Data_Preprocessing_Part_2_Practice

December 5, 2024

[51]:

```
import numpy as np
import pandas as pd

from sklearn.preprocessing import LabelEncoder, OrdinalEncoder, OneHotEncoder
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

[9]:

```
data = {'id':[1,2,3,4,5,6,7,8,9,10],
       'product':
 ↪['apple','orange','banana','carrot','laptop','phone','shirt','pants','pen','box'],
       'color':
 ↪['red','orange','yellow','orange','black','white','blue','black','red','black'],
        'category':
 ↪['fruit','fruit','fruit','vegetable','electronic','electronic','cloth','cloth','stationary'
        'sales':
 ↪['low','high','average','average','low','high','high','high','low','high'],
       'rating':[5,4,3,2,5,4,3,4,5,4],
       'price':[100,120,25,15,80000,20000,300,500,15,50]}

df = pd.DataFrame(data)
df
```

[9]:

```
   id product   color    category    sales  rating  price
0   1   apple     red       fruit      low       5    100
1   2  orange  orange       fruit     high       4    120
2   3  banana  yellow       fruit  average       3     25
3   4  carrot  orange   vegetable  average       2     15
4   5  laptop   black  electronic      low       5  80000
5   6   phone   white  electronic     high       4  20000
6   7   shirt    blue       cloth     high       3    300
7   8   pants   black       cloth     high       4    500
8   9     pen     red  stationary      low       5     15
9  10     box   black  stationary     high       4     50
```

1 Feature Encoding

2 Applying label encoding on 'category' column

[11]:

```
label_encoding = LabelEncoder() # create an object/instance
# normally the original values are replaced with encoded values; new columns␣
 ↪need not be created
df['category_encoded'] = label_encoding.fit_transform(df['category'])
df
```

[11]:

```
   id product   color    category    sales  rating  price  category_encoded
0   1   apple     red       fruit      low       5    100                 2
1   2  orange  orange       fruit     high       4    120                 2
2   3  banana  yellow       fruit  average       3     25                 2
3   4  carrot  orange   vegetable  average       2     15                 4
4   5  laptop   black  electronic      low       5  80000                 1
5   6   phone   white  electronic     high       4  20000                 1
6   7   shirt    blue       cloth     high       3    300                 0
7   8   pants   black       cloth     high       4    500                 0
8   9     pen     red  stationary      low       5     15                 3
9  10     box   black  stationary     high       4     50                 3
```

[12]:

```
df['product_encoded'] = label_encoding.fit_transform(df['product'])
df
```

[12]:

```
   id product   color    category    sales  rating  price  category_encoded  \
0   1   apple     red       fruit      low       5    100                 2
1   2  orange  orange       fruit     high       4    120                 2
2   3  banana  yellow       fruit  average       3     25                 2
3   4  carrot  orange   vegetable  average       2     15                 4
4   5  laptop   black  electronic      low       5  80000                 1
5   6   phone   white  electronic     high       4  20000                 1
6   7   shirt    blue       cloth     high       3    300                 0
7   8   pants   black       cloth     high       4    500                 0
8   9     pen     red  stationary      low       5     15                 3
9  10     box   black  stationary     high       4     50                 3

   product_encoded
0                0
1                5
2                1
3                3
4                4
5                8
6                9
7                6
8                7
```

```
9                 2
```

3 Applying ordinal encoding on ‘sales’ column

[15]:
```
sales_order = ['low','average','high'] # for ordinal encoding, the order of␣
 ↪values must be explicitly specified
sales_encoding = OrdinalEncoder(categories=[sales_order]) # create an object/
 ↪instance and give the 'categories' parameter as the defined order

