互评作业1: 数据探索性分析与数据预处理

1. 问题描述

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。

2. 数据集

可选数据集包括：

Consumer & Visitor Insights For Neighborhoods

Wine Reviews

Oakland Crime Statistics 2011 to 2016

Chicago Building Violations

Trending YouTube Video Statistics

Melbourne Airbnb Open Data

MLB Pitch Data 2015-2018

3. 数据分析要求

3.1 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5数概括及缺失值的个数

数据可视化

  使用直方图、盒图等检查数据分布及离群点

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

注意：在处理后，要可视化地对比新旧数据集。

4. 提交内容

分析过程报告（PDF格式）

程序所在代码仓库地址（建议使用Github或码云），仓库中应包含完整的处理数据的程序和使用说明

所选择的数据集应在仓库的README文件中说明

相关的数据文件不要上传到代码仓库中

建议：使用Jupyter Notebook将分析报告和代码组织在一起，使用Notebook的导出功能将报告导出为PDF格式的文件上传到乐学。

```
# coding:utf-8
import matplotlib
matplotlib.use('Agg')

import numpy as np
import scipy as sp
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import json
import math
import re
import sys
import csv
import random

'''
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5数概括及缺失值的个数

# 给出数据的基本信息，判断标称属性/数值属性
def columns_info(df,show_shape=True,new_df=None):
    print(df.info())
    if show_shape: print(df.shape)
    if new_df is not None:
        print()
        print(new_df.info())
        if show_shape: print(new_df.shape)
    return None

# 标称属性每个可能取值的频数及直方图
def nominal_summary(df,nominal_index=None,head_n=50,new_df=None):
    # 根据频数绘制直方图
    def bar_describe(data,new_data=None,head_n=50):
        if new_data is None:
            plt.figure(figsize=(24,8))
            plt.title(data.name, fontsize=30)
            plt.bar(data.index[:head_n], data.values[:head_n])
            plt.xticks(rotation=90)
            plt.show()
            fig_name = "./figs/niminal{}.jpg".format(random.randint(0,9999))
            plt.savefig(fig_name)
            print(fig_name)
        else:
            plt.figure(figsize=(24,8))
            plt.title(data.name, fontsize=30)
            plt.bar(data.index[:head_n],
data.values[:head_n],color='b',label=data.name)


 plt.bar(new_data.index[:head_n],new_data.values[:head_n],color='r',label='new_'+new_da
ta.name)
            plt.xticks(rotation=90)
            plt.legend()
            plt.show()
            fig_name = "./figs/nominal{}.jpg".format(random.randint(0,9999))
            plt.savefig(fig_name)
            print(fig_name)

    # 获取数据中每个取值的频数
    frequency = {key: df[key].value_counts() for key in df.columns}
    new_frequency = None
    if new_df is not None:
        new_frequency = {key: new_df[key].value_counts() for key in new_df.columns}

    if nominal_index is None:
        nominal_index=df.columns
    if new_frequency is None:
        for key in nominal_index:
            bar_describe(data=frequency[key],new_data=None,head_n=head_n)
    else:
        for key in nominal_index:
            bar_describe(data=frequency[key],new_data=new_frequency[key],head_n=head_n)
    return None

# 数值属性缺失值个数及五数概括
def numerical_summary(df,numerical_index=None,new_df=None):
    shape=df.shape # 样本总数
    # 格式化输出五数概括和缺失值个数
    def data_describe(data,new_data=None):
        print('descriptive statistics (%s):' % data.name)
        info=data.describe()
        new_info = None
        if new_data is not None:
            new_info = new_data.describe()

        print("Min: {:.4f}\tQ1(25%): {:.4f} \tQ2(50%): {:.4f} \tQ3(75%): {:.4f} \tMax:
{:.4f}".format(
            info['min'],info['25%'],info['50%'],info['75%'],info['max']))
        print("Missing: {:d}".format(int(shape[0] - info['count'])))
        print()

        if new_info is not None:
            print("\033[95m new Min: {:.4f}\tQ1(25%): {:.4f} \tQ2(50%): {:.4f}
\tQ3(75%): {:.4f} \tMax: {:.4f} \033[0m".format(

