26/8/21

DWDM

Tutorial - 2

Sahil Bondre - U18CO021

Q1 Needs of Preprocessing Data:

i Incomplete Data: Lacking attribute values, lacking attributes of interest or containing errors or outliers.

Eg: 'salary' = NA

ii Noisy: Data containing errors or outliers.

Eg: 'age' = -40 ; 'salary' = True

iii Inconsistent: Data containing discrepencies in codes & data

Eg: PersonA.age = 45 & PersonB.dob = 1997
CarA.state = 'GUJARAT' &
CarB.state = 'GJ'

Q2 Factors compromising data quality:

i Accuracy
ii Completness
iii Consistency
iv Believability
v Interpretability
vi Accessibility
vii Timeliness.

Q3 Tasks in Data Preprocessing

1 Data Cleaning: This task involves filling of missing values, smoothing or removing noisy data & outliers along with inconsistencies.

Eg: Data collected from sensors can contain no noise which needs to be cleaned.

2 Data Integration: This task involves integrating data from various sources.

Eg: Data can be present in sensors, databases, third party APIs which needs to be combined

3 Data Transformation: This involves normalisation & aggregation of data according to the needs of dataset

Eg: Many analytical modes perform better with normalised data

4 Data Reduction: In this steps, the irrelevant records, attributes & dimensions can be removed

Eg: color attribute in housing dataset will not have much effect on final price of the house & can be removed

5 Data Discretization: Numerical attributes are replaced with normal one

Eg: Exact longitude & latitude coordinates can be replaced by binning into regions.

Q4 Data Cleaning Methods:

1 Listwise Deletion: Entire record is excluded if even one attribute is missing

2 Pairwise Deletion: Pairwise columns are taken & only if attributes from these columns are absent then the record is excluded

3 Single Value Imputation: Missing value is replaced by a single value

4 Hot Deck Imputation: Find all samples similar to other variables then randomly choose one of their values

5 Cold Deck Imputation: Systematically choose the values from an individual with similar values on other variables

6 KNN based Imputation : Finding nearest cluster using KNN & using the value of the cluster

7 Regression Based Imputation : Regression is used to predict the missing value

8 Multiple Imputation : Imputation technique that assigns several imputed values to each missing value & take mean of the result

Q5

i Mean : It is the distributive measure of data

$$\mu = \frac{\sum x_i}{N}$$

Weighted Mean : $\bar{x} = \frac{\sum \omega_i x_i}{\sum \omega_i}$

Eg: $S = \{1, 2, 2, 2, 3, 7, 8\}$

$$\mu = \frac{\sum s_i}{n} = \frac{25}{7} = 3.57$$

ii Median : It is the midpoint of the list when the distribution is in sorted order

Eg: $S = \{1, 2, 2, 3, 7, 8\} \Rightarrow m = \frac{2+3}{2} = 2.5$

iii. **Mode**: It is defined as the most frequently occuring score

Eg: $S = \{1, 2, 2, 2, 3, 7, 8, 8\}$

$M = 2$

DWDM Tutorial 2

U18CO021: SAHIL BONDRE

6. Analyse Pre-Processing techniques which were discussed in class

0.1 Importing Libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```
from sklearn.datasets import load_boston
```

0.2 Load Data

[3]:
```
data = load_boston()
df = pd.DataFrame(np.array(data.data), columns=data.feature_names)
# Add random NA Values for analysis
df = df.mask(np.random.random(df.shape) < 0.01)
df["PRICE"] = data.target
df.describe()
```

[3]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	\
count	495.000000	500.00000	502.000000	499.00000	497.000000	501.000000	
mean	3.570719	11.25600	11.168426	0.07014	0.553887	6.283409	
std	8.419298	23.23515	6.875861	0.25564	0.115157	0.704384	
min	0.009060	0.00000	0.460000	0.00000	0.385000	3.561000	
25%	0.082325	0.00000	5.190000	0.00000	0.449000	5.885000	
50%	0.253560	0.00000	9.690000	0.00000	0.538000	6.208000	
75%	3.685665	12.50000	18.100000	0.00000	0.624000	6.618000	
max	88.976200	100.00000	27.740000	1.00000	0.871000	8.780000	

	AGE	DIS	RAD	TAX	PTRATIO	B	\
count	499.000000	504.000000	503.000000	501.000000	499.000000	502.000000	
mean	68.586373	3.787333	9.542744	408.413174	18.460521	356.520657	
std	28.136710	2.098040	8.702341	168.707590	2.161934	91.632485	
min	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	
25%	45.050000	2.097050	4.000000	279.000000	17.400000	375.607500	
50%	77.700000	3.207450	5.000000	330.000000	19.100000	391.475000	
75%	94.050000	5.141475	24.000000	666.000000	20.200000	396.237500	
max	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	