df['sales_encoding'] = sales_encoding.fit_transform(df[['sales']])
df
```

[15]:
```
   id product   color    category    sales  rating  price  category_encoded  \
0   1   apple     red       fruit      low       5    100                 2
1   2  orange  orange       fruit     high       4    120                 2
2   3  banana  yellow       fruit  average       3     25                 2
3   4  carrot  orange   vegetable  average       2     15                 4
4   5  laptop   black  electronic      low       5  80000                 1
5   6   phone   white  electronic     high       4  20000                 1
6   7   shirt    blue       cloth     high       3    300                 0
7   8   pants   black       cloth     high       4    500                 0
8   9     pen     red  stationary      low       5     15                 3
9  10     box   black  stationary     high       4     50                 3

   product_encoded  sales_encoding
0                0             0.0
1                5             2.0
2                1             1.0
3                3             1.0
4                4             0.0
5                8             2.0
6                9             2.0
7                6             2.0
8                7             0.0
9                2             2.0
```

[17]:
```
color_order = ['black','red','blue','yellow','orange','white'] # for ordinal␣
 ↪encoding, the order of values must be explicitly specified
color_encoding = OrdinalEncoder(categories=[color_order]) # create an object/
 ↪instance and give the 'categories' parameter as the defined order

df['color_encoding'] = color_encoding.fit_transform(df[['color']])
df
```

[17]:
```
   id product   color    category    sales  rating  price  category_encoded  \
0   1   apple     red       fruit      low       5    100                 2
```

```
1   2  orange  orange       fruit     high        4    120                 2
2   3  banana  yellow       fruit  average        3     25                 2
3   4  carrot  orange   vegetable  average        2     15                 4
4   5  laptop   black  electronic      low        5  80000                 1
5   6   phone   white  electronic     high        4  20000                 1
6   7   shirt    blue       cloth     high        3    300                 0
7   8   pants   black       cloth     high        4    500                 0
8   9     pen     red  stationary      low        5     15                 3
9  10     box   black  stationary     high        4     50                 3

   product_encoded  sales_encoding  color_encoding
0                0             0.0             1.0
1                5             2.0             4.0
2                1             1.0             3.0
3                3             1.0             4.0
4                4             0.0             0.0
5                8             2.0             5.0
6                9             2.0             2.0
7                6             2.0             0.0
8                7             0.0             1.0
9                2             2.0             0.0
```

[ ]:

```
# to do multiple columns at the same time
# sales_order = ['low','average','high']
# color_order = ['black','red','blue','yellow','orange','white']
# ordinal_encoding = OrdinalEncoder(categories=[sales_order, color_order])

# df[['sales_encoding','color_encoding']] = ordinal_encoding.
 ↪fit_transform(df[['sales','color']])
# df
```

4 On Hot Encoding

[18]:

```
# Pandas method
color_onehot_encoding = pd.get_dummies(df['color'])
color_onehot_encoding
```

[18]:

```
   black   blue  orange    red  white  yellow
0  False  False   False   True  False   False
1  False  False    True  False  False   False
2  False  False   False  False  False    True
3  False  False    True  False  False   False
4   True  False   False  False  False   False
5  False  False   False  False   True   False
6  False   True   False  False  False   False
7   True  False   False  False  False   False
```

```
8  False  False   False   True  False   False
9   True  False   False  False  False   False
```

[19]:

```
color_onehot_encoding = pd.get_dummies(df,columns=['color'])
color_onehot_encoding
```

[19]:

```
   id product    category    sales  rating  price  category_encoded  \
0   1   apple       fruit      low       5    100                 2
1   2  orange       fruit     high       4    120                 2
2   3  banana       fruit  average       3     25                 2
3   4  carrot   vegetable  average       2     15                 4
4   5  laptop  electronic      low       5  80000                 1
5   6   phone  electronic     high       4  20000                 1
6   7   shirt       cloth     high       3    300                 0
7   8   pants       cloth     high       4    500                 0
8   9     pen  stationary      low       5     15                 3
9  10     box  stationary     high       4     50                 3