 new_info['min'],new_info['25%'],new_info['50%'],new_info['75%'],new_info['max']))
            print("\033[95m new Missing: {:d} \033[0m".format(int(shape[0] -
new_info['count'])))
            print()

    if numerical_index is None:
        print("Please provide the numerical index needed to be describe")
        return None

    # 获取数值属性的5数概况和缺失值个数
    if new_df is None:
        for key in numerical_index:
            data_describe(df[key])
    else:
        for key in numerical_index:
            data_describe(df[key],new_df[key])

    return None

# 数据可视化
#      使用直方图、盒图等检查数据分布及离群点

# 数据可视化，用于绘制数值属性的直方图，盒图及散点图
def box_plot(df,ylabel=None,new_df=None):
    if new_df is None:
        plt.figure()
        plt.title('Boxplot')
        sns.boxplot(y=ylabel,data=df,palette='Set2')
        fig_name = "./figs/Box{}.jpg".format(random.randint(0,9999))
        plt.savefig(fig_name)
        print(fig_name)
    else:
        plt.figure(figsize=(14,8))
        plt.subplot(121)
        plt.title('Boxplot')
        sns.boxplot(y=ylabel,data=df,palette='Set1')
        plt.subplot(122)
        plt.title('new_Boxplot')
        sns.boxplot(y=ylabel,data=new_df,palette='Set2')
        fig_name = "./figs/BiBox{}.jpg".format(random.randint(0,9999))
        plt.savefig(fig_name)
        print(fig_name)
```

```python
def hist_plot(df,ylabel=None,bins=10,new_df=None):
    if ylabel is None:
        print("Please provide ylabel")
        return None
    if new_df is None:
        plt.figure()
        plt.title('Histogram')
        sns.distplot(a=df[ylabel].dropna(),bins=bins,hist=True,kde=False)
        fig_name = "./figs/Hist{}.jpg".format(random.randint(0,9999))
        plt.savefig(fig_name)
        print(fig_name)
    else:
        plt.figure(figsize=(14,8))
        plt.subplot(121)
        plt.title('Histogram')
        sns.distplot(a=df[ylabel].dropna(),bins=bins,hist=True,kde=False)

        plt.subplot(122)
        plt.title('new_Histogram')
        sns.distplot(a=new_df[ylabel].dropna(),bins=bins,hist=True,kde=False)
        fig_name = "./figs/BiHist{}.jpg".format(random.randint(0,9999))
        plt.savefig(fig_name)
        print(fig_name)

def scatter_plot(df,xlabel=None,ylabel=None,new_df=None):
    if xlabel is None or ylabel is None:
        print("Please provide xlabel and ylabel")
        return None
    if new_df is None:
        plt.figure()
        plt.title('Scatter plot')
        sns.scatterplot(x=xlabel,y=ylabel,data=df)
        fig_name = "./figs/Scat{}.jpg".format(random.randint(0,9999))
        plt.savefig(fig_name)
        print(fig_name)
    else:
        plt.figure(figsize=(14,8))
        plt.subplot(121)
        plt.title('Scatter plot')
        sns.scatterplot(x=xlabel,y=ylabel,data=df)

        plt.subplot(122)
        plt.title('new Scatter plot')
        sns.scatterplot(x=xlabel,y=ylabel,data=new_df)
        fig_name = "./figs/BiScat{}.jpg".format(random.randint(0,9999))
        plt.savefig(fig_name)
        print(fig_name)

def viz_pairs(df,ylabel=None,xlabel=None,bins=20,new_df=None):
    if ylabel is None or xlabel is None:
        print("Please provide xlabel and ylabel")
        return None
    box_plot(df,ylabel=ylabel,new_df=new_df)
    box_plot(df,ylabel=xlabel,new_df=new_df)
    hist_plot(df,ylabel=ylabel,bins=bins,new_df=new_df)
    hist_plot(df,ylabel=xlabel,bins=bins,new_df=new_df)
    scatter_plot(df,xlabel=xlabel,ylabel=ylabel,new_df=new_df)

'''
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
'''
#将缺失部分剔除
def miss_drop(df):
    return df.dropna(axis=0,inplace=False)

#用最高频率值来填补缺失值
def miss_mode_fill(df):
    cpdf = df.copy(deep=True)
