Analysis and Systems of Big Data Practise

Lab - 4

Sreepathy Jayanand
CED17I038

Q1

Select a **subset of relevant attributes** from the given dataset that are necessary to know about the total volume of avocados with product lookup codes (PLU) 4046, 4225, 4770) which are of organic type. (Use AVOCADO dataset)

Logic:

The data is read and if type is "organic" it is printed

Packages used:

1. Pandas

Code:

```
data_avoc1 = data_avoc[['Date','Total Volume','4046','4225','4770','type']]
only_avoc = data_avoc1[data_avoc1['type']=='organic']
only_avoc
```

Output:

	Date	Total Volume	4046	4225	4770	type
9126	27-12-2015	989.55	8.16	88.59	0.00	organic
9127	20-12-2015	1163.03	30.24	172.14	0.00	organic
9128	13-12-2015	995.96	10.44	178.70	0.00	organic
9129	06-12-2015	1158.42	90.29	104.18	0.00	organic
9130	29-11-2015	831.69	0.00	94.73	0.00	organic
...	...	...	...	...	...	...
18245	28-01-2018	13888.04	1191.70	3431.50	0.00	organic
18246	21-01-2018	13766.76	1191.92	2452.79	727.94	organic
18247	14-01-2018	16205.22	1527.63	2981.04	727.01	organic
18248	07-01-2018	17489.58	2894.77	2356.13	224.53	organic
18249	18-03-2018	15896.38	2055.35	1499.55	0.00	organic

9124 rows × 6 columns

Q2

Discard all **duplicate** entries in the given dataset and fill all the missing values in the attribute "AveragePrice" as 1.25. Also print the size of the dataset before and after removing duplicates. (Use Trail dataset)

Logic:

The duplicates are removed using inbuilt libraries and Null/NaN values are replaced with 1.25

Packages used:

1. Pandas

Code:

```
print("Shape before - ",data_trail.shape)
data_trail2 = data_trail.drop_duplicates()
print("Shape after removing duplicates - ",data_trail2.shape)
data_trail2 = data_trail2['AveragePrice']
data_trail2.fillna(1.25,inplace=True)
data_trail2
```

Output:

```
Shape before -  (202, 13)
Shape after removing duplicates -  (195, 13)
0       1.35
1       0.93
2       1.08
3       1.25
5       1.21
        ...
197     1.25
198     1.25
199     1.25
200     1.25
201     1.25
Name: AveragePrice, Length: 195, dtype: object
```

Q3

Binarize the attribute “Year”. Set the threshold above 2016 and print it without truncation. (Use AVOCADO dataset)

Logic:

The year is checked and if greater than 2016 bool True is assigned, else false is assigned.

Packages used:

1. Pandas

Code:

```
data_avoc1 = data_avoc.copy()
data_avoc1['year'] = (data_avoc1['year']>=2016)
data_avoc1
```

Output:

Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
27-12-2015	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	False	Albany
20-12-2015	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventional	False	Albany
13-12-2015	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conventional	False	Albany
06-12-2015	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conventional	False	Albany
29-11-2015	1.29	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conventional	False	Albany
...	...	...	...	...	...	...	...	...	...	...	...	...
28-01-2018	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80	0.0	organic	True	WestTexNewMexico
21-01-2018	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31	0.0	organic	True	WestTexNewMexico
14-01-2018	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00	0.0	organic	True	WestTexNewMexico
07-01-2018	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01	0.0	organic	True	WestTexNewMexico
18-03-2018	1.56	15896.38	2055.35	1499.55	0.00	12341.48	12114.81	226.67	0.0	organic	True	WestTexNewMexico

Q4

Transform all categorical attributes in the dataset AVOCADO using **Integer Encoding.**

Logic:

Unique values in attributes are selected to give them an encoding

Packages used:

1. Pandas

Code:

```
data_avoc1 = data_avoc.copy()

label_encoder = LabelEncoder()
cat_cols = ['type','year','region']
data_avoc1['type'] = label_encoder.fit_transform(data_avoc1['type'])
data_avoc1['year'] = label_encoder.fit_transform(data_avoc1['year'])
data_avoc1['region'] = label_encoder.fit_transform(data_avoc1['region'])
data_avoc1
```

Output:

	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	reg
0	27-12-2015	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	0	0	
1	20-12-2015	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	0	0	
2	13-12-2015	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	0	0	
3	06-12-2015	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	0	0	
4	29-11-2015	1.29	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
18245	28-01-2018	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80	0.0	1	3	
18246	21-01-2018	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31	0.0	1	3	
18247	14-01-2018	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00	0.0	1	3	
18248	07-01-2018	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01	0.0	1	3	
18249	18-03-2018	1.56	15896.38	2055.35	1499.55	0.00	12341.48	12114.81	226.67	0.0	1	3	

Q5

Transform the attribute = “Region” in the given dataset AVOCADO using **One-Hot Encoding**.