   product_encoded  sales_encoding  color_encoding  color_black  color_blue  \
0                0             0.0             1.0        False       False
1                5             2.0             4.0        False       False
2                1             1.0             3.0        False       False
3                3             1.0             4.0        False       False
4                4             0.0             0.0         True       False
5                8             2.0             5.0        False       False
6                9             2.0             2.0        False        True
7                6             2.0             0.0         True       False
8                7             0.0             1.0        False       False
9                2             2.0             0.0         True       False

   color_orange  color_red  color_white  color_yellow
0         False       True        False         False
1          True      False        False         False
2         False      False        False          True
3          True      False        False         False
4         False      False        False         False
5         False      False         True         False
6         False      False        False         False
7         False      False        False         False
8         False       True        False         False
9         False      False        False         False
```

[43]:

```
#can change prefix, dtype of new columns and also specify drop_first as True to␣
 ↪avoid multi-collinearity (dummy variable trap)
color_onehot_encoding = pd.
 ↪get_dummies(df,columns=['color'],prefix='col',dtype='int', drop_first=True)
color_onehot_encoding
```

[43]:

```
   id product    category    sales  rating  price  category_encoded  \
0   1   apple       fruit      low       5    100                 2   
1   2  orange       fruit     high       4    120                 2   
2   3  banana       fruit  average       3     25                 2   
3   4  carrot   vegetable  average       2     15                 4   
4   5  laptop  electronic      low       5  80000                 1   
5   6   phone  electronic     high       4  20000                 1   
6   7   shirt       cloth     high       3    300                 0   
7   8   pants       cloth     high       4    500                 0   
8   9     pen  stationary      low       5     15                 3   
9  10     box  stationary     high       4     50                 3   

   product_encoded  sales_encoding  color_encoding  col_blue  col_orange  \
0                0             0.0             1.0         0           0   
1                5             2.0             4.0         0           1   
2                1             1.0             3.0         0           0   
3                3             1.0             4.0         0           1   
4                4             0.0             0.0         0           0   
5                8             2.0             5.0         0           0   
6                9             2.0             2.0         1           0   
7                6             2.0             0.0         0           0   
8                7             0.0             1.0         0           0   
9                2             2.0             0.0         0           0   

   col_red  col_white  col_yellow  
0        1          0           0  
1        0          0           0  
2        0          0           1  
3        0          0           0  
4        0          0           0  
5        0          1           0  
6        0          0           0  
7        0          0           0  
8        1          0           0  
9        0          0           0  
```

[49]:

```python
# using sklearn
one_hot_encoding = OneHotEncoder()
color_mod_encoding = one_hot_encoding.fit_transform(df[['color']])
print(color_mod_encoding)
```

```
  (0, 3)        1.0
  (1, 2)        1.0
  (2, 5)        1.0
  (3, 2)        1.0
  (4, 0)        1.0
  (5, 4)        1.0
  (6, 1)        1.0
  (7, 0)        1.0
  (8, 3)        1.0
  (9, 0)        1.0
```

[48]:

```
one_hot_df = pd.DataFrame(color_mod_encoding, columns=one_hot_encoding.
 ↪get_feature_names_out(['color']))
# OR
# one_hot_df = pd.DataFrame(color_mod_encoding,␣
 ↪columns=['black','red','blue','yellow','orange','white'])
one_hot_df
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[48], line 1
----> 1 one_hot_df = pd.DataFrame(color_mod_encoding, columns=one_hot_encoding.
 ↪get_feature_names_out(['color']))
      2 # OR
      3 one_hot_df

File /opt/anaconda3/lib/python3.12/site-packages/pandas/core/frame.py:867, in␣
 ↪DataFrame.__init__(self, data, index, columns, dtype, copy)
    859         mgr = arrays_to_mgr(
    860             arrays,
    861             columns,
   (…)
    864             typ=manager,
    865         )
    866     else:
--> 867         mgr = ndarray_to_mgr(
    868             data,
    869             index,
    870             columns,
    871             dtype=dtype,
    872             copy=copy,
    873             typ=manager,
    874         )
    875 else:
    876     mgr = dict_to_mgr(
    877         {},
    878         index,
   (…)
    881         typ=manager,
    882     )