    for key in cpdf.columns:
        cpdf[key].fillna(cpdf[key].mode()[0],inplace=True)
    return cpdf

#通过属性的相关关系来填补缺失值
from sklearn.preprocessing import OrdinalEncoder
from sklearn.ensemble import RandomForestRegressor
def attr_corr_fill(df,miss_index,complete_index):
  def set_miss_values(df, complete_index):
    enc_label = OrdinalEncoder()
    enc_fea = OrdinalEncoder()
    missing_index = complete_index[0]

    # Take out the existing numerical data (no NaN) and throw them in Random Forest
Regressor
    train_df = df[complete_index]
    # known & unknow values
    known_values = np.array(train_df[train_df[missing_index].notnull()])
    unknow_values = np.array(train_df[train_df[missing_index].isnull()])

    # y is the know missing_index
    y = known_values[:, 0].reshape(-1, 1)
    enc_label.fit(y)
    y = enc_label.transform(y)
    # X are the features
    X = known_values[:, 1:]
    test_X = unknow_values[:, 1:]
    all_X = np.row_stack((X,test_X))
    enc_fea.fit(all_X)
    X = enc_fea.transform(X)

    # fit
    rfr = RandomForestRegressor(random_state=0, n_estimators=2000, n_jobs=-1)
    rfr.fit(X, y.ravel())
    # predict
    predicted_values = rfr.predict(enc_fea.transform(unknow_values[:, 1:]))
    predicted_values = enc_label.inverse_transform(predicted_values.reshape(-1, 1))
    # fill in with predicted values
    df.loc[ (df[missing_index].isnull()), missing_index] = predicted_values
    return df

  cpdf = df.copy(deep=True)
  for i in range(0,len(miss_index)):
    complete_index.insert(0,miss_index[i])
    cpdf = set_miss_values(cpdf,complete_index)
  return cpdf


from impyute.imputation.cs import fast_knn
#通过数据对象之间的相似性来填补缺失值
def obj_corr_fill(df,numerical_index,k=30):
  sys.setrecursionlimit(100000) #Increase the recursion limit of the OS
  # start the KNN training
  cpdf = df.copy(deep=True)
  imputed_training=fast_knn(cpdf[numerical_index].values, k=k)
  imputed_training=pd.DataFrame(data=imputed_training,columns=numerical_index)
  cpdf[numerical_index] = imputed_training[numerical_index]
  return cpdf

# Wine Reviews
def wineReviews(csv_name):
    wine_reviews_dir = "/home/LiuBJ/dm/Wine Reviews"
    winemag =
pd.read_csv(wine_reviews_dir+'/'+csv_name,sep=',',header='infer',index_col=0)
    print('[0] 数据摘要&可视化')
    columns_info(winemag)
    # Nominal index
    nominal_index = ['country', 'designation', 'province', 'region_1', 'region_2',
'variety', 'winery']
    # Numerical index
    numerical_index = ['points', 'price']
    nominal_summary(winemag,new_df=None,nominal_index=nominal_index,head_n=50)
    numerical_summary(winemag,new_df=None,numerical_index=numerical_index)
    viz_pairs(winemag,xlabel='points',ylabel='price',bins=20,new_df=None)
    # 将缺失部分剔除
    print('[1] 将缺失部分剔除')
    dw=miss_drop(winemag)
    columns_info(winemag,show_shape=True,new_df=dw)
    nominal_summary(winemag,nominal_index=nominal_index,head_n=50,new_df=dw)
    numerical_summary(winemag,numerical_index=numerical_index,new_df=dw)
    viz_pairs(winemag,xlabel='points',ylabel='price',bins=20,new_df=dw)
    # 用最高频率值填补缺失值
    print('[2] 用最高频率值填补缺失值')
    md=miss_mode_fill(winemag)
    columns_info(winemag,show_shape=True,new_df=md)
    nominal_summary(winemag,nominal_index=nominal_index,head_n=50,new_df=md)
    numerical_summary(winemag,numerical_index=numerical_index,new_df=md)
    viz_pairs(winemag,xlabel='points',ylabel='price',bins=20,new_df=md)
    # 通过属性的相关关系来填补缺失值
    print('[3] 通过属性的相关关系来填补缺失值')
    miss_index=['country','designation','price','province','region_1','region_2']
    comp_index=['description','points','variety','winery']
