数据挖掘大作业一：数据探索性分析与数据预处理

1 数据说明

1.1 数据集

选择数据集 1：Wine Reviews；数据集 2：Oakland Crime Statistics 2011 to 2016

1.2 数据集描述

数据集 1：Wine Reviews
winemag-data-130k-v2.csv 包含 13 列和 130k 行葡萄酒评论。
属性列表：
country：葡萄酒来自的国家
description
designation：酿酒厂内的葡萄园，酿造葡萄酒的葡萄来自葡萄园
points：WineEnthusiast 对葡萄酒的评分为 1-100（尽管他们说他们只对评分> = 80 的葡萄酒发表评论）
price：一瓶葡萄酒的成本
province：葡萄酒来自的省或州
region_1：省或州的葡萄种植区（即纳帕）
region_2：有时在葡萄酒种植区域（即纳帕谷内的卢瑟福）中指定了更多特定区域，但此值有时可能为空白
taster_name
taster_twitter_handle
title：葡萄酒评论的标题，如果您对提取该功能感兴趣，通常会包含葡萄酒
variety：用于酿造葡萄酒的葡萄种类（即黑比诺）
winery：制作葡萄酒的酿酒厂
选择有意义的属性，其中，属于标称属性的有：country，designation，province，region_1，region_2, taster_name, taster_twitter_handle, variety, winery; 数值属性的有: 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。
选择有意义的属性，其中，属于标称属性的有：country，designation，province，region_1，region_2，variety，winery；数值属性的有：points，price。

数据集 2：Oakland Crime Statistics 2011 to 2016
包含 records-for-2011.csv 到 records-for-2016.csv 共六个文件

属性列表：
Agency，Create Time，Location，Area Id，Beat，Priority，Incident Type Id，Incident Type，Description，Event Number，Closed Time

2 数据分析

以数据集 1 为例

2.1 数据摘要

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)
```

对标称属性，给出每个可能取值的频数。
首先按列分别取出各标称属性，

```
col1 = df1.country.unique()      #country
col6 = df1.province.unique()      #province
col7 = df1.region_1.unique()      #region_1
col8 = df1.region_2.unique()      #region_2
```

然后通过 list 类型统计频数

```
for each in col1:
    resdic["col1"][each] = list(df1.country).count(each)
```

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。
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

```
Ndf = pd.DataFrame(df1, columns=["points", "price"])
describe = Ndf.describe()
```

指标如下，然后将最大、最小、均值、中位数、四分位数按序取出并存储

	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

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"]["Nan"] = Ndf.points.isna().sum()
```

2.2 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。
首先对属性列进行去空值处理

```
#去空值
Ndf = Ndf.dropna(subset=["price"])
```

然后使用 Matplotlib 包中的 hist()方法绘制直方图

```
plt.hist(Ndf["price"])
plt.xlabel("Interval")
plt.ylabel("Frequency")
plt.title("price--Frequency distribution histogram")
plt.show()
```

QQ 图全称 Quantile Quantile 图，是两种分布的分位数相对彼此进行绘制的图。首先对数值属性值排序，然后对序数的目标累计分布函数值求标准正太分布累计分布函数的逆，然后再绘制一个二维点分布图，得到直线则可证明是正态分布，反之不是。

```
#qq图
points = df1["points"]
price = df1["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

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

2.3 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:

2.3.1 将缺失部分剔除

上文中的数据分析过程使用的即为剔除缺失值方法

2.3.2 用最高频率值来填补缺失值

使用 Value_count()方法计算得到要计算的数值属性列不同取值的频率，

```
feq_points = Ndf["points"].value_counts()
```

然后使用字典及列表转换在通过下标或得到频率最高的取值。

```
fill_value = {
    "points": list(dict(feq_points))[0],
    "price": list(dict(feq_price))[0]
}
```

使用 fillna()方法填补对应列的缺失值并调用之前的可视化函数，来可视化填补缺失值后的结果。

```
Ndf = Ndf.fillna(value=fill_value, inplace = True)  #without inplace = True, the value in source df won't change
Numeric(Ndf)
```

2.3.3 通过属性的相关关系来填补缺失值

插值法就是一个从已知点近似计算未知点的近似计算方法，即构造一个多项式函数，使其通过所有已知点，然后用求得的函数预测位置点。pandas 内置有 interpolate()方法进行插值操作，该方法默认为线性插值即 method=linear。除此之外，还有{'linear', 'time', 'index', 'values'}等方法。这里选择简单的 value 方法。