Logic:

The unique values are obtained for the region attribute and then the one-hot encoding is done using the inbuilt packages.

Packages used:

1. Pandas
2. OneHotEncoder
3. Sklearn.preprocessing

Code:

```
data_avoc1 = data_avoc.copy()

enc = OneHotEncoder(handle_unknown='ignore')
enc_data = pd.DataFrame(enc.fit_transform(data_avoc1[['region']]).toarray())

data_avoc1 = data_avoc1.join(enc_data)
data_avoc1
```

Output:

25	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
.85	48.16	8696.87	8603.62	93.25	0.0	conventional	2015	Albany	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
.81	58.33	9505.56	9408.07	97.49	0.0	conventional	2015	Albany	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
.67	130.50	8145.35	8042.21	103.14	0.0	conventional	2015	Albany	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
.41	72.58	5811.16	5677.40	133.76	0.0	conventional	2015	Albany	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
.39	75.78	6183.95	5986.26	197.69	0.0	conventional	2015	Albany	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
.50	0.00	9264.84	8940.04	324.80	0.0	organic	2018	WestTexNewMexico	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
.79	727.94	9394.11	9351.80	42.31	0.0	organic	2018	WestTexNewMexico	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
.04	727.01	10969.54	10919.54	50.00	0.0	organic	2018	WestTexNewMexico	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
.13	224.53	12014.15	11988.14	26.01	0.0	organic	2018	WestTexNewMexico	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
.55	0.00	12341.48	12114.81	226.67	0.0	organic	2018	WestTexNewMexico	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Q6

Ignore the tuples that hold missing values and print the subset of data from AVOCADO dataset.

Logic:

Using the inbuilt dropna command we find missing / NULL/ NaN entries

Packages used:

1. Pandas

Code:

```
data_avoc1 = data_avoc.copy()

data_avoc1['AveragePrice'] = pd.to_numeric(data_avoc1['AveragePrice'],errors='coerce')
data_avoc1 = data_avoc1.dropna(how='any',axis=0)
data_avoc1
```

Output:

	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	
0	27-12-2015	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	c
1	20-12-2015	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	c
2	13-12-2015	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	c
3	06-12-2015	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	c
4	29-11-2015	1.29	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	c
...	...	...	...	...	...	...	...	...	...	...	
18245	28-01-2018	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80	0.0	
18246	21-01-2018	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31	0.0	
18247	14-01-2018	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00	0.0	
18248	07-01-2018	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01	0.0	
18249	18-03-2018	1.56	15896.38	2055.35	1499.55	0.00	12341.48	12114.81	226.67	0.0	

Q7

Drop the attribute that has **high nullity** as it facilitates efficient prediction. (Use AVOCADO dataset)

Logic:

The attributes with most null values us found out using .isNull() function and the count is taken. Any attribute which doesn't have any null values is only taken for the final table.

Packages used:

1. Pandas

Code:

```
data_avoc1 = data_avoc.copy()
print("NULL value count :")
data_avoc1['AveragePrice'] = pd.to_numeric(data_avoc1['AveragePrice'],errors='coerce')
print(data_avoc1.isnull().sum())
print("Updated table :")
data_avoc1 = data_avoc1.loc[:,data_avoc1.isnull().sum() == 0]
data_avoc1
```

Output:

```
NULL value count :
Date            0
AveragePrice   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
dtype: int64
Updated table :
```

	Date	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
0	27-12-2015	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	2015	Albany
1	20-12-2015	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventional	2015	Albany
2	13-12-2015	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conventional	2015	Albany
3	06-12-2015	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conventional	2015	Albany
4	29-11-2015	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conventional	2015	Albany
...	...	...	...	...	...	...	...	...	...	...	...	...
18245	28-01-2018	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80	0.0	organic	2018	WestTexNewMexico
18246	21-01-2018	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31	0.0	organic	2018	WestTexNewMexico
18247	14-01-2018	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00	0.0	organic	2018	WestTexNewMexico
18248	07-01-2018	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01	0.0	organic	2018	WestTexNewMexico
18249	18-03-2018	15896.38	2055.35	1499.55	0.00	12341.48	12114.81	226.67	0.0	organic	2018	WestTexNewMexico

Q8

Study the entire dataset and report the complete **statistical summary** about the data (Use AVOCADO dataset)

Logic:

All the values like mean, median, correlation, skewness for the 4046 attribute is calculated using inbuilt packages.