    att=attr_corr_fill(winemag,miss_index=miss_index,complete_index=comp_index)
    columns_info(winemag,show_shape=True,new_df=att)
    nominal_summary(winemag,nominal_index=nominal_index,head_n=50,new_df=att)
    numerical_summary(winemag,numerical_index=numerical_index,new_df=att)
    viz_pairs(winemag,xlabel='points',ylabel='price',bins=20,new_df=att)
    # 通过数据对象之间的相似性来填补缺失值
    print('[4] 通过数据对象之间的相似性来填补缺失值')
    obj=obj_corr_fill(winemag,numerical_index=numerical_index,k=10)
    columns_info(winemag,show_shape=True,new_df=obj)
    nominal_summary(winemag,nominal_index=nominal_index,head_n=50,new_df=obj)
    numerical_summary(winemag,numerical_index=numerical_index,new_df=obj)
    viz_pairs(winemag,xlabel='points',ylabel='price',bins=20,new_df=obj)

def oaklandCrimeStatistics(csv_name):
    records_dir = '/home/LiuBJ/dm/Oakland Crime Statistics 2011 to 2016'
    records_for
=pd.read_csv(records_dir+'/'+csv_name,sep=',',header='infer',index_col=None)
    print('[0] 数据摘要&可视化')
    columns_info(records_for)
    # Nominal index
    # In 'records-for-2013.csv', 'Location'->'Location '
    nominal_index = ['Agency', 'Create Time', 'Location', 'Beat', 'Incident Type Id',
'Incident Type Description', 'Closed Time']
    # Numerical index
    numerical_index = ['Area Id', 'Priority']
    nominal_summary(records_for,new_df=None,nominal_index=nominal_index,head_n=50)
    numerical_summary(records_for,new_df=None,numerical_index=numerical_index)
    viz_pairs(records_for,xlabel='Area Id',ylabel='Priority',bins=20,new_df=None)
    # 将缺失部分剔除
    print('[1] 将缺失部分剔除')
    dw=miss_drop(records_for)
    columns_info(records_for,show_shape=True,new_df=dw)
    nominal_summary(records_for,nominal_index=nominal_index,head_n=50,new_df=dw)
    numerical_summary(records_for,numerical_index=numerical_index,new_df=dw)
    viz_pairs(records_for,xlabel='Area Id',ylabel='Priority',bins=20,new_df=None)
    # 用最高频率值填补缺失值
    print('[2] 用最高频率值填补缺失值')
    md=miss_mode_fill(records_for)
    columns_info(records_for,show_shape=True,new_df=md)
    nominal_summary(records_for,nominal_index=nominal_index,head_n=50,new_df=md)
    numerical_summary(records_for,numerical_index=numerical_index,new_df=md)
    viz_pairs(records_for,xlabel='Area Id',ylabel='Priority',bins=20,new_df=None)
    # 通过属性的相关关系来填补缺失值
    print('[3] 通过属性的相关关系来填补缺失值')
    miss_index=[]
    comp_index=['Location', 'Area Id', 'Beat', 'Create Time', 'Closed Time',
'Priority', 'Incident Type Id']
    att=attr_corr_fill(records_for,miss_index=miss_index,complete_index=comp_index)
    columns_info(records_for,show_shape=True,new_df=att)
    nominal_summary(records_for,nominal_index=nominal_index,head_n=50,new_df=att)
    numerical_summary(records_for,numerical_index=numerical_index,new_df=att)
    viz_pairs(records_for,xlabel='Area Id',ylabel='Priority',bins=20,new_df=None)
    # 通过数据对象之间的相似性来填补缺失值
    print('[4] 通过数据对象之间的相似性来填补缺失值')
    obj=obj_corr_fill(records_for,numerical_index=numerical_index,k=10)
    columns_info(records_for,show_shape=True,new_df=obj)
    nominal_summary(records_for,nominal_index=nominal_index,head_n=50,new_df=obj)
    numerical_summary(records_for,numerical_index=numerical_index,new_df=obj)
    viz_pairs(records_for,xlabel='Area Id',ylabel='Priority',bins=20,new_df=None)

wineReviews('winemag-data_first150k.csv')
# wineReviews('winemag-data-130k-v2.csv')
# oaklandCrimeStatistics('records-for-2011.csv')
# oaklandCrimeStatistics('records-for-2012.csv')
# oaklandCrimeStatistics('records-for-2013.csv')
# oaklandCrimeStatistics('records-for-2014.csv')
# oaklandCrimeStatistics('records-for-2015.csv')
# oaklandCrimeStatistics('records-for-2016.csv')
```