```
Ndf = pd.DataFrame(df1, columns=["points", "price"])
Ndf.interpolate(method="values")
Numeric(Ndf)
```

2.3.4 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的基本思想在于：使用聚类的方法，通过对其他数值属性对每一条数据进行聚类，然后对于包含缺失值的行，通过其他不缺少的属性列来确定这一行所属的聚类，然后根据这一类的平均属性对这一行的缺失值进行回归。
首先训练聚类模型，先将数据集中的缺失值去掉构建干净的训练数据集。

```
Ndf = Ndf.dropna(axis=0, how="any")
```

通过 sklearn 包中的 KNeighborsRegressor 初始化一个聚类（用于回归），然后喂给完整的两列数值属性列进行聚类的训练。

```
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))  #reshape(1, -1) rather than reshape(-1, 1)
```

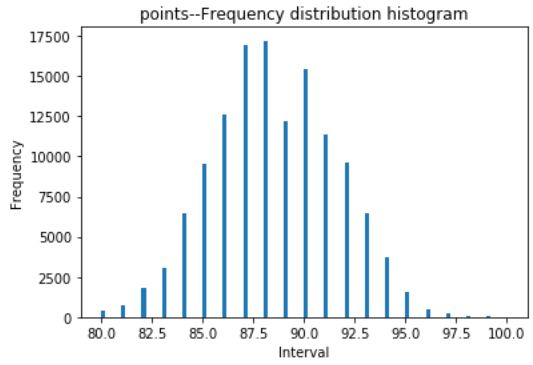
然后循环遍历数据中的每一行， 对 price 列中包含缺失值的行，使用 points 列的值进行回归计算，并将值赋给对应的 price。

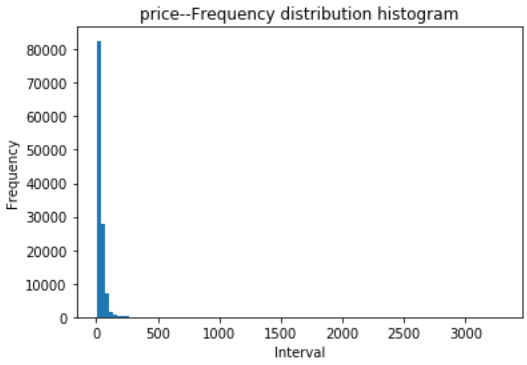
```
for i in range(0, len(OriNdf)):
    if pd.isna(OriNdf.iloc[i]["price"]):
