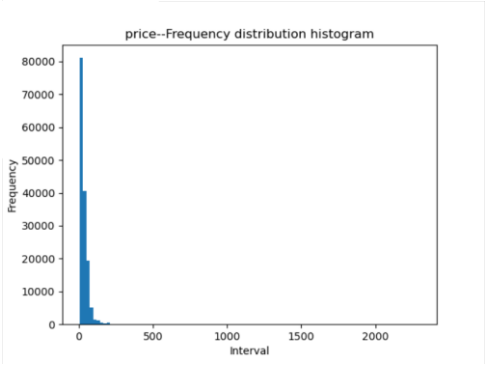


4. 通过数据对象之间的相似性来填补缺失值

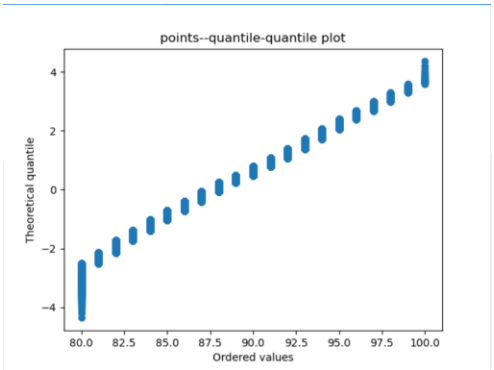
```
# 通过数据对象之间的相似性来填补缺失值
def similarity_process(df1, k_num):
    OriNdf = pd.DataFrame(df1, columns=["points", "price"])
    Ndf = pd.DataFrame(df1, columns=["points", "price"])
    Ndf = Ndf.dropna(axis=0, how="any")
    clf = KNeighborsRegressor(n_neighbors=k_num, weights="distance")
    clf.fit(np.array(list(Ndf["points"])).reshape(-1, 1), np.array(list(Ndf["price"])).reshape(-1, 1))

    for i in range(0, len(OriNdf)):
        if pd.isna(OriNdf.iloc[i]["price"]):
            new_value = clf.predict(np.array([OriNdf.iloc[i]["points"]]).reshape(-1, 1))
            OriNdf.set_value(i, "price", new_value)
    Numeric(OriNdf)
```

