Analysis and Systems of Big Data Practise Lab - 3

Sreepathy Jayanand
CED17I038

Q1

1. Suppose that the data for analysis includes the attribute *age*. The *age* values for the data tuples are (in increasing order) 13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70.

(a) Use ***min-max normalization*** to transform the values of age to the range [0:1].
(b) Use ***z-score normalization*** to transform the values of *age.*
(c) Use normalization by ***decimal scaling*** to transform the values of *age* such that the transformed value is less than 1.

2. Use the given dataset and perform the operations listed below.

Logic : Using the formulas mentioned below, the new set of values is found out.

Min-Max Value:

age[i] = new_min + (age[i] - old_min) * (new_max - new_min) / (old_max - old_min)

Z-Score Value:

age[i] = (age[i] - Mean)/ Standard_Deviation

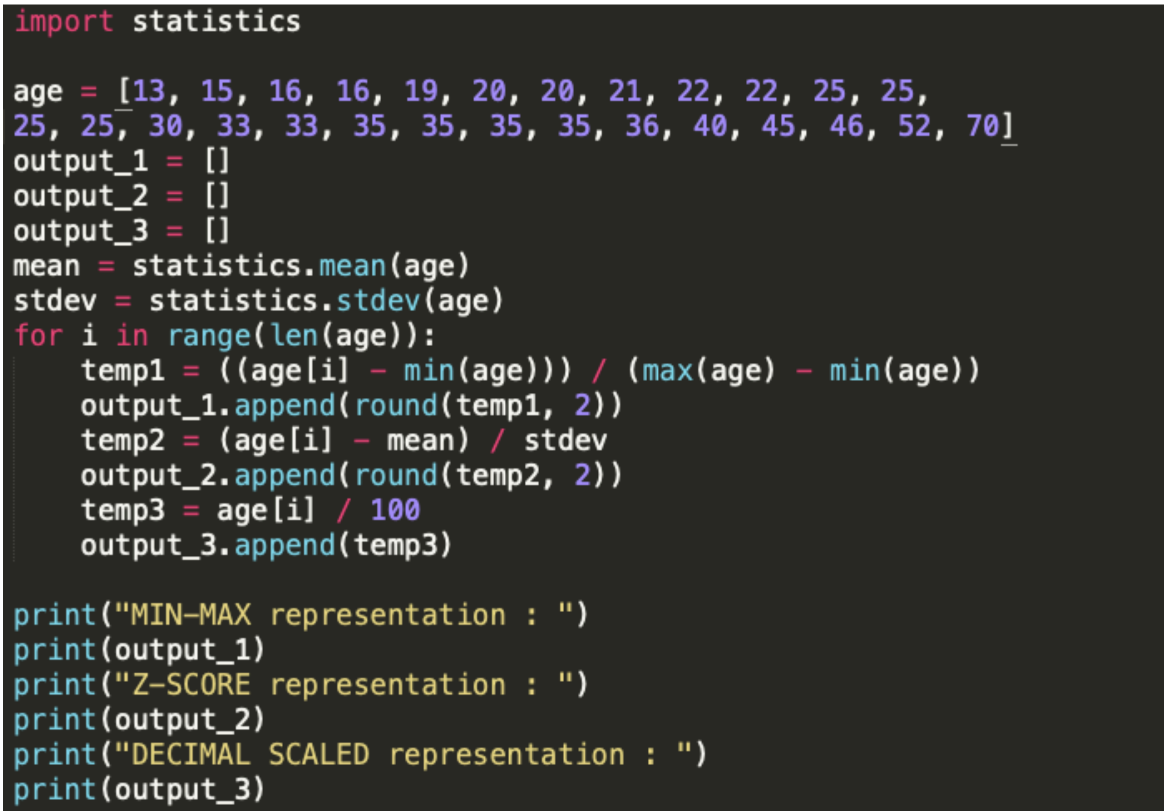
Decimal Scaling:

age[i] = age[i] / 100

Libraries used:

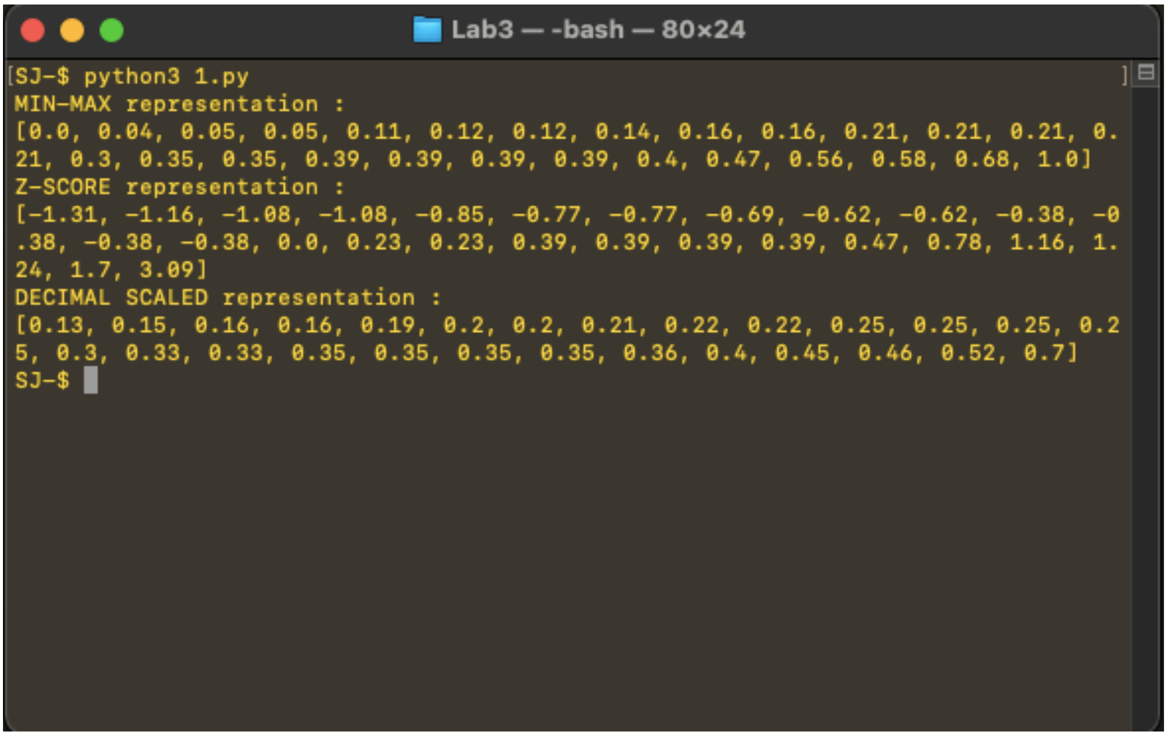
- Statistics - For mean and standard deviation

Code:



```
import statistics

age = [13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25,
25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70]
output_1 = []
output_2 = []
output_3 = []
mean = statistics.mean(age)
stdev = statistics.stdev(age)
for i in range(len(age)):
    temp1 = ((age[i] - min(age))) / (max(age) - min(age))
    output_1.append(round(temp1, 2))
    temp2 = (age[i] - mean) / stdev
    output_2.append(round(temp2, 2))
    temp3 = age[i] / 100
    output_3.append(temp3)

print("MIN-MAX representation : ")
print(output_1)
print("Z-SCORE representation : ")
print(output_2)
print("DECIMAL SCALED representation : ")
print(output_3)
```

Output:



```
[SJ-$ python3 1.py
MIN-MAX representation :
[0.0, 0.04, 0.05, 0.05, 0.11, 0.12, 0.12, 0.14, 0.16, 0.16, 0.21, 0.21, 0.21, 0.
21, 0.3, 0.35, 0.35, 0.39, 0.39, 0.39, 0.39, 0.4, 0.47, 0.56, 0.58, 0.68, 1.0]
Z-SCORE representation :
[-1.31, -1.16, -1.08, -1.08, -0.85, -0.77, -0.77, -0.69, -0.62, -0.62, -0.38, -0
.38, -0.38, -0.38, 0.0, 0.23, 0.23, 0.39, 0.39, 0.39, 0.39, 0.47, 0.78, 1.16, 1.
24, 1.7, 3.09]
DECIMAL SCALED representation :
[0.13, 0.15, 0.16, 0.16, 0.19, 0.2, 0.2, 0.21, 0.22, 0.22, 0.25, 0.25, 0.25, 0.2
5, 0.3, 0.33, 0.33, 0.35, 0.35, 0.35, 0.35, 0.36, 0.4, 0.45, 0.46, 0.52, 0.7]
SJ-$
```

Q2

1. Suppose that the data for analysis includes the attribute *age*. The *age* values for the data tuples are (in increasing order) 13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70.