File /opt/anaconda3/lib/python3.12/site-packages/pandas/core/internals/
 ↪construction.py:336, in ndarray_to_mgr(values, index, columns, dtype, copy,␣
 ↪typ)
    331 # _prep_ndarraylike ensures that values.ndim == 2 at this point
    332 index, columns = _get_axes(
    333     values.shape[0], values.shape[1], index=index, columns=columns
    334 )
--> 336 _check_values_indices_shape_match(values, index, columns)
    338 if typ == "array":
    339     if issubclass(values.dtype.type, str):

File /opt/anaconda3/lib/python3.12/site-packages/pandas/core/internals/
 ↪construction.py:420, in _check_values_indices_shape_match(values, index,␣
 ↪columns)
    418 passed = values.shape
    419 implied = (len(index), len(columns))
--> 420 raise ValueError(f"Shape of passed values is {passed}, indices imply␣
 ↪{implied}")

ValueError: Shape of passed values is (10, 1), indices imply (10, 6)
```

[ ]:
```
new_df = pd.concat(df,one_hot_df,axis=1)
new_df
```

5 beer_servings.csv

[26]:
```
# continent - one hot
df_beer = pd.read_csv('beer-servings.csv')
df_beer
```

[26]:
```
     Unnamed: 0      country  beer_servings  spirit_servings  wine_servings  \
0             0  Afghanistan            0.0              0.0            0.0
1             1      Albania           89.0            132.0           54.0
2             2      Algeria           25.0              0.0           14.0
3             3      Andorra          245.0            138.0          312.0
4             4       Angola          217.0             57.0           45.0
..          …          …              …                …              …
188         188    Venezuela            NaN            100.0            3.0
189         189      Vietnam          111.0              2.0            1.0
190         190        Yemen            6.0              0.0            0.0
191         191       Zambia           32.0             19.0            4.0
192         192     Zimbabwe           64.0             18.0            4.0

     total_litres_of_pure_alcohol      continent
0                             0.0           Asia
1                             4.9         Europe
2                             0.7         Africa
3                            12.4         Europe
4                             5.9         Africa
..                            …            …
```

```
188                           7.7  South America
189                           2.0           Asia
190                           0.1           Asia
191                           2.5         Africa
192                           4.7         Africa

[193 rows x 7 columns]
```

[28]:

```
# country - label encoding
df_beer['country_encoding'] = label_encoding.fit_transform(df_beer['country'])
df_beer
```

[28]:

```
     Unnamed: 0      country  beer_servings  spirit_servings  wine_servings  \
0             0  Afghanistan            0.0              0.0            0.0
1             1      Albania           89.0            132.0           54.0
2             2      Algeria           25.0              0.0           14.0
3             3      Andorra          245.0            138.0          312.0
4             4       Angola          217.0             57.0           45.0
..          …          …            …              …            …
188         188    Venezuela            NaN            100.0            3.0
189         189      Vietnam          111.0              2.0            1.0
190         190        Yemen            6.0              0.0            0.0
191         191       Zambia           32.0             19.0            4.0
192         192     Zimbabwe           64.0             18.0            4.0

     total_litres_of_pure_alcohol      continent  country_encoding
0                             0.0           Asia                 0
1                             4.9         Europe                 1
2                             0.7         Africa                 2
3                            12.4         Europe                 3
4                             5.9         Africa                 4
..                          …            …                …
188                           7.7  South America               188
189                           2.0           Asia               189
190                           0.1           Asia               190
191                           2.5         Africa               191
192                           4.7         Africa               192

[193 rows x 8 columns]
```

[42]:

```
# continent - one hot encoding
continent_onehot_encoding = pd.
 ↪get_dummies(df_beer,columns=['continent'],prefix='cont',dtype='int',drop_first=True)
continent_onehot_encoding
```

[42]:

```
     Unnamed: 0      country  beer_servings  spirit_servings  wine_servings  \
0             0  Afghanistan            0.0              0.0            0.0
```

```
1             1      Albania           89.0            132.0           54.0
2             2      Algeria           25.0              0.0           14.0
3             3      Andorra          245.0            138.0          312.0
4             4       Angola          217.0             57.0           45.0
..          …          …            …              …            …
188         188    Venezuela            NaN            100.0            3.0
189         189      Vietnam          111.0              2.0            1.0
190         190        Yemen            6.0              0.0            0.0
191         191       Zambia           32.0             19.0            4.0
192         192     Zimbabwe           64.0             18.0            4.0

     total_litres_of_pure_alcohol  country_encoding  cont_Asia  cont_Europe  \
0                             0.0                 0          1            0
1                             4.9                 1          0            1
2                             0.7                 2          0            0
3                            12.4                 3          0            1
4                             5.9                 4          0            0
..                            …               …        …          …
188                           7.7               188          0            0
189                           2.0               189          1            0
190                           0.1               190          1            0
191                           2.5               191          0            0
192                           4.7               192          0            0

     cont_North America  cont_Oceania  cont_South America
0                     0             0                   0
1                     0             0                   0
2                     0             0                   0
3                     0             0                   0
4                     0             0                   0
..                  …           …                 …
188                   0             0                   1
189                   0             0                   0
190                   0             0                   0
191                   0             0                   0
192                   0             0                   0

[193 rows x 12 columns]
```

6 Scaling

7 Standard Scaling

[53]:
```
std_scaler = StandardScaler() # instance
df['price_standardised_scale'] = std_scaler.fit_transform(df[['price']])
df
```

[53]:
```
   id product   color    category    sales  rating  price  category_encoded  \
0   1   apple     red       fruit      low       5    100                 2
1   2  orange  orange       fruit     high       4    120                 2
2   3  banana  yellow       fruit  average       3     25                 2
3   4  carrot  orange   vegetable  average       2     15                 4
4   5  laptop   black  electronic      low       5  80000                 1
5   6   phone   white  electronic     high       4  20000                 1
6   7   shirt    blue       cloth     high       3    300                 0
7   8   pants   black       cloth     high       4    500                 0
8   9     pen     red  stationary      low       5     15                 3
9  10     box   black  stationary     high       4     50                 3

   product_encoded  sales_encoding  color_encoding  price_standardised_scale
0                0             0.0             1.0                 -0.416546
1                5             2.0             4.0                 -0.415714
2                1             1.0             3.0                 -0.419667
3                3             1.0             4.0                 -0.420083
4                4             0.0             0.0                  2.907505
5                8             2.0             5.0                  0.411346
6                9             2.0             2.0                 -0.408226
7                6             2.0             0.0                 -0.399905
8                7             0.0             1.0                 -0.420083
9                2             2.0             0.0                 -0.418627
```

[54]:
```
df['price'].describe()
```

[54]:
```
count       10.000000
mean     10112.500000
std      25337.151933
min         15.000000
25%         31.250000
50%        110.000000
75%        450.000000
max      80000.000000
Name: price, dtype: float64
```

[56]:
```
df['price_standardised_scale'].describe()
```

[56]:

```
count    1.000000e+01
mean     2.220446e-17
std      1.054093e+00
min     -4.200827e-01
25%     -4.194067e-01
50%     -4.161305e-01
75%     -4.019856e-01
max      2.907505e+00
Name: price_standardised_scale, dtype: float64
```

8 Min-Max Scaling

[57]:

```
norm_scaler = MinMaxScaler() # instance
# norm_scaler = MinMaxScaler(feature_reange=(0,1)) # instance
df['price_normalised_scale'] = norm_scaler.fit_transform(df[['price']])
df
```

[57]:

```
   id product   color    category    sales  rating  price  category_encoded  \
0   1   apple     red       fruit      low       5    100                 2
1   2  orange  orange       fruit     high       4    120                 2
2   3  banana  yellow       fruit  average       3     25                 2
3   4  carrot  orange   vegetable  average       2     15                 4
4   5  laptop   black  electronic      low       5  80000                 1
5   6   phone   white  electronic     high       4  20000                 1
6   7   shirt    blue       cloth     high       3    300                 0
7   8   pants   black       cloth     high       4    500                 0
8   9     pen     red  stationary      low       5     15                 3
9  10     box   black  stationary     high       4     50                 3

   product_encoded  sales_encoding  color_encoding  price_standardised_scale  \
0                0             0.0             1.0                 -0.416546
1                5             2.0             4.0                 -0.415714
2                1             1.0             3.0                 -0.419667
3                3             1.0             4.0                 -0.420083
4                4             0.0             0.0                  2.907505
5                8             2.0             5.0                  0.411346
6                9             2.0             2.0                 -0.408226
7                6             2.0             0.0                 -0.399905
8                7             0.0             1.0                 -0.420083
9                2             2.0             0.0                 -0.418627

   price_normalised_scale
0                0.001063
1                0.001313
2                0.000125
3                0.000000
```

```
4                 1.000000
5                 0.249859
6                 0.003563
7                 0.006064
8                 0.000000
9                 0.000438
```

[58]:
```
df['price'].describe()
```

[58]:
```
count       10.000000
mean     10112.500000
std      25337.151933
min         15.000000
25%         31.250000
50%        110.000000
75%        450.000000
max      80000.000000
Name: price, dtype: float64
```

[60]:
```
df['price_normalised_scale'].describe()
```

[60]:
```
count    10.000000
mean      0.126242
std       0.316774
min       0.000000
25%       0.000203
50%       0.001188
75%       0.005439
max       1.000000
Name: price_normalised_scale, dtype: float64
```

9 Correlation check

[67]:
```
# Read dataset
df_beer = pd.read_csv('beer-servings.csv')

# Deleting "Unnamed:0" column because it is useless
df_beer = df_beer.iloc[:,1:]

# to delete any duplicate rows
df_beer.drop_duplicates(inplace=True)

# Splitting numerical and categorical columns
num_df_beer = df_beer.select_dtypes(include="number")
cat_df_beer = df_beer.select_dtypes(include="object_")

num_cols = num_df_beer.columns.tolist()
```

```python
cat_cols = cat_df_beer.columns.tolist()

# Filling missing values for numerical columns
for col in num_cols:
    df_beer[col] = df_beer[col].fillna(df_beer[col].median())

# Filling missing values for categorical columns¶
for col in cat_cols:
    df_beer[col] = df_beer[col].fillna(df_beer[col].mode()[0])

def remove_outliers(df, column_name):
    q1 = df[column_name].quantile(0.25)
    q3 = df[column_name].quantile(0.75)
    iqr = q3-q1
    lower_bound = q1 - 1.5*iqr
    upper_bound = q3 + 1.5*iqr
    df[column_name] = df[column_name].clip(upper=upper_bound)
    df[column_name] = df[column_name].clip(lower=lower_bound)
    return df[column_name]

for col in num_cols:
    df_beer[col] = remove_outliers(df_beer, col)

df_beer
```

```
[67]:          country  beer_servings  spirit_servings  wine_servings  \
      0    Afghanistan            0.0              0.0            0.0
      1        Albania           89.0            132.0           54.0
      2        Algeria           25.0              0.0           14.0
      3        Andorra          245.0            138.0          146.0
      4         Angola          217.0             57.0           45.0
      ..           …              …                …             …
      188    Venezuela           76.0            100.0            3.0
      189      Vietnam          111.0              2.0            1.0
      190        Yemen            6.0              0.0            0.0
      191       Zambia           32.0             19.0            4.0
      192     Zimbabwe           64.0             18.0            4.0

           total_litres_of_pure_alcohol      continent
      0                             0.0           Asia
      1                             4.9         Europe
      2                             0.7         Africa
      3                            12.4         Europe
      4                             5.9         Africa
      ..                             …              …
      188                           7.7  South America
      189                           2.0           Asia
```