        new_value = clf.predict(np.array([OriNdf.iloc[i]["points"]]).reshape(-1, 1))
        OriNdf.set_value(i, "price", new_value)
Numeric(OriNdf)
```

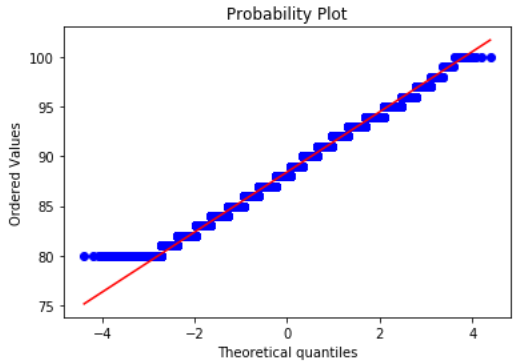
填补完缺失值后再对数据做可视化。

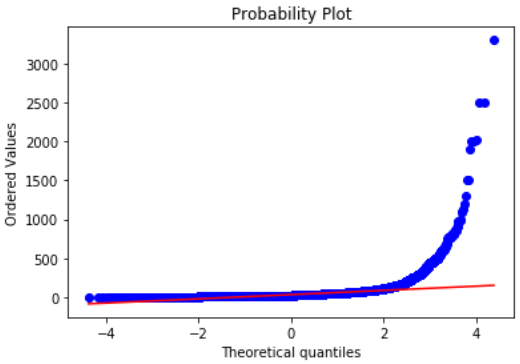
2.3.5 可视化地对比新旧数据集

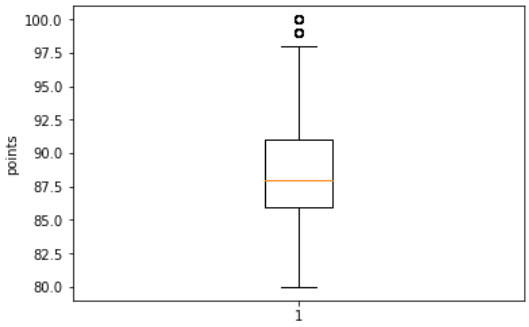
原数据:

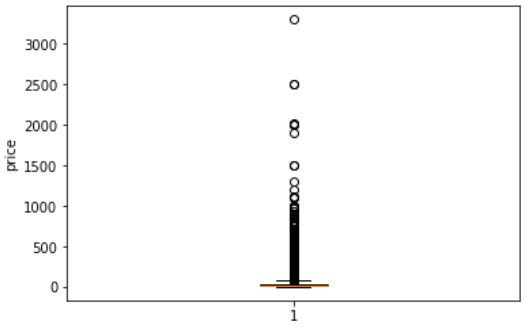




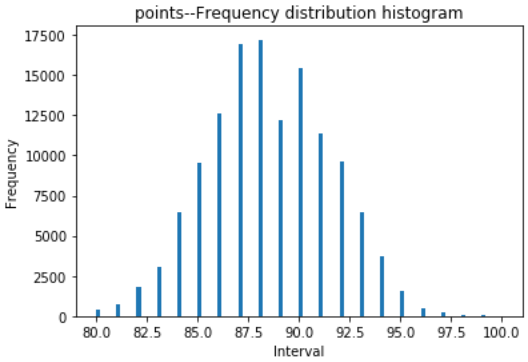


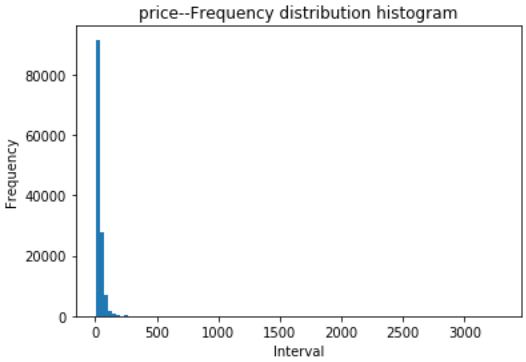


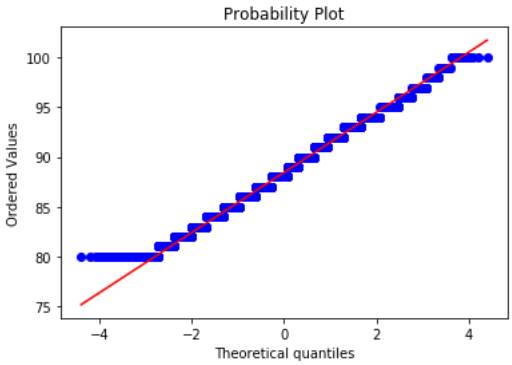


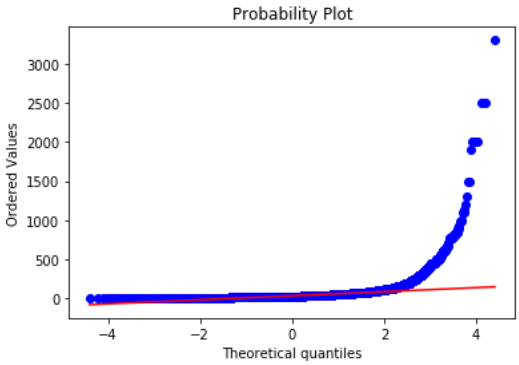


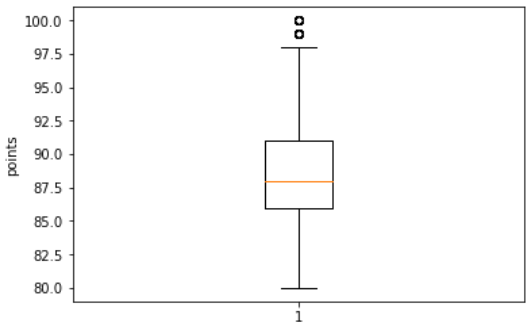
通过最高频取值填补:

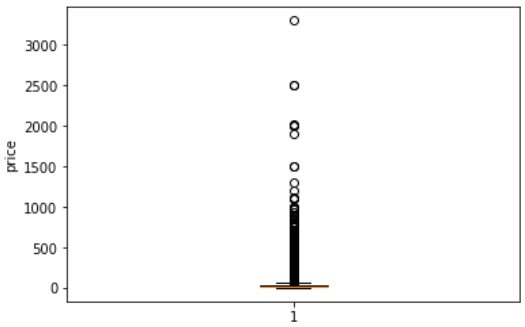




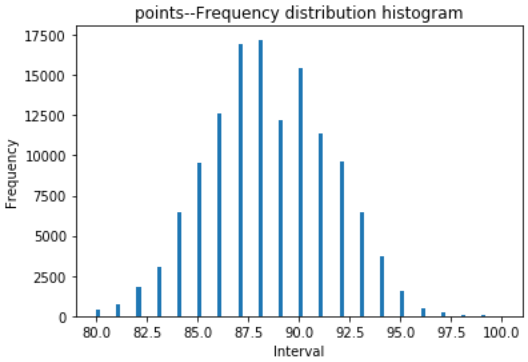


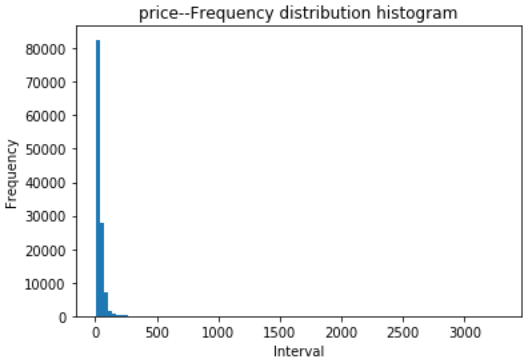