From the Avocado Dataset,

Sort the attribute "Total Volume" in the given dataset and distribute the data into equal sized/frequency bins. Let the number of bins be 250. ***Smooth*** the sorted data by

(i) bin-means

(ii) bin-medians

(iii) bin-boundaries

b. The dataset represents weekly retail scan data for National retail volume (units) and price. However, the company is interested in knowing the *monthly (total per month) and annual sales (total per year)*, rather than the total per week. So, ***reduce*** the data accordingly.
c. Summarise the number of missing values for each attribute
d. Populate data for the ***missing values*** of the attribute= "Average Price" by averaging all the values of
the "Avg Price" attribute that fall under the same **"REGION**" attribute value.
e. ***Discretise*** the attribute= "Date" using *concept hierarchy* into {Old, New, Recent}

Logic : a) Different bins with size 250 is created. The data is smoothened by bin-means, bin medians, bin boundaries

b) The data is reduced by considering the total per month and year instead of week.
c) The columns with datatype float containing null value is found by converting the value to string datatype and removing the '.' , if the resultant string is not a digit then it is considered as null value. For columns with string datatype we are directly checking whether it is a missing value or no.

d) The missing values are replaced by average in that particular region.

e) The date in a data frame is replaced by old if the corresponding year is either 2015 or 2016, the date is replaced by new if the corresponding year is 2017 and the date is replaced by recent if the corresponding year is 2018.

Libraries used:

- Csv - To import csv files
- Math - For ceil function to compute number of bins
- Numpy - For array manipulation

Code:

```
import csv

import math
import numpy as np


with open("Lab_Data.csv", 'r') as f:
    data = list(csv.reader(f, delimiter = ","))

total_volume = []

for i in range(1, len(data)):
        total_volume.append(float(data[i][2]))

total_volume = np.array(total_volume)
total_volume = np.sort(total_volume)

no_of_bins = 250
bin_size = math.ceil(len(total_volume) / no_of_bins)

bin1 = np.zeros((no_of_bins, bin_size))
bin2 = np.zeros((no_of_bins, bin_size))
```

```
bin3 = np.zeros((no_of_bins, bin_size))

for i in range (0, no_of_bins*bin_size, bin_size):
    k = int(i/bin_size)
    mean = sum(total_volume[i:i+bin_size])/bin_size
    for j in range(bin_size):
        bin1[k,j] = round(mean, 6)
print("Bin Mean: \n",bin1)

for i in range (0, no_of_bins*bin_size, bin_size):
    k = int(i/bin_size)
    for j in range (bin_size):
        bin2[k,j] = total_volume[i+math.floor(bin_size/2)]
print("Bin Median: \n",bin2)

for i in range (0, no_of_bins*bin_size, bin_size):
    k = int(i/bin_size)
    for j in range (bin_size):
        if (total_volume[i+j]-total_volume[i]) < (total_volume[i+bin_size-1]-
total_volume[i+j]):
            bin3[k,j] = total_volume[i]
        else:
            bin3[k,j] = total_volume[i+bin_size-1]
print("Bin Boundaries: \n",bin3)



date = data[1][0][-7:]
region = data[1][12]
c = 0


list_date = []
list_avg_price = []
list_total_volume = []
list_4046 = []
```