```
             LSTAT       PRICE
count  504.000000  506.000000
mean    12.660179   22.532806
std      7.152720    9.197104
min      1.730000    5.000000
25%      6.927500   17.025000
50%     11.360000   21.200000
75%     16.992500   25.000000
max     37.970000   50.000000
```

[4]:
```
df.isna().sum()
```

[4]:
```
CRIM       11
ZN          6
INDUS       4
CHAS        7
NOX         9
RM          5
AGE         7
DIS         2
RAD         3
TAX         5
PTRATIO     7
B           4
LSTAT       2
PRICE       0
dtype: int64
```

0.3 Data Cleaning using Imputation

[5]:
```
from sklearn.impute import KNNImputer, SimpleImputer

simple_imputer = SimpleImputer(missing_values=np.NAN)
simple_df = pd.DataFrame(simple_imputer.fit_transform(df))
simple_df.columns = df.columns
simple_df.index = df.index
print(simple_df.isna().values.any())
simple_df.describe()
```

```
False
```

[5]:
```
             CRIM          ZN       INDUS        CHAS         NOX          RM  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     3.570719   11.256000   11.168426    0.070140    0.553887    6.283409
std      8.327098   23.096707    6.848575    0.253862    0.114126    0.700888
min      0.009060    0.000000    0.460000    0.000000    0.385000    3.561000
```

```
25%      0.083235    0.000000    5.190000    0.000000    0.450000    5.887250
50%      0.268880    0.000000    9.690000    0.000000    0.538000    6.210000
75%      3.570719   12.500000   18.100000    0.000000    0.624000    6.613500
max     88.976200  100.000000   27.740000    1.000000    0.871000    8.780000

              AGE         DIS         RAD         TAX     PTRATIO           B  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean    68.586373    3.787333    9.542744  408.413174   18.460521  356.520657
std     27.941022    2.093881    8.676454  167.870326    2.146898   91.268862
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000
25%     45.450000    2.100175    4.000000  279.250000   17.400000  374.835000
50%     76.800000    3.215700    5.000000  330.000000   19.000000  391.385000
75%     93.900000    5.118000   24.000000  666.000000   20.200000  396.225000
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000

            LSTAT       PRICE
count  506.000000  506.000000
mean    12.660179   22.532806
std      7.138542    9.197104
min      1.730000    5.000000
25%      6.950000   17.025000
50%     11.395000   21.200000
75%     16.955000   25.000000
max     37.970000   50.000000
```

[6]:
```
knn_imputer = KNNImputer(missing_values=np.NAN)
knn_df = pd.DataFrame(knn_imputer.fit_transform(df))
knn_df.columns = df.columns
knn_df.index = df.index
print(knn_df.isna().values.any())
knn_df.describe()
```

```
False
```

[6]:
```
             CRIM          ZN       INDUS        CHAS         NOX          RM  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     3.537289   11.323320   11.140123    0.069170    0.554318    6.286315
std      8.344714   23.154645    6.857791    0.253994    0.114938    0.702649
min      0.009060    0.000000    0.460000    0.000000    0.385000    3.561000
25%      0.082268    0.000000    5.190000    0.000000    0.449000    5.887250
50%      0.256510    0.000000    9.690000    0.000000    0.538000    6.209000
75%      3.677083   12.500000   18.100000    0.000000    0.624000    6.623500
max     88.976200  100.000000   27.740000    1.000000    0.871000    8.780000