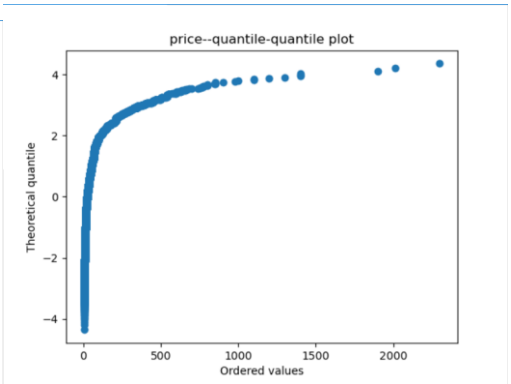
直方图：



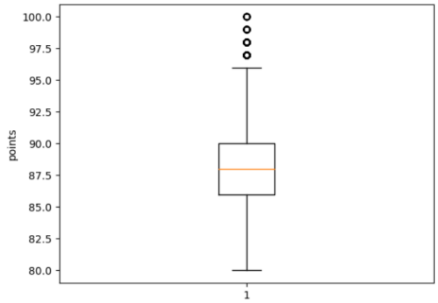


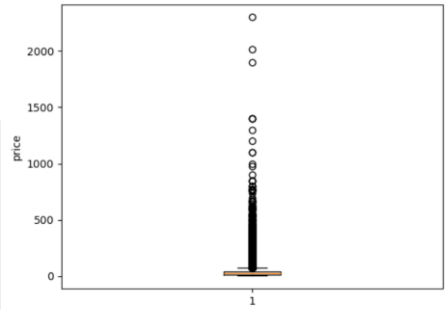
QQ 图：





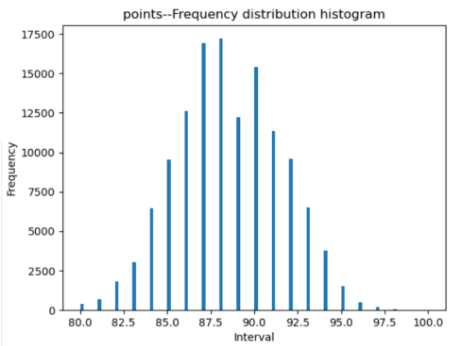
盒图：

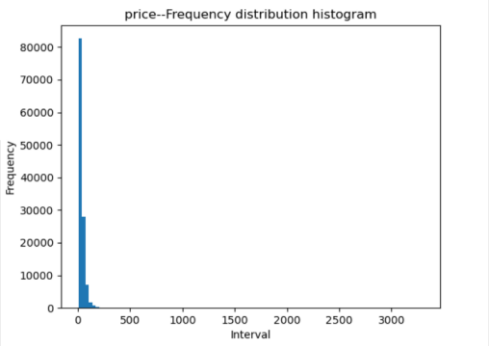


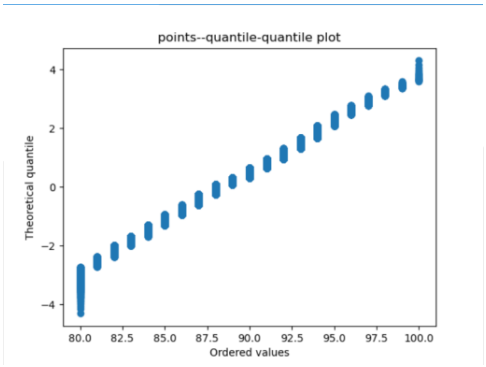


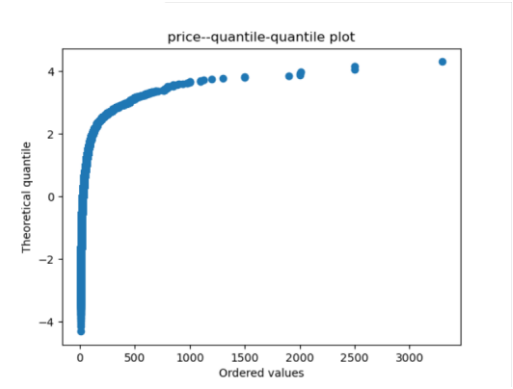
文件 2：winemag-data-130k-v2.csv