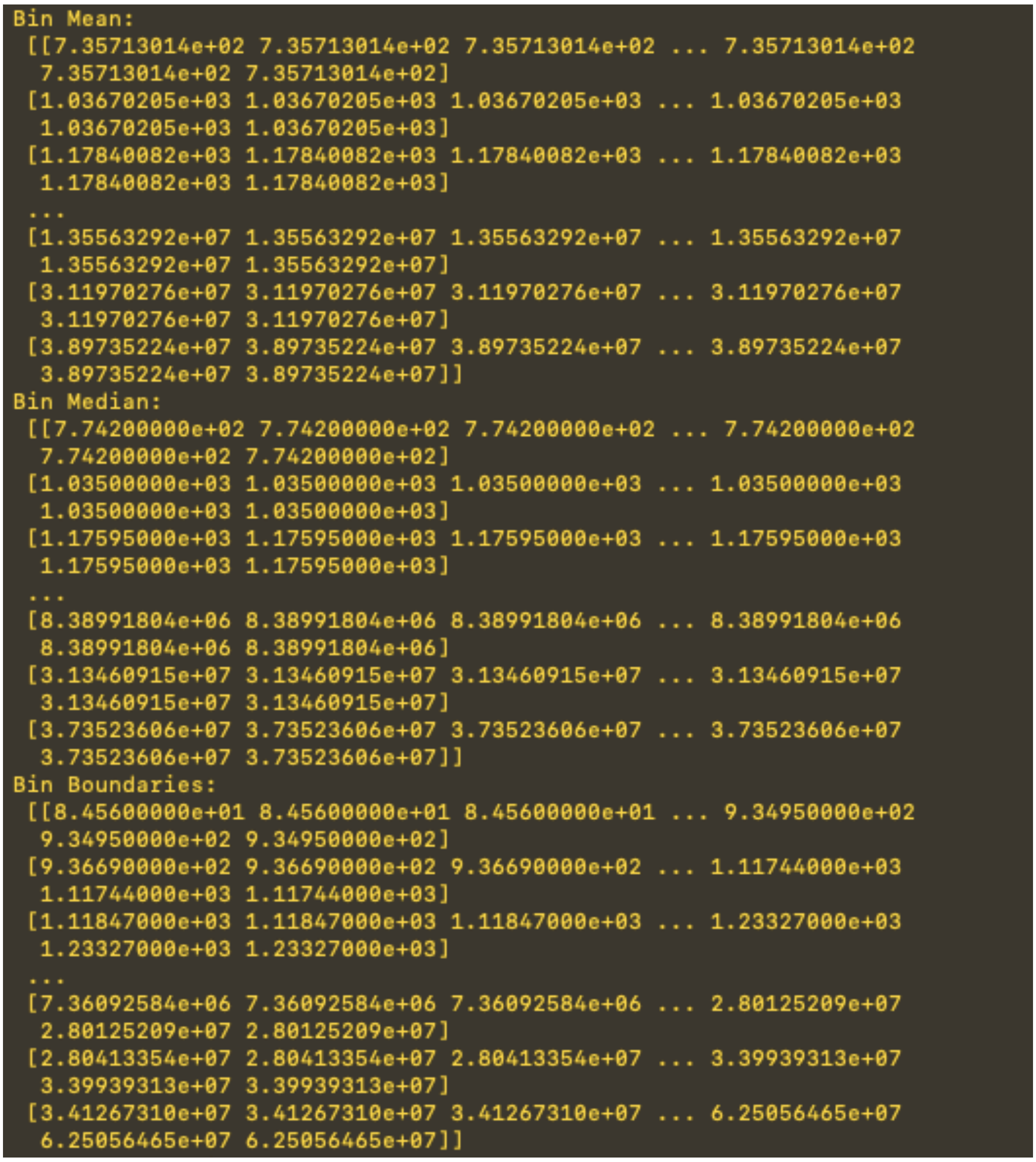
```
list_4225 = []
list_4770 = []
list_total_bags = []
list_small_bags = []
list_large_bags = []
list_xlarge_bags = []
list_region = []

sum_avg_price = 0
sum_total_volume = 0
sum_4046 = 0
sum_4225 = 0
sum_4770 = 0
sum_total_bags = 0
sum_small_bags = 0
sum_large_bags = 0
sum_xlarge_bags = 0

for i in range(1, len(data)):
    if(date == data[i][0][-7:] and region == data[i][12]):
        d = data[i][1]

        if(d.replace('.', '', 1).isdigit() == True):
            sum_avg_price += float(data[i][1])
            c += 1

        sum_total_volume += float(data[i][2])
        sum_4046 += float(data[i][3])
        sum_4225 += float(data[i][4])
        sum_4770 += float(data[i][5])
        sum_total_bags += float(data[i][6])
        sum_small_bags += float(data[i][7])
        sum_large_bags += float(data[i][8])
        sum_xlarge_bags += float(data[i][9])
    else:
        list_date.append(date)

        if(c != 0):

            list_avg_price.append(sum_avg_price/c)
        else:
            list_avg_price.append(0)
        list_total_volume.append(sum_total_volume)
        list_4046.append(sum_4046)
        list_4225.append(sum_4225)
        list_4770.append(sum_4770)
        list_total_bags.append(sum_total_bags)
        list_small_bags.append(sum_small_bags)
        list_large_bags.append(sum_large_bags)
        list_xlarge_bags.append(sum_xlarge_bags)
        list_region.append(region)
        date = data[i][0][-7:]
        region = data[i][12]
        d = data[i][1]
        if(d.replace('.', '', 1).isdigit() == True):
            sum_avg_price = float(data[i][1])
            c = 1
        else:
            sum_avg_price = 0
            c = 0
        sum_total_volume = float(data[i][2])
        sum_4046 = float(data[i][3])
        sum_4225 = float(data[i][4])
        sum_4770 = float(data[i][5])
        sum_total_bags = float(data[i][6])
        sum_small_bags = float(data[i][7])
        sum_large_bags = float(data[i][8])
        sum_xlarge_bags = float(data[i][9])

print("Date     avg_price     total_volume          4046     4225     4770     total_bags
small_bags     large_bags     xlarge_bags     list_region")
```

```
for i in range(20):
        print(list_date[i], "    ", round(list_avg_price[i], 2), "        ",
round(list_total_volume[i], 2), "     ", round(list_4046[i], 2), "     ",
                round(list_4225[i], 2), "         ", round(list_4770[i], 2), "     ",
round(list_total_bags[i], 2), "           ", round(list_small_bags[i], 2), "     ",
                round(list_large_bags[i], 2), "          ", round(list_xlarge_bags[i], 2),
"           ", list_region[i])




temp_lists0 = 0
temp_lists1 = 0
temp_lists2 = 0
temp_lists3 = 0
temp_lists4 = 0
temp_lists5 = 0
temp_lists6 = 0
temp_lists7 = 0
temp_lists8 = 0
temp_lists9 = 0
temp_lists10 = 0
temp_lists11 = 0
temp_lists12 = 0
for i in range(1, len(data)):
        d0 = data[i][0]

        d1 = data[i][1]
        d2 = data[i][2]
        d3 = data[i][3]
        d4 = data[i][4]
        d5 = data[i][5]
        d6 = data[i][6]
        d7 = data[i][7]
        d8 = data[i][8]
        d9 = data[i][9]
        if(d0 == ''):
```

```
        print(d0)
        temp_lists0+=1
    elif(d1 == '' or d1.replace('.', '', 1).isdigit() == False):
        temp_lists1+=1
    elif(d2 == '' or d2.replace('.', '', 1).isdigit() == False):
        temp_lists2+=1
    elif(d3 == '' or d3.replace('.', '', 1).isdigit() == False):
        temp_lists3+=1
    elif(d4 == '' or d4.replace('.', '', 1).isdigit() == False):
        temp_lists4+=1
    elif(d5 == '' or d5.replace('.', '', 1).isdigit() == False):
        temp_lists5+=1
    elif(d6 == '' or d6.replace('.', '', 1).isdigit() == False):
        temp_lists6+=1
    elif(d7 == '' or d7.replace('.', '', 1).isdigit() == False):
        temp_lists7+=1
    elif(d8 == '' or d8.replace('.', '', 1).isdigit() == False):
        temp_lists8+=1
    elif(d9 == '' or d9.replace('.', '', 1).isdigit() == False):
        temp_lists9+=1
    elif(data[i][10] == ''):
        temp_lists10+=1
    elif(data[i][11].isdigit() == False):
        temp_lists11+=1
    elif(data[i][12] == ''):
        temp_lists12+=1

print("The count of the missing values are-")
print("Date - ", temp_lists0)
print("Averge price - ", temp_lists1)
print("Total volume - ", temp_lists2)
print("4046 - ", temp_lists3)
print("4225 - ", temp_lists4)
print("4770 - ", temp_lists5)
print("Total bags - ", temp_lists6)
print("Small bags - ", temp_lists7)
print("Large bags - ", temp_lists8)
print("XLarge bags - ", temp_lists9)
print("Type - ", temp_lists10)
print("Year - ", temp_lists11)
print("Region - ", temp_lists12)

can = 0
for i in range(1, len(data)):
    d = data[i][1]
    if(data[i][1] == '' or d.replace('.', '', 1).isdigit() == False):
        s = 0
        count = 0
        for j in range(1, len(data)):
            d = data[j][1]
            if(data[j][12] == data[i][12] and d.replace('.', '', 1).isdigit() ==
True):
                s = s + float(data[j][1])
                count+=1

        data[i][1] = str(round(s/count, 6))
print("Earlier missing entries : ")
print(data[5])
print(data[6])
print(data[7])
print(data[8])
print(data[9])
print(data[10])
print(data[11])




for i in range(1, len(data)):
    if((data[i][11] == '2015') or (data[i][11] == '2016')):
        data[i][0] = "Old"
        elif(data[i][11] == '2017'):
            data[i][0] = "New"
        elif(data[i][11] == '2018'):
            data[i][0] = "Recent"
print(data[10])
print(data[3000])
print(data[6000])
print(data[9000])
print(data[12000])

print(data[15000])
print(data[18000])
```