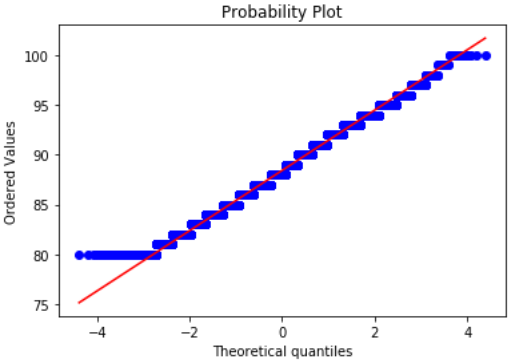


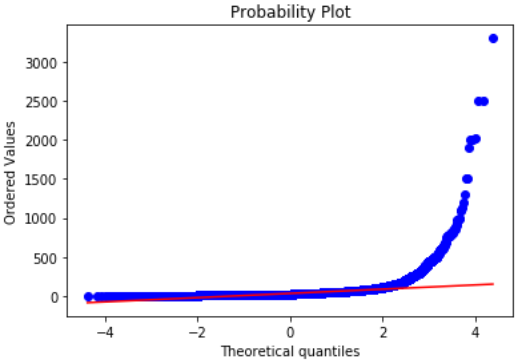


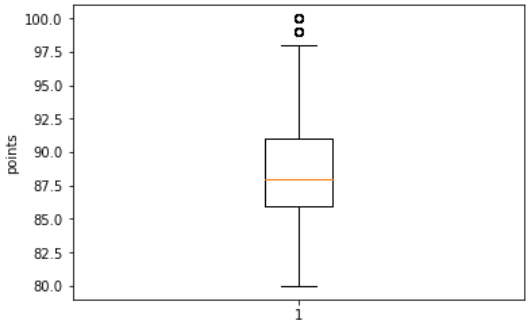
通过属性相关性填补:

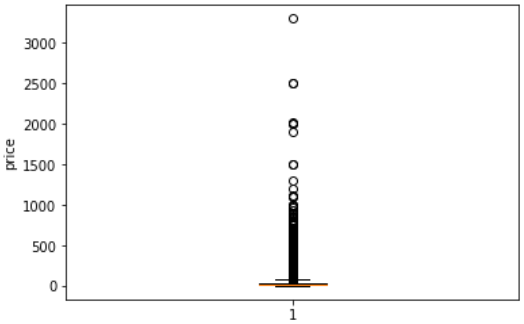

points--Frequency distribution histogram
Frequency
Interval
17500
15000
12500
10000
7500
5000
2500
0
80.0
82.5
85.0
87.5
90.0
92.5
95.0
97.5
100.0


price--Frequency distribution histogram
Frequency
Interval
80000
70000
60000
50000
40000
30000
20000
10000
0
0
500
1000
1500
2000
2500
3000


Probability Plot
Ordered Values
Theoretical quantiles
100
95
90
85
80
75
−4
−2
0
2
4


Probability Plot
Ordered Values
Theoretical quantiles
3000
2500
2000
1500
1000
500
0
−4
−2
0
2
4


points
100.0
97.5
95.0
92.5
90.0
87.5
85.0
82.5
80.0
1


price
3000
2500
2000
1500
1000
500
0
1