Solved Examples

In [13]:

```
# Program for Filling Missing Values
import pandas as pd
import numpy as np
#Creating a DataFrame with Missing Values
df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f','h'],
                  columns=['C01', 'C02', 'C03'])
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print("\n Reindexed Data Values")
print("-------------------------")
print(df)
#Method 1 - Filling Every Missing Values with 0
print("\n\n Every Missing Value Replaced with '0':")
print("-------------------------------------------")
print(df.fillna(0))
#Method 2 - Dropping Rows Having Missing Values
print("\n\n Dropping Rows with Missing Values:")
print("----------------------------------------")
print(df.dropna())
```

```
 Reindexed Data Values
-------------------------
        C01       C02       C03
a -0.146488 -1.318994  2.254398
b       NaN       NaN       NaN
c  0.230573 -0.581657 -2.077945
d       NaN       NaN       NaN
e  0.205695  0.961457 -0.224647
f -0.699122 -0.266711 -0.187175
g       NaN       NaN       NaN
h -0.206159 -0.488579 -0.069995


 Every Missing Value Replaced with '0':
-------------------------------------------
        C01       C02       C03
a -0.146488 -1.318994  2.254398
b  0.000000  0.000000  0.000000
c  0.230573 -0.581657 -2.077945
d  0.000000  0.000000  0.000000
e  0.205695  0.961457 -0.224647
f -0.699122 -0.266711 -0.187175
g  0.000000  0.000000  0.000000
h -0.206159 -0.488579 -0.069995


 Dropping Rows with Missing Values:
----------------------------------------
        C01       C02       C03
a -0.146488 -1.318994  2.254398
c  0.230573 -0.581657 -2.077945
e  0.205695  0.961457 -0.224647
f -0.699122 -0.266711 -0.187175
h -0.206159 -0.488579 -0.069995
```

In [2]:

```
# Program for Data Transformation
import pandas as pd
import numpy as np
from sklearn import preprocessing
import scipy.stats as s
#Creating a DataFrame
d = {'C01':[1,3,7,4],'C02':[12,2,7,1],'C03':[22,34,-11,9]}
df2 = pd.DataFrame(d)
print("\n ORIGINAL DATA VALUES")
print("------------------------")
print(df2)
#Method 1: Rescaling Data
print("\n\n Data Scaled Between 0 to 1")
data_scaler = preprocessing.MinMaxScaler(feature_range = (0, 1))
data_scaled = data_scaler.fit_transform(df2)
print("\n Min Max Scaled Data ")
print("-----------------------")
print(data_scaled.round(2))
#Method 2: Normalization rescales such that sum of each row is 1.
dn = preprocessing.normalize(df2, norm = 'l1')
print("\n L1 Normalized Data ")
print(" ----------------------")
print(dn.round(2))
#Method 3: Binarize Data (Make Binary)
data_binarized = preprocessing.Binarizer(threshold=5).transform(df2)
print("\n Binarized data ")
print(" -----------------")
print(data_binarized)
#Method 4: Standardizing Data
print("\n Standardizing Data ")
print("-----------------------")
X_train = np.array([[ 1., -1., 2.],[ 2., 0., 0.],[ 0., 1., -1.]])
print(" Orginal Data \n", X_train)
print("\n Initial Mean : ", s.tmean(X_train).round(2))
print(" Initial Standard Deviation : ",round(X_train.std(),2))
X_scaled = preprocessing.scale(X_train)
X_scaled.mean(axis=0)
X_scaled.std(axis=0)
print("\n Standardized Data \n", X_scaled.round(2))
print("\n Scaled Mean : ",s.tmean(X_scaled).round(2))
print(" Scaled Standard Deviation : ",round(X_scaled.std(),2))
```

```
 ORIGINAL DATA VALUES
------------------------
   C01  C02  C03
0    1   12   22
1    3    2   34
2    7    7  -11
3    4    1    9


 Data Scaled Between 0 to 1

 Min Max Scaled Data
-----------------------
[[0.   1.   0.73]
 [0.33 0.09 1.  ]
 [1.   0.55 0.  ]
 [0.5  0.   0.44]]

 L1 Normalized Data
```

```
 ----------------------
[[ 0.03  0.34  0.63]
 [ 0.08  0.05  0.87]
 [ 0.28  0.28 -0.44]
 [ 0.29  0.07  0.64]]

 Binarized data
 -----------------
[[0 1 1]
 [0 0 1]
 [1 1 0]
 [0 0 1]]

 Standardizing Data
----------------------
 Orginal Data
 [[ 1. -1.  2.]
 [ 2.  0.  0.]
 [ 0.  1. -1.]]

 Initial Mean :  0.44
 Initial Standard Deviation :  1.07

 Standardized Data
 [[ 0.   -1.22  1.34]
 [ 1.22  0.   -0.27]
 [-1.22  1.22 -1.07]]

 Scaled Mean :  0.0
```

```
E:\anoconda\lib\site-packages\sklearn\base.py:443: UserWarning: X has featu
re names, but Binarizer was fitted without feature names
  warnings.warn(
```