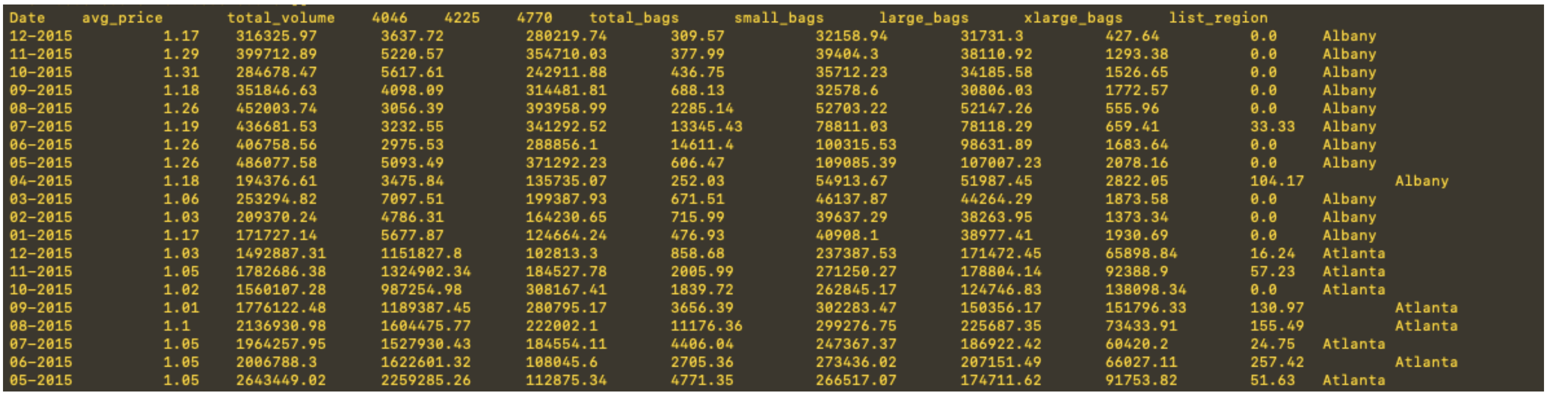
Output:

(a)Bin mean, Bin median, Bin boundaries (Total 250 bins)



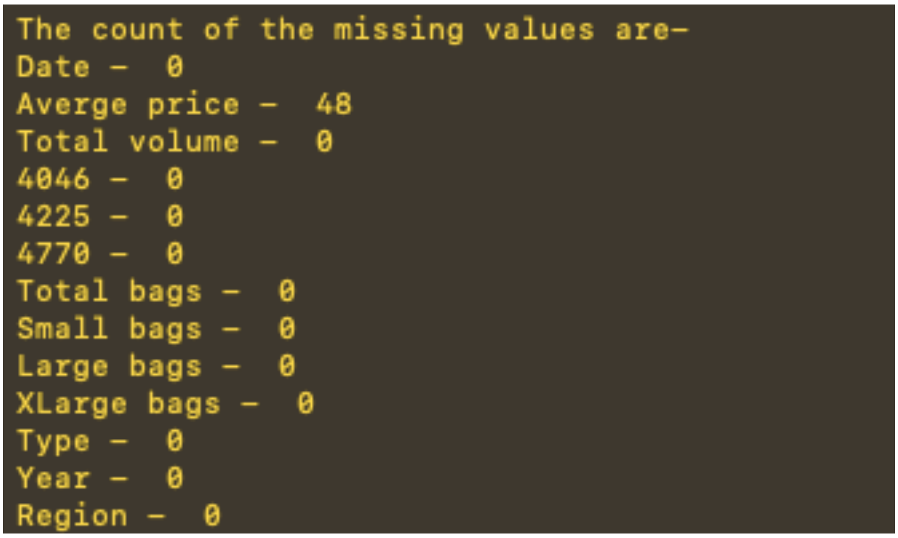
```
Bin Mean:
 [[7.35713014e+02 7.35713014e+02 7.35713014e+02 ... 7.35713014e+02
  7.35713014e+02 7.35713014e+02]
 [1.03670205e+03 1.03670205e+03 1.03670205e+03 ... 1.03670205e+03
  1.03670205e+03 1.03670205e+03]
 [1.17840082e+03 1.17840082e+03 1.17840082e+03 ... 1.17840082e+03
  1.17840082e+03 1.17840082e+03]
 ...
 [1.35563292e+07 1.35563292e+07 1.35563292e+07 ... 1.35563292e+07
  1.35563292e+07 1.35563292e+07]
 [3.11970276e+07 3.11970276e+07 3.11970276e+07 ... 3.11970276e+07
  3.11970276e+07 3.11970276e+07]
 [3.89735224e+07 3.89735224e+07 3.89735224e+07 ... 3.89735224e+07
  3.89735224e+07 3.89735224e+07]]
Bin Median:
 [[7.74200000e+02 7.74200000e+02 7.74200000e+02 ... 7.74200000e+02
  7.74200000e+02 7.74200000e+02]
 [1.03500000e+03 1.03500000e+03 1.03500000e+03 ... 1.03500000e+03
  1.03500000e+03 1.03500000e+03]
 [1.17595000e+03 1.17595000e+03 1.17595000e+03 ... 1.17595000e+03
  1.17595000e+03 1.17595000e+03]
 ...
 [8.38991804e+06 8.38991804e+06 8.38991804e+06 ... 8.38991804e+06
  8.38991804e+06 8.38991804e+06]
 [3.13460915e+07 3.13460915e+07 3.13460915e+07 ... 3.13460915e+07
  3.13460915e+07 3.13460915e+07]
 [3.73523606e+07 3.73523606e+07 3.73523606e+07 ... 3.73523606e+07
  3.73523606e+07 3.73523606e+07]]
Bin Boundaries:
 [[8.45600000e+01 8.45600000e+01 8.45600000e+01 ... 9.34950000e+02
  9.34950000e+02 9.34950000e+02]
 [9.36690000e+02 9.36690000e+02 9.36690000e+02 ... 1.11744000e+03
  1.11744000e+03 1.11744000e+03]
 [1.11847000e+03 1.11847000e+03 1.11847000e+03 ... 1.23327000e+03
  1.23327000e+03 1.23327000e+03]
 ...
 [7.36092584e+06 7.36092584e+06 7.36092584e+06 ... 2.80125209e+07
  2.80125209e+07 2.80125209e+07]
 [2.80413354e+07 2.80413354e+07 2.80413354e+07 ... 3.39939313e+07
  3.39939313e+07 3.39939313e+07]
 [3.41267310e+07 3.41267310e+07 3.41267310e+07 ... 6.25056465e+07
  6.25056465e+07 6.25056465e+07]]
```