效果演示

选择的数据集为Wine Reviews和Oakland Crime Statistics 2011 to 2016，代码的356～363分别是对两个数据集的csv文件进行预处理。

以Wine Reviews数据集中的winemag-data_first150k.csv为例进行演示。

数据摘要&可视化

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150930 entries, 0 to 150929
Data columns (total 10 columns):
country          150925 non-null object
description      150930 non-null object
designation      105195 non-null object
points           150930 non-null int64
price            137235 non-null float64
province         150925 non-null object
region_1         125870 non-null object
region_2         60953 non-null object
variety          150930 non-null object
winery           150930 non-null object
dtypes: float64(1), int64(1), object(8)
memory usage: 12.7+ MB
None
(150930, 10)
```

```
descriptive statistics (points):
Min: 80.0000     Q1(25%): 86.0000        Q2(50%): 88.0000        Q3(75%): 90.0000
Max: 100.0000
Missing: 0
()
descriptive statistics (price):
Min: 4.0000      Q1(25%): 16.0000        Q2(50%): 24.0000        Q3(75%): 40.0000
Max: 2300.0000
Missing: 13695
```

数据缺失的处理

将缺失部分剔除

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150930 entries, 0 to 150929
Data columns (total 10 columns):
country        150925 non-null object
description    150930 non-null object
designation    105195 non-null object
points         150930 non-null int64
price          137235 non-null float64
province       150925 non-null object
region_1       125870 non-null object
region_2       60953 non-null object
variety        150930 non-null object
winery         150930 non-null object
dtypes: float64(1), int64(1), object(8)
memory usage: 12.7+ MB
None
(150930, 10)
()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 39241 entries, 0 to 150916
Data columns (total 10 columns):
country        39241 non-null object
description    39241 non-null object
designation    39241 non-null object
points         39241 non-null int64
price          39241 non-null float64
province       39241 non-null object
region_1       39241 non-null object
region_2       39241 non-null object
variety        39241 non-null object
winery         39241 non-null object
dtypes: float64(1), int64(1), object(8)
memory usage: 3.3+ MB
None
(39241, 10)
```

```
descriptive statistics (points):
Min: 80.0000    Q1(25%): 86.0000        Q2(50%): 88.0000        Q3(75%): 90.0000
Max: 100.0000
Missing: 0
()
 new Min: 80.0000       Q1(25%): 86.0000        Q2(50%): 88.0000        Q3(75%):
91.0000        Max: 100.0000
 new Missing: 111689
()
descriptive statistics (price):
Min: 4.0000     Q1(25%): 16.0000        Q2(50%): 24.0000        Q3(75%): 40.0000
Max: 2300.0000
Missing: 13695
()
 new Min: 4.0000        Q1(25%): 22.0000        Q2(50%): 32.0000        Q3(75%):
45.0000        Max: 2013.0000
 new Missing: 111689
()
```

用最高频率值填补缺失值

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150930 entries, 0 to 150929
Data columns (total 10 columns):
country          150925 non-null object
description      150930 non-null object
designation      105195 non-null object
points           150930 non-null int64
price            137235 non-null float64
province         150925 non-null object
region_1         125870 non-null object
region_2         60953 non-null object
variety          150930 non-null object
winery           150930 non-null object
dtypes: float64(1), int64(1), object(8)
```

```
memory usage: 12.7+ MB
None
(150930, 10)
()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150930 entries, 0 to 150929
Data columns (total 10 columns):
country        150930 non-null object
description    150930 non-null object
designation    150930 non-null object
points         150930 non-null int64
price          150930 non-null float64
province       150930 non-null object
region_1       150930 non-null object
region_2       150930 non-null object
variety        150930 non-null object
winery         150930 non-null object
dtypes: float64(1), int64(1), object(8)
memory usage: 12.7+ MB
None
(150930, 10)
```