Create own dataset and do simple preprocessing Dataset Name: Data.CSV (save following data in Excel and save it with .CSV extension) Country, Age, Salary, Purchased France, 44, 72000, No Spain, 27, 48000, Yes Germany, 30, 54000, No Spain, 38, 61000, No Germany, 40, , Yes France, 35, 58000, Yes Spain, 52000, No France, 48, 79000, Yes Germany, 50, 83000, No France, 37, 67000, Yes *Above dataset is also available at: https://github.com/suneet10/DataPreprocessing/blob/main/Data.csv

Set A

In [3]:

```
#1.Import Dataset and do the followings:
#a) Describing the dataset
#b) Shape of the dataset
#c) Display first 3 rows from dataset
import pandas as pd
df = pd.read_csv("country.csv")
print(df)
print("\n Describing the dataset ")
print(df.describe())
print("\nShape of the dataset")
print(df.shape)
print("\n Display first 3 rows from dataset ")
print(df.head(3))
```

```
   Country   Age   Salary Purchased
0   France  44.0  72000.0        No
```

```
1    Spain  27.0  48000.0       Yes
2  Germany  30.0  54000.0        No
3    Spain  38.0  61000.0        No
4  Germany  40.0      NaN       Yes
5   France  35.0  58000.0       Yes
6    Spain   NaN  52000.0        No
7   France  48.0  79000.0       Yes
8  Germany  50.0  83000.0        No
9   France  37.0  67000.0       Yes

 Describing the dataset
             Age        Salary
count   9.000000      9.000000
mean   38.777778  63777.777778
std     7.693793  12265.579662
min    27.000000  48000.000000
25%    35.000000  54000.000000
50%    38.000000  61000.000000
75%    44.000000  72000.000000
max    50.000000  83000.000000

Shape of the dataset
(10, 4)

 Display first 3 rows from dataset
  Country   Age   Salary Purchased
0  France  44.0  72000.0        No
1   Spain  27.0  48000.0       Yes
```

In [4]:

```
#2.Handling Missing Value: a) Replace missing value of salary,age column wi
import pandas as pd
df = pd.read_csv("country.csv")
df['Salary']=df['Salary'].fillna(df['Salary'].mode()[0])
df['Age']=df['Age'].fillna(df['Age'].mode()[0])
df
```

Out[4]:

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	48000.0	Yes
5	France	35.0	58000.0	Yes
6	Spain	27.0	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

In [5]:

```
#3. Data.csv have two categorical column (the country column, and the purch
#a. Apply OneHot coding on Country column.
#b. Apply Label encoding on purchased column

import pandas as pd
from sklearn.preprocessing import*
df = pd.read_csv("country.csv")
enc=OneHotEncoder(handle_unknown='ignore')
enc=pd.DataFrame(enc.fit_transform(df[['Country']]).toarray())
df=df.join(enc)
print('Applying OneHotEncoding on Country')
print(df)
```

```
Applying OneHotEncoding on Country
   Country   Age   Salary Purchased    0    1    2
0   France  44.0  72000.0        No  1.0  0.0  0.0
1    Spain  27.0  48000.0       Yes  0.0  0.0  1.0
2  Germany  30.0  54000.0        No  0.0  1.0  0.0
3    Spain  38.0  61000.0        No  0.0  0.0  1.0
4  Germany  40.0      NaN       Yes  0.0  1.0  0.0
5   France  35.0  58000.0       Yes  1.0  0.0  0.0
6    Spain   NaN  52000.0        No  0.0  0.0  1.0
7   France  48.0  79000.0       Yes  1.0  0.0  0.0
8  Germany  50.0  83000.0        No  0.0  1.0  0.0
9   France  37.0  67000.0       Yes  1.0  0.0  0.0
```