              AGE         DIS         RAD         TAX     PTRATIO           B  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean    68.629605    3.786801    9.549407  408.188933   18.458261  356.751510
std     28.051530    2.095925    8.707268  168.337003    2.161832   91.307441
```

```
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000
25%     45.175000    2.100175    4.000000  279.000000   17.400000  375.607500
50%     77.500000    3.207450    5.000000  330.000000   19.050000  391.440000
75%     94.075000    5.118000   24.000000  666.000000   20.200000  396.225000
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000

            LSTAT       PRICE
count  506.000000  506.000000
mean    12.654273   22.532806
std      7.139641    9.197104
min      1.730000    5.000000
25%      6.950000   17.025000
50%     11.360000   21.200000
75%     16.955000   25.000000
max     37.970000   50.000000
```

0.4 Normalisation

[7]:
```
from sklearn import preprocessing
```

[8]:
```
df = knn_df
min_max_scalar = preprocessing.MinMaxScaler()
min_max_df = min_max_scalar.fit_transform(df)
min_max_df = pd.DataFrame(min_max_df)
min_max_df.columns = df.columns
min_max_df.describe()
```

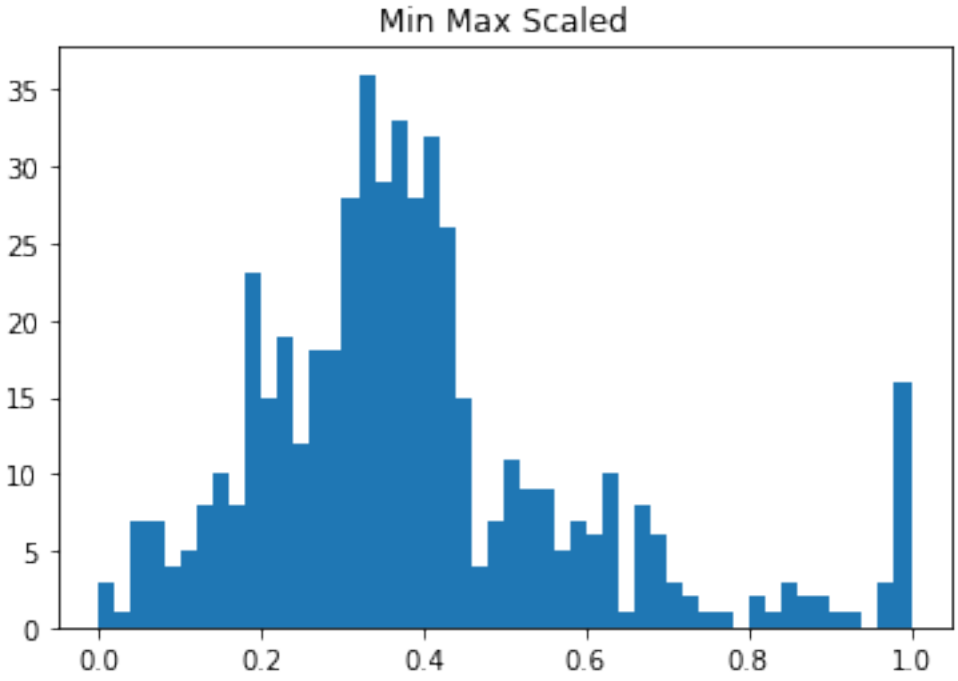
[8]:
```
             CRIM          ZN       INDUS        CHAS         NOX          RM  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     0.039658    0.113233    0.391500    0.069170    0.348392    0.522191
std      0.093795    0.231546    0.251385    0.253994    0.236499    0.134633
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000823    0.000000    0.173387    0.000000    0.131687    0.445727
50%      0.002781    0.000000    0.338343    0.000000    0.314815    0.507377
75%      0.041229    0.125000    0.646628    0.000000    0.491770    0.586798
max      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000

              AGE         DIS         RAD         TAX     PTRATIO           B  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     0.676927    0.241632    0.371713    0.422116    0.623219    0.898763
std      0.288893    0.190592    0.378577    0.321254    0.229982    0.230237
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%      0.435376    0.088259    0.130435    0.175573    0.510638    0.946310
50%      0.768280    0.188949    0.173913    0.272901    0.686170    0.986232
75%      0.938980    0.362684    1.000000    0.914122    0.808511    0.998298
max      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000
```