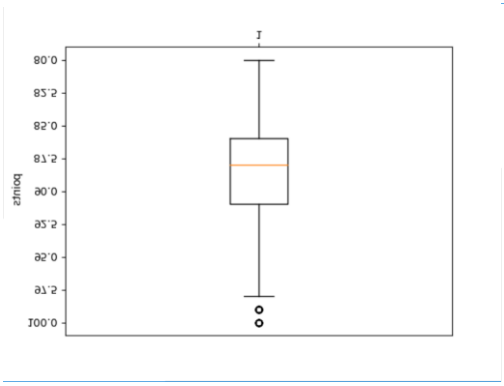
1. 原始数据（将缺失部分剔除）：

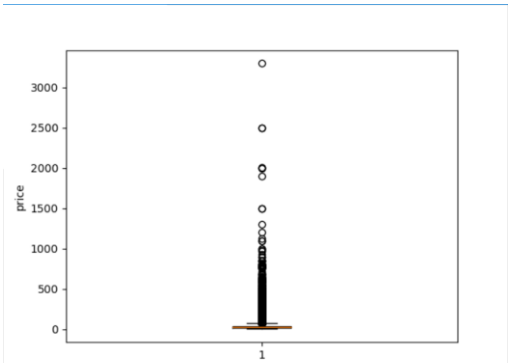




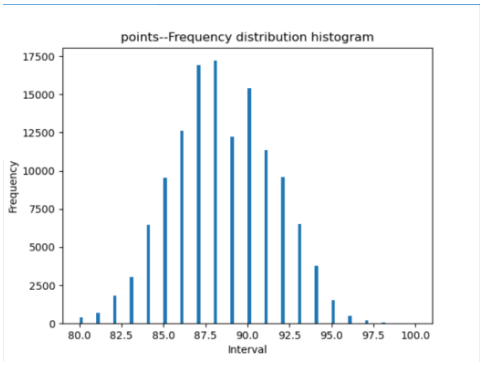


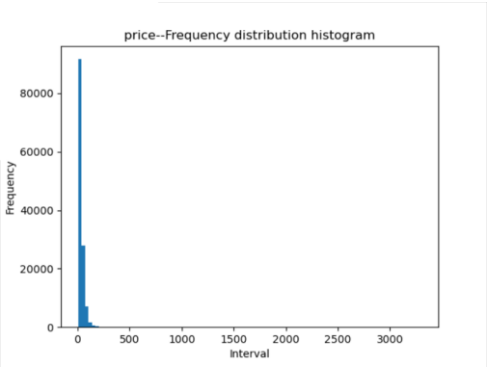


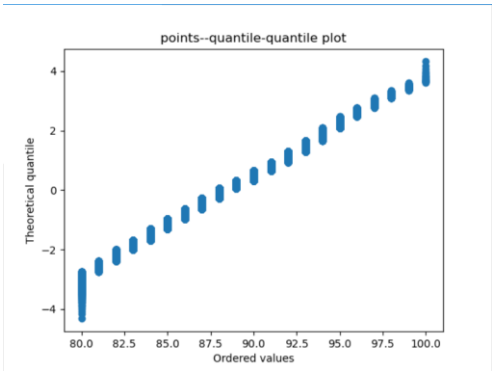


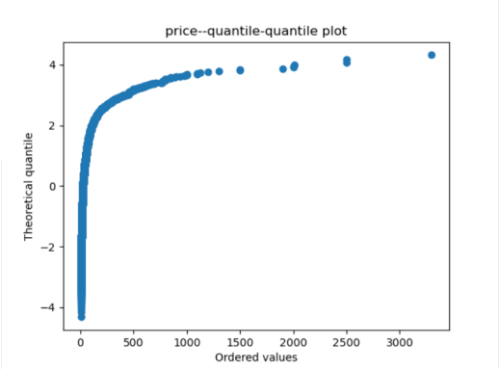


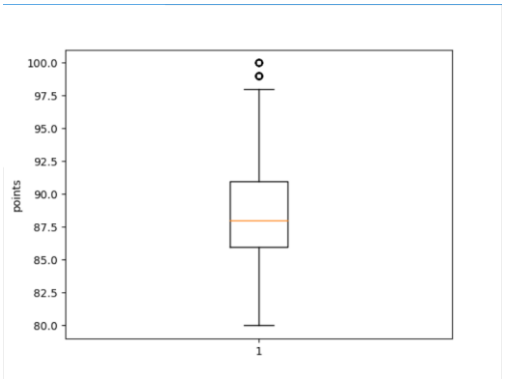
2. 用最高频率值来填补缺失值