```
190                            0.1          Asia
191                            2.5        Africa
192                            4.7        Africa

[193 rows x 6 columns]
```

[68]:

```
# Make copies of the dataset to apply std and norm scaling
df_beer_std_copy = df_beer
df_beer_norm_copy = df_beer
```

[74]:

```
std_df = std_scaler.fit_transform(df_beer_std_copy[num_cols])
pd.DataFrame(std_df)
```

[74]:

```
            0         1         2         3
0   -1.056880 -0.971267 -0.723136 -1.264356
1   -0.151713  0.608759  0.302520  0.042922
2   -0.802619 -0.971267 -0.457225 -1.077602
3    1.434871  0.680578  2.049934  2.043856
4    1.150100 -0.288983  0.131578  0.309713
..        …         …         …         …
188 -0.283928  0.225722 -0.666155  0.789937
189  0.072036 -0.947327 -0.704142 -0.730773
190 -0.995857 -0.971267 -0.723136 -1.237677
191 -0.731427 -0.743839 -0.647161 -0.597378
192 -0.405973 -0.755809 -0.647161 -0.010437

[193 rows x 4 columns]
```

[75]:

```
norm_df = norm_scaler.fit_transform(df_beer_norm_copy[num_cols])
pd.DataFrame(norm_df)
```

[75]:

```
            0       1         2         3
0    0.000000  0.0000  0.000000  0.000000
1    0.236702  0.4224  0.369863  0.340278
2    0.066489  0.0000  0.095890  0.048611
3    0.651596  0.4416  1.000000  0.861111
4    0.577128  0.1824  0.308219  0.409722
..        …       …         …         …
188  0.202128  0.3200  0.020548  0.534722
189  0.295213  0.0064  0.006849  0.138889
190  0.015957  0.0000  0.000000  0.006944
191  0.085106  0.0608  0.027397  0.173611
192  0.170213  0.0576  0.027397  0.326389

[193 rows x 4 columns]
```

[79]:

```
# Correlation should not be affected even after all the preprocessing is done
print("Correlation of original df = \n", pd.DataFrame(df_beer[num_cols]).corr())

print("\n\nCorrelation of standardised df = \n", pd.DataFrame(std_df).corr())

print("\n\nCorrelation of normalised df = \n", pd.DataFrame(norm_df).corr())
```

```
Correlation of original df =
                               beer_servings  spirit_servings  wine_servings  \
beer_servings                      1.000000         0.473518       0.617509
spirit_servings                    0.473518         1.000000       0.280068
wine_servings                      0.617509         0.280068       1.000000
total_litres_of_pure_alcohol       0.829418         0.659014       0.716568

                              total_litres_of_pure_alcohol
beer_servings                                     0.829418
spirit_servings                                   0.659014
wine_servings                                     0.716568
total_litres_of_pure_alcohol                      1.000000


Correlation of standardised df =
           0         1         2         3
0  1.000000  0.473518  0.617509  0.829418
1  0.473518  1.000000  0.280068  0.659014
2  0.617509  0.280068  1.000000  0.716568
3  0.829418  0.659014  0.716568  1.000000


Correlation of normalised df =
           0         1         2         3
0  1.000000  0.473518  0.617509  0.829418
1  0.473518  1.000000  0.280068  0.659014
2  0.617509  0.280068  1.000000  0.716568
3  0.829418  0.659014  0.716568  1.000000
```

[ ]:

```

```

[ ]:

```

```