Set B

Import standard dataset and Use Transformation Techniques Dataset Name: winequality-red.csv Dataset Link: http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv

In [6]:

```
#1. Import Dataset from above link.
import pandas as pd
from sklearn.preprocessing import *
df = pd.read_csv("winequality-red.csv")
df
```

Out[6]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcol
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
...	...	...	...	...	...	...	...	...	...	...	
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	1
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	1
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	1
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	1
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	1

1599 rows × 12 columns

In [7]:

```
#2. Rescaling: Normalised the dataset using MinMaxScaler class
from numpy import asarray
from sklearn.preprocessing import MinMaxScaler
# define data
data = pd.read_csv("winequality-red.csv")
print(data)
# define min max scaler
scaler = MinMaxScaler()
# transform data
scaled = scaler.fit_transform(data)
print(scaled)
```

```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorid
es  \
0               7.4             0.700         0.00             1.9      0.0
76
1               7.8             0.880         0.00             2.6      0.0
98
2               7.8             0.760         0.04             2.3      0.0
92
3              11.2             0.280         0.56             1.9      0.0
75
4               7.4             0.700         0.00             1.9      0.0
76
...             ...               ...          ...             ...
...
1594            6.2             0.600         0.08             2.0      0.0
90
1595            5.9             0.550         0.10             2.2      0.0
62
1596            6.3             0.510         0.13             2.3      0.0
76
1597            5.9             0.645         0.12             2.0      0.0
75
1598            6.0             0.310         0.47             3.6      0.0
67

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates
\
0                    11.0                  34.0  0.99780  3.51       0.56
1                    25.0                  67.0  0.99680  3.20       0.68
2                    15.0                  54.0  0.99700  3.26       0.65
3                    17.0                  60.0  0.99800  3.16       0.58
4                    11.0                  34.0  0.99780  3.51       0.56
...                   ...                   ...      ...   ...        ...
1594                 32.0                  44.0  0.99490  3.45       0.58
1595                 39.0                  51.0  0.99512  3.52       0.76
1596                 29.0                  40.0  0.99574  3.42       0.75
1597                 32.0                  44.0  0.99547  3.57       0.71
1598                 18.0                  42.0  0.99549  3.39       0.66

      alcohol  quality
0         9.4        5
1         9.8        5
2         9.8        5
3         9.8        6
4         9.4        5
...       ...      ...
1594     10.5        5
```

```
1595     11.2        6
1596     11.0        6
1597     10.2        5
1598     11.0        6

[1599 rows x 12 columns]
[[0.24778761 0.39726027 0.         ... 0.13772455 0.15384615 0.4       ]
 [0.28318584 0.52054795 0.         ... 0.20958084 0.21538462 0.4       ]
 [0.28318584 0.43835616 0.04       ... 0.19161677 0.21538462 0.4       ]
 ...
 [0.15044248 0.26712329 0.13       ... 0.25149701 0.4        0.6       ]
 [0.11504425 0.35958904 0.12       ... 0.22754491 0.27692308 0.4       ]
 [0.12389381 0.13013699 0.47       ... 0.19760479 0.4        0.6       ]]
```