```
descriptive statistics (points):
Min: 80.0000    Q1(25%): 86.0000        Q2(50%): 88.0000        Q3(75%): 90.0000
Max: 100.0000
Missing: 0
()
 new Min: 80.0000       Q1(25%): 86.0000        Q2(50%): 88.0000        Q3(75%):
90.0000        Max: 100.0000
 new Missing: 0
()
descriptive statistics (price):
Min: 4.0000     Q1(25%): 16.0000        Q2(50%): 24.0000        Q3(75%): 40.0000
Max: 2300.0000
Missing: 13695
()
```

```
 new Min: 4.0000        Q1(25%): 16.0000        Q2(50%): 22.0000        Q3(75%):
38.0000        Max: 2300.0000
 new Missing: 0
()
```

通过属性的相关关系来填补缺失值

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150930 entries, 0 to 150929
Data columns (total 10 columns):
country        150925 non-null object
description    150930 non-null object
designation    105195 non-null object
points         150930 non-null int64
price          137235 non-null float64
province       150925 non-null object
region_1       125870 non-null object
region_2       60953 non-null object
variety        150930 non-null object
winery         150930 non-null object
dtypes: float64(1), int64(1), object(8)
memory usage: 12.7+ MB
None
(150930, 10)
()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150930 entries, 0 to 150929
Data columns (total 10 columns):
country        150930 non-null object
description    150930 non-null object
designation    150930 non-null object
points         150930 non-null int64
price          150930 non-null float64
province       150930 non-null object
region_1       150930 non-null object
region_2       150930 non-null object
variety        150930 non-null object
winery         150930 non-null object
dtypes: float64(1), int64(1), object(8)
memory usage: 12.7+ MB
None
```

```
(150930, 10)
```

```
descriptive statistics (points):
Min: 80.0000    Q1(25%): 86.0000        Q2(50%): 88.0000        Q3(75%): 90.0000
Max: 100.0000
Missing: 0
()
 new Min: 80.0000       Q1(25%): 86.0000        Q2(50%): 88.0000        Q3(75%):
90.0000        Max: 100.0000
 new Missing: 0
()
descriptive statistics (price):
Min: 4.0000     Q1(25%): 16.0000        Q2(50%): 24.0000        Q3(75%): 40.0000
Max: 2300.0000
Missing: 13695
()
 new Min: 4.0000        Q1(25%): 16.0000        Q2(50%): 25.0000        Q3(75%):
40.0000        Max: 2300.0000
 new Missing: 0
()
```

通过数据对象之间的相似性来填补缺失值

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150930 entries, 0 to 150929
Data columns (total 10 columns):
country        150925 non-null object
description    150930 non-null object
designation    105195 non-null object
```

```
points          150930 non-null int64
price           137235 non-null float64
province        150925 non-null object
region_1        125870 non-null object
region_2        60953 non-null object
variety         150930 non-null object
winery          150930 non-null object
dtypes: float64(1), int64(1), object(8)
memory usage: 12.7+ MB
None
(150930, 10)
()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150930 entries, 0 to 150929
Data columns (total 10 columns):
country         150925 non-null object
description     150930 non-null object
designation     105195 non-null object
points          150930 non-null float64
price           137310 non-null float64
province        150925 non-null object
region_1        125870 non-null object
region_2        60953 non-null object
variety         150930 non-null object
winery          150930 non-null object
dtypes: float64(2), object(8)
memory usage: 12.7+ MB
None
(150930, 10)
```

```
descriptive statistics (points):
Min: 80.0000    Q1(25%): 86.0000        Q2(50%): 88.0000        Q3(75%): 90.0000
Max: 100.0000
Missing: 0
()
 new Min: 80.0000       Q1(25%): 86.0000        Q2(50%): 88.0000        Q3(75%):
90.0000        Max: 100.0000
```

```
 new Missing: 0
()
descriptive statistics (price):
Min: 4.0000     Q1(25%): 16.0000        Q2(50%): 24.0000        Q3(75%): 40.0000
Max: 2300.0000
Missing: 13695
()
 new Min: 4.0000        Q1(25%): 16.0000        Q2(50%): 24.0000        Q3(75%):
40.0000        Max: 2300.0000
 new Missing: 13620
()
```