```
             LSTAT       PRICE
count  506.000000  506.000000
mean     0.301442    0.389618
std      0.197010    0.204380
min      0.000000    0.000000
25%      0.144040    0.267222
50%      0.265728    0.360000
75%      0.420116    0.444444
max      1.000000    1.000000
```

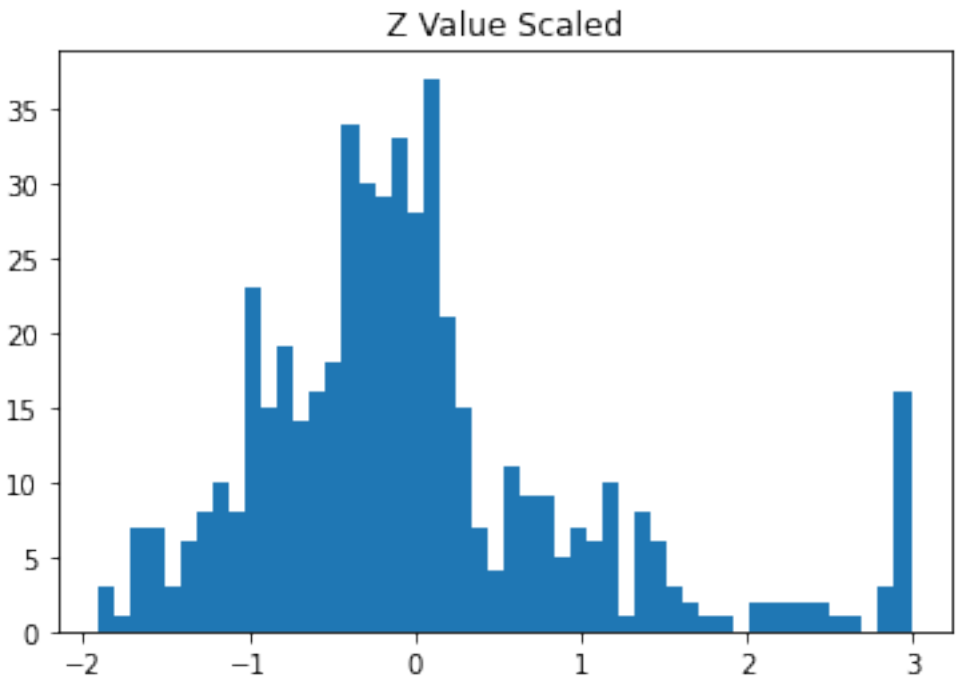
[9]:
```
df = knn_df
z_scalar = preprocessing.StandardScaler()
z_df = z_scalar.fit_transform(df)
z_df = pd.DataFrame(z_df)
z_df.columns = df.columns
z_df.describe()
```

[9]:
```
               CRIM            ZN         INDUS          CHAS           NOX  \
count  5.060000e+02  5.060000e+02  5.060000e+02  5.060000e+02  5.060000e+02
mean  -3.452443e-16  3.920886e-16  5.356936e-16 -3.100287e-16  4.120551e-16
std    1.000990e+00  1.000990e+00  1.000990e+00  1.000990e+00  1.000990e+00
min   -4.232285e-01 -4.895141e-01 -1.558912e+00 -2.725986e-01 -1.474580e+00
25%   -4.144469e-01 -4.895141e-01 -8.685028e-01 -2.725986e-01 -9.172094e-01
50%   -3.935457e-01 -4.895141e-01 -2.116655e-01 -2.725986e-01 -1.421155e-01
75%    1.676888e-02  5.086860e-02  1.015891e+00 -2.725986e-01  6.068517e-01
max    1.024882e+01  3.833548e+00  2.422982e+00  3.668398e+00  2.757955e+00