In [8]:

```
#3.Standardizing Data (transform them into a standard Gaussian distributior
import pandas as pd
import numpy as np
from sklearn import preprocessing
import scipy.stats as s
df = pd.read_csv("winequality-red.csv")
print('Standarlizing Data')
x_train=df
print('Initial mean:',s.tmean(x_train).round(2))
print('Initial SD:',round(x_train.std(),2))
x_scaled=preprocessing.scale(x_train)
x_scaled.mean(axis=0)
x_scaled.std(axis=0)
print('Standarizing Data:',x_scaled.round(2))
print('Standarizing mean:',s.tmean(x_scaled).round(2))
print('Standarizing SD:',round(x_scaled.std(),2))
```

```
Standarlizing Data
Initial mean: 7.93
Initial SD: fixed acidity            1.74
volatile acidity         0.18
citric acid              0.19
residual sugar           1.41
chlorides                0.05
free sulfur dioxide     10.46
total sulfur dioxide    32.90
density                  0.00
pH                       0.15
sulphates                0.17
alcohol                  1.07
quality                  0.81
dtype: float64
Standarizing Data: [[-0.53  0.96 -1.39 ... -0.58 -0.96 -0.79]
 [-0.3   1.97 -1.39 ...  0.13 -0.58 -0.79]
 [-0.3   1.3  -1.19 ... -0.05 -0.58 -0.79]
 ...
 [-1.16 -0.1  -0.72 ...  0.54  0.54  0.45]
 [-1.39  0.65 -0.78 ...  0.31 -0.21 -0.79]
 [-1.33 -1.22  1.02 ...  0.01  0.54  0.45]]
Standarizing mean: -0.0
Standarizing SD: 1.0
```

In [9]:

```
#4. Normalizing Data ( rescale each observation to a length of 1 (a unit no
import pandas as pd
import numpy as np
from sklearn import preprocessing
import scipy.stats as s
df = pd.read_csv("winequality-red.csv")
dn=preprocessing.normalize(df)
print("\n L1 Normalized Data")
print("--------------------")
print(dn.round(2))
```

```
 L1 Normalized Data
--------------------
[[0.19 0.02 0.   ... 0.01 0.25 0.13]
 [0.11 0.01 0.   ... 0.01 0.13 0.07]
 [0.13 0.01 0.   ... 0.01 0.17 0.09]
 ...
 [0.12 0.01 0.   ... 0.01 0.21 0.12]
 [0.11 0.01 0.   ... 0.01 0.18 0.09]
 [0.12 0.01 0.01 ... 0.01 0.23 0.12]]
```

In [10]:

```
#5. Binarizing Data using we use the Binarizer class (Using a binary thresh
#data by marking the values above it 1 and those equal to or below it, 0)
import pandas as pd
import numpy as np
from sklearn import preprocessing
import scipy.stats as s
df = pd.read_csv("winequality-red.csv")
d_binarized=preprocessing.Binarizer(threshold=5).transform(df)
print("\n Binarized Data")
print("--------------------")
print(d_binarized)
```

```
 Binarized Data
--------------------
[[1. 0. 0. ... 0. 1. 0.]
 [1. 0. 0. ... 0. 1. 0.]
 [1. 0. 0. ... 0. 1. 0.]
 ...
 [1. 0. 0. ... 0. 1. 1.]
 [1. 0. 0. ... 0. 1. 0.]
 [1. 0. 0. ... 0. 1. 1.]]
E:\anoconda\lib\site-packages\sklearn\base.py:443: UserWarning: X has featu
re names, but Binarizer was fitted without feature names
  warnings.warn(
```

Set C

Import dataset and perform Discretization of Continuous Data Dataset name: Student_bucketing.csv Dataset link: https://github.com/TrainingByPackt/Data-Science-with-Python/blob/master/Chapter01/Data/Student_bucketing.csv The dataset consists of student details such as Student_id, Age, Grade, Employed, and marks. Write a program in python to perform following task

In [11]:

```
#1) Write python code to import the required libraries and load the dataset
import pandas as pd
df = pd.read_csv("Student_bucketing.csv")
df
```

Out[11]:

	Student_id	Age	Grade	Employed	marks
0	1	19	1st Class	yes	29
1	2	20	2nd Class	no	41
2	3	18	1st Class	no	57
3	4	21	2nd Class	no	29
4	5	19	1st Class	no	57
...	...	...	...	...	...
227	228	21	1st Class	no	42
228	229	20	2nd Class	no	47
229	230	20	3rd Class	yes	21
230	231	19	1st Class	yes	64
231	232	20	3rd Class	yes	30

232 rows × 5 columns

In [12]:

```
#2) Display the first five rows of the dataframe.
import pandas as pd
df = pd.read_csv("Student_bucketing.csv")
df.head(5)
```

Out[12]:

	Student_id	Age	Grade	Employed	marks
0	1	19	1st Class	yes	29
1	2	20	2nd Class	no	41
2	3	18	1st Class	no	57
3	4	21	2nd Class	no	29
4	5	19	1st Class	no	57