(b) Reduced format



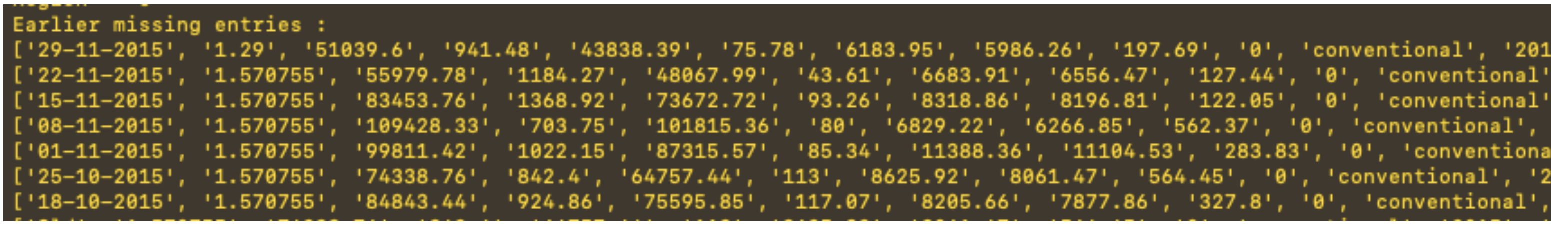
```
Date     avg_price        total_volume     4046     4225     4770     total_bags       small_bags       large_bags       xlarge_bags      list_region
12-2015          1.17     316325.97        3637.72           280219.74        309.57           32158.94         31731.3          427.64           0.0      Albany
11-2015          1.29     399712.89        5220.57           354710.03        377.99           39404.3          38110.92         1293.38          0.0      Albany
10-2015          1.31     284678.47        5617.61           242911.88        436.75           35712.23         34185.58         1526.65          0.0      Albany
09-2015          1.18     351846.63        4098.09           314481.81        688.13           32578.6          30806.03         1772.57          0.0      Albany
08-2015          1.26     452003.74        3056.39           393958.99        2285.14          52703.22         52147.26         555.96           0.0      Albany
07-2015          1.19     436681.53        3232.55           341292.52        13345.43         78811.03         78118.29         659.41           33.33    Albany
06-2015          1.26     406758.56        2975.53           288856.1         14611.4          100315.53        98631.89         1683.64          0.0      Albany
05-2015          1.26     486077.58        5093.49           371292.23        606.47           109085.39        107007.23        2078.16          0.0      Albany
04-2015          1.18     194376.61        3475.84           135735.07        252.03           54913.67         51987.45         2822.05          104.17           Albany
03-2015          1.06     253294.82        7097.51           199387.93        671.51           46137.87         44264.29         1873.58          0.0      Albany
02-2015          1.03     209370.24        4786.31           164230.65        715.99           39637.29         38263.95         1373.34          0.0      Albany
01-2015          1.17     171727.14        5677.87           124664.24        476.93           40908.1          38977.41         1930.69          0.0      Albany
12-2015          1.03     1492887.31       1151827.8         102813.3         858.68           237387.53        171472.45        65898.84         16.24    Atlanta
11-2015          1.05     1782686.38       1324902.34        184527.78        2005.99          271250.27        178804.14        92388.9          57.23    Atlanta
10-2015          1.02     1560107.28       987254.98         308167.41        1839.72          262845.17        124746.83        138098.34        0.0      Atlanta
09-2015          1.01     1776122.48       1189387.45        280795.17        3656.39          302283.47        150356.17        151796.33        130.97           Atlanta
08-2015          1.1      2136930.98       1604475.77        222002.1         11176.36         299276.75        225687.35        73433.91         155.49           Atlanta
07-2015          1.05     1964257.95       1527930.43        184554.11        4406.04          247367.37        186922.42        60420.2          24.75    Atlanta
06-2015          1.05     2006788.3        1622601.32        108045.6         2705.36          273436.02        207151.49        66027.11         257.42           Atlanta
05-2015          1.05     2643449.02       2259285.26        112875.34        4771.35          266517.07        174711.62        91753.82         51.63    Atlanta
```