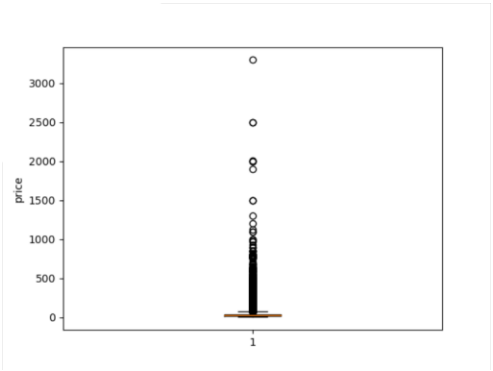




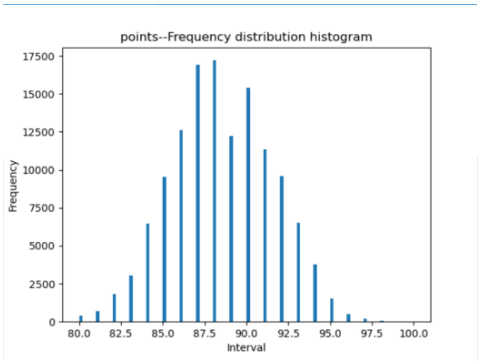


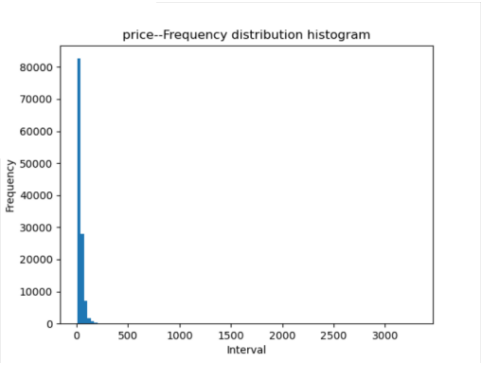


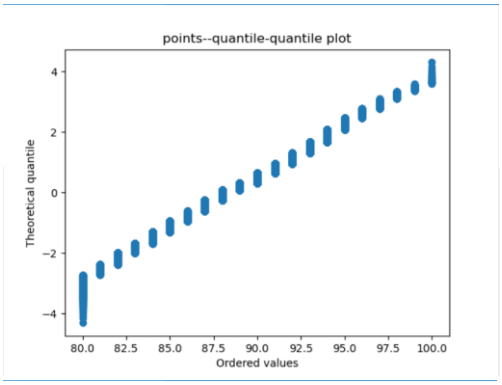


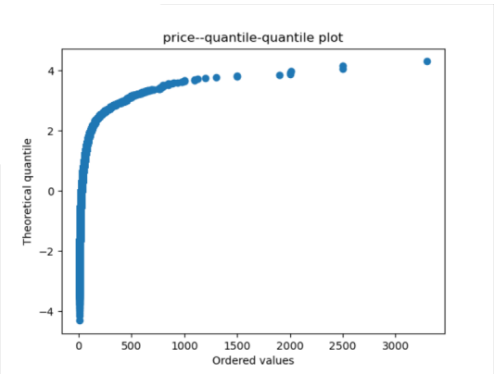


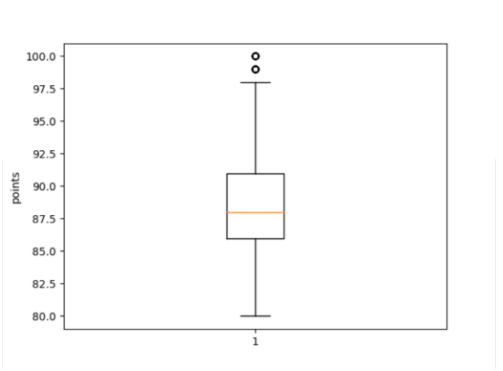
3. 通过属性的相关关系来填补缺失值：

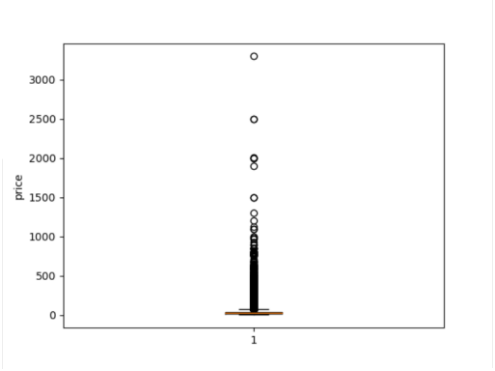












4. 通过数据对象之间的相似性来填补缺失值：