Packages used:

1. Pandas

2. Statistics

Code:

```
print('mean total volume: ',statistics.mean(data_avoc['Total Volume']))
print('mean 4046:         ',statistics.mean(data_avoc['4046']))
print('mean total Bags   : ',statistics.mean(data_avoc['Total Bags']))

print('std total volume: ',statistics.stdev(data_avoc['Total Volume']))
print('std 4046:         ',statistics.stdev(data_avoc['4046']))
print('std total bags   : ',statistics.stdev(data_avoc['Total Bags']))

print('lower quartile totalvolume ',data_avoc['Total Volume'].quantile(0.25))
print('lower quartile 4046        ',data_avoc['4046'].quantile(0.25))
print('lower quartile total bags  ',data_avoc['year'].quantile(0.25))
print('median totalvolume ',data_avoc['Total Volume'].quantile(0.5))
print('median 4046        ',data_avoc['4046'].quantile(0.5))
print('median total bags  ',data_avoc['year'].quantile(0.5))
print('upper quartile totalvolume ',data_avoc['Total Volume'].quantile(0.75))
print('upper quartile 4046        ',data_avoc['4046'].quantile(0.75))
print('upper quartile total bags  ',data_avoc['year'].quantile(0.75))

data_avoc['type'].value_counts()
```

Output:

```
mean total volume:  850598.2734126027
mean 4046:          292992.48189643834
mean total Bags  :  239626.747390137
std total volume:  3453456.259184955
std 4046:          1264956.2556407494
std total bags  :  986216.8122681369
lower quartile totalvolume  10839.627499999999
lower quartile 4046         854.21
lower quartile total bags   2015.0
median totalvolume  107365.505
median 4046         8643.2
median total bags   2016.0
upper quartile totalvolume  432952.665
upper quartile 4046         111008.7125
upper quartile total bags   2017.0
conventional    9126
organic         9124
Name: type, dtype: int64
```

Correlation :

	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	year
Total Volume	1.000000	0.977863	0.974181	0.872203	0.963047	0.967238	0.880640	0.747158	0.017165
4046	0.977863	1.000000	0.926110	0.833390	0.920057	0.925280	0.838645	0.699378	0.003328
4225	0.974181	0.926110	1.000000	0.887855	0.905788	0.916031	0.810016	0.688809	-0.009584
4770	0.872203	0.833390	0.887855	1.000000	0.792315	0.802733	0.698472	0.679862	-0.036550
Total Bags	0.963047	0.920057	0.905788	0.792315	1.000000	0.994335	0.943009	0.804233	0.071520
Small Bags	0.967238	0.925280	0.916031	0.802733	0.994335	1.000000	0.902589	0.806845	0.063883
Large Bags	0.880640	0.838645	0.810016	0.698472	0.943009	0.902589	1.000000	0.710859	0.087858
XLarge Bags	0.747158	0.699378	0.688809	0.679862	0.804233	0.806845	0.710859	1.000000	0.081005
year	0.017165	0.003328	-0.009584	-0.036550	0.071520	0.063883	0.087858	0.081005	1.000000

```
skewness : Total Volume     9.007930
4046                 8.648456
4225                 8.942706
4770                10.159671
Total Bags           9.756334
Small Bags           9.540917
Large Bags           9.796719
XLarge Bags         13.140106
year                 0.215371
dtype: float64
```

Q9

Test drive the use of Gini Index, Information gain, Entropy, and other measures that are supported in your platform, performing the role of data selection.

Logic:

The Gini index, Information Gain, Entropy is calculated using the inbuilt functions.

Packages used:

1. Pandas
2. Scipy
3. Entropy
4. Info_gain
5. Gini

Code:

```
from pygini import gini
import pandas as pd
import numpy as np
x1=[0 for i in range (18250)]
x2=[0 for i in range (18250)]
x3=[0 for i in range (18250)]

df = pd.read_csv("/content/sample_data/trail.csv")
for i in range (df.shape[0]):
  y=float(df["year"][i])
  z=float(df["Total Volume"][i])
  a=float(df["4046"][i])
  x1[i]=y
  x2[i]=z
  x3[i]=a

m = np.asarray(x1)
n = np.asarray(x2)
o = np.asarray(x3)

GI1 = gini(m)
GI2 = gini(n)
GI3 = gini(o)
print('Gini index of year = ',GI1)
print('Gini index of total volume = ',GI2)
print('Gini index of 4046 = ',GI3)
```

```python
from scipy.stats import entropy
import pandas as pd

df = pd.read_csv("/content/sample_data/trail.csv")

print('entropy year         ',entropy(df["year"]))
print('entropy Total Volume',entropy(df["Total Volume"]))
print('entropy Total Volume',entropy(df["Total Bags"]))
```

```python
from info_gain import info_gain

ig1  = info_gain.info_gain(df["Total Volume"], df["Total Bags"])
ig2  = info_gain.info_gain(df["Total Volume"], df["4046"])
ig3  = info_gain.info_gain(df["4046"], df["Total Bags"])

print('information gain (Total Volume wrt Total Bags)',ig1)
print('information gain (Total Volume wrt LOP4046)',ig2)
print('information gain (LOp4046 wrt Total Bags)',ig3)
```

Output:

```
Gini index of year =  0.9889315064059077
Gini index of total volume =  0.9936196008889837
Gini index of 4046 =  0.9955881761075827
```

```
entropy year         5.308267697401204
entropy Total Volume 4.98765195248629
entropy Total Volume 4.840911980171588
```

```
information gain (Total Volume wrt Total Bags) 5.225005408274409
information gain (Total Volume wrt LOP4046) 5.225005408274409
information gain (LOp4046 wrt Total Bags) 5.225005408274409
```

Q10

Test drive the implementation support in your platform of choice for data preprocessing phases such as cleaning, selection, transformation, integration in addition to the earlier exercises.

Logic:

The null values are not appended into the list. Following which we calculate the min-max and z-score normalization for the created list.

Packages used:

1. Pandas
2. Entropy
3. Statistics

Code:

```
from scipy.stats import entropy
import statistics
import pandas as pd

df = pd.read_csv("/content/sample_data/trail.csv")
z=[]

for i in range (df.shape[0]):
  x=df["AveragePrice"][i]
  if(str(x).startswith('n') or str(x).startswith('N') or str(x) == "" ):
    continue
  else:
    z.append(float(df["AveragePrice"][i]))
    print(i,x)

m1=statistics.mean(z)
m2=statistics.median(z)
#cleaning 1 -> using mean dor null values
for i in range (df.shape[0]):
  x=df["AveragePrice"][i]
  if(str(x).startswith('n') or str(x).startswith('N') or str(x) == "" ):
    df["AveragePrice"]=m1
#cleaning 1 -> using median dor null values
for i in range (df.shape[0]):
  x=df["AveragePrice"][i]
  if(str(x).startswith('n') or str(x).startswith('N') or str(x) == "" ):
    df["AveragePrice"]=m2
```

```python
df = pd.read_csv("/content/sample_data/trail.csv")
z=[]
min_max_TV=[]
z_score_TV=[]
z=df['Total Volume'].to_list()

mini=df['Total Volume'].min()
maxi=df['Total Volume'].max()
di= maxi-mini

mean1=statistics.mean(df["Total Volume"])
std1=statistics.stdev(df["Total Volume"])

print('min-max normalisation     z_score normalisation \n')
for i in range (df.shape[0]):
  min_max_TV.append((z[i]-mini)/di * 1)#min max
  z_score_TV.append((z[i]-mean1)/std1)
  print(min_max_TV[i],'   ',z_score_TV[i])
```

Output:

```
min-max normalisation     z_score normalisation

0.014861361651129569     -1.1376312042037628
0.08037226575258084     -0.9213487013781145
0.03980177501268471     -1.0552910946093377
0.016001900484367025     -1.133865745662076
0.014861361651129569     -1.1376312042037628
0.08037226575258084     -0.9213487013781145
0.03980177501268471     -1.0552910946093377
0.010892662967878957     -1.150733757068016
0.016001900484367025     -1.133865745662076
0.03980177501268471     -1.0552910946093377
0.016001900484367025     -1.133865745662076
0.03980177501268471     -1.0552910946093377
0.010892662967878957     -1.150733757068016
0.014861361651129569     -1.1376312042037628
0.08037226575258084     -0.9213487013781145
0.006039322734223629 6     -1.166756930419033
0.0026252974160817 63     -1.1780282439605827
0.0     -1.1866955933675234
0.015604718281323393     -1.135177032113245
0.005344533118546 3345     -1.1690507598063247
0.001096543075197 9986     -1.1830753856477112
0.011114368907016713     -1.1500180082247 62
0.03140540074302568     -1.08301150034348
```