(c) Average price missing 48 entries



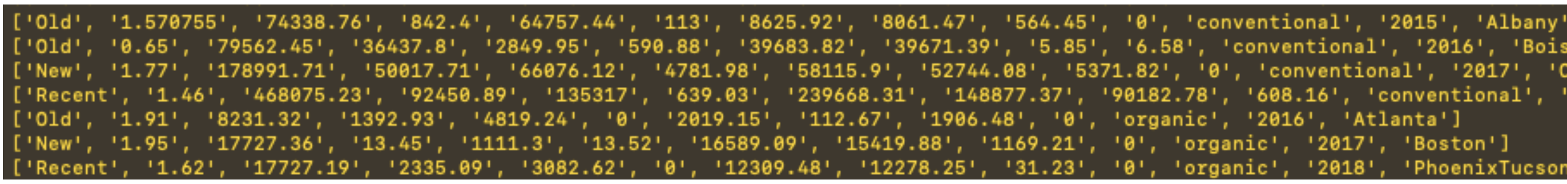
```
The count of the missing values are-
Date -  0
Averge price -  48
Total volume -  0
4046 -  0
4225 -  0
4770 -  0
Total bags -  0
Small bags -  0
Large bags -  0
XLarge bags -  0
Type -  0
Year -  0
Region -  0
```

(d) Replaced avg price with mean 1.570755



```
Earlier missing entries :
['29-11-2015', '1.29', '51039.6', '941.48', '43838.39', '75.78', '6183.95', '5986.26', '197.69', '0', 'conventional', '201
['22-11-2015', '1.570755', '55979.78', '1184.27', '48067.99', '43.61', '6683.91', '6556.47', '127.44', '0', 'conventional'
['15-11-2015', '1.570755', '83453.76', '1368.92', '73672.72', '93.26', '8318.86', '8196.81', '122.05', '0', 'conventional'
['08-11-2015', '1.570755', '109428.33', '703.75', '101815.36', '80', '6829.22', '6266.85', '562.37', '0', 'conventional',
['01-11-2015', '1.570755', '99811.42', '1022.15', '87315.57', '85.34', '11388.36', '11104.53', '283.83', '0', 'conventiona
['25-10-2015', '1.570755', '74338.76', '842.4', '64757.44', '113', '8625.92', '8061.47', '564.45', '0', 'conventional', '2
['18-10-2015', '1.570755', '84843.44', '924.86', '75595.85', '117.07', '8205.66', '7877.86', '327.8', '0', 'conventional',
```

(e)According to year, they have been categorised as Old, New, recent. Different examples have been taken from 2015, 2016, 2017, 2018 to illustrate.



```
['Old', '1.570755', '74338.76', '842.4', '64757.44', '113', '8625.92', '8061.47', '564.45', '0', 'conventional', '2015', 'Albany'
['Old', '0.65', '79562.45', '36437.8', '2849.95', '590.88', '39683.82', '39671.39', '5.85', '6.58', 'conventional', '2016', 'Bois
['New', '1.77', '178991.71', '50017.71', '66076.12', '4781.98', '58115.9', '52744.08', '5371.82', '0', 'conventional', '2017', 'C
['Recent', '1.46', '468075.23', '92450.89', '135317', '639.03', '239668.31', '148877.37', '90182.78', '608.16', 'conventional', '
['Old', '1.91', '8231.32', '1392.93', '4819.24', '0', '2019.15', '112.67', '1906.48', '0', 'organic', '2016', 'Atlanta']
['New', '1.95', '17727.36', '13.45', '1111.3', '13.52', '16589.09', '15419.88', '1169.21', '0', 'organic', '2017', 'Boston']
['Recent', '1.62', '17727.19', '2335.09', '3082.62', '0', '12309.48', '12278.25', '31.23', '0', 'organic', '2018', 'PhoenixTucson
```