                 RM           AGE           DIS           RAD           TAX  \
count  5.060000e+02  5.060000e+02  5.060000e+02  5.060000e+02  5.060000e+02
mean   3.342737e-16 -8.096290e-17  4.348739e-16  2.698763e-16 -9.500525e-16
std    1.000990e+00  1.000990e+00  1.000990e+00  1.000990e+00  1.000990e+00
min   -3.882468e+00 -2.345492e+00 -1.269048e+00 -9.828418e-01 -1.315265e+00
25%   -5.685062e-01 -8.369531e-01 -8.055128e-01 -6.379611e-01 -7.682017e-01
50%   -1.101431e-01  3.165308e-01 -2.766912e-01 -5.230009e-01 -4.649382e-01
75%    4.803512e-01  9.079924e-01  6.357653e-01  1.661244e+00  1.533033e+00
max    3.552488e+00  1.119420e+00  3.982943e+00  1.661244e+00  1.800619e+00

             PTRATIO             B         LSTAT         PRICE
count  5.060000e+02  5.060000e+02  5.060000e+02  5.060000e+02
mean  -1.925557e-15 -2.049305e-16  6.209350e-17 -4.247810e-16
std    1.000990e+00  1.000990e+00  1.000990e+00  1.000990e+00
min   -2.712541e+00 -3.907505e+00 -1.531601e+00 -1.908226e+00
25%   -4.900047e-01  2.067154e-01 -7.997486e-01 -5.994557e-01
50%    2.739920e-01  3.802846e-01 -1.814592e-01 -1.450593e-01
75%    8.064745e-01  4.327419e-01  6.029692e-01  2.685231e-01
max    1.639925e+00  4.401418e-01  3.549307e+00  2.989460e+00
```

[10]:
```
plt.hist(min_max_df.PRICE, bins=50);
plt.title("Min Max Scaled");
```



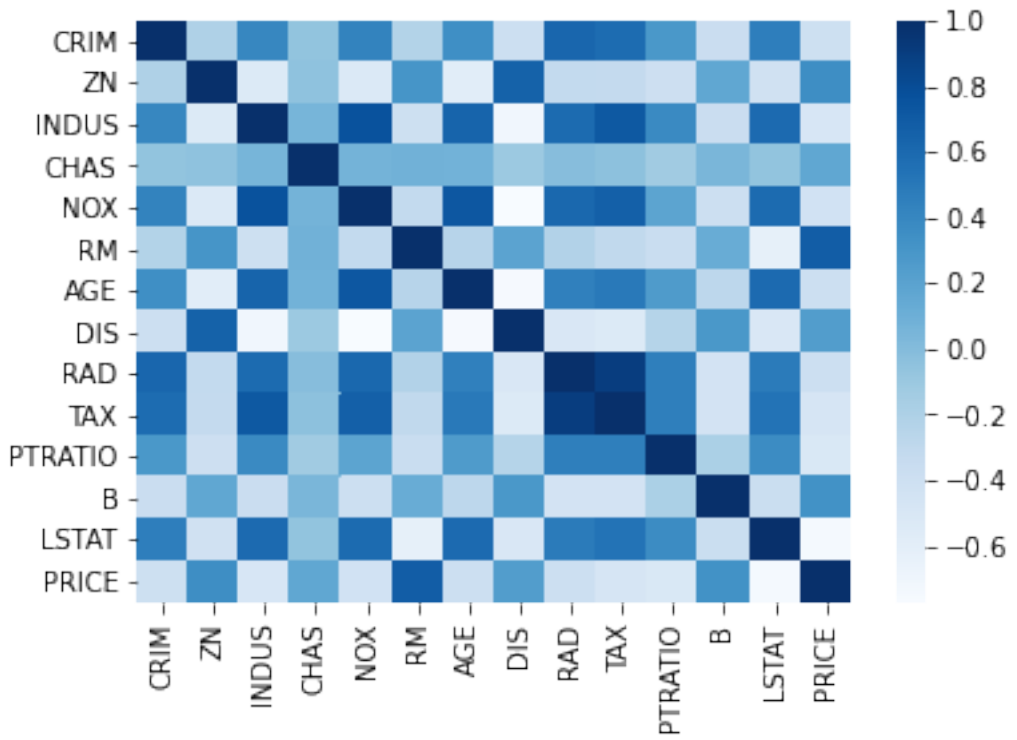
[11]:
```
plt.hist(z_df.PRICE, bins=50);
plt.title("Z Value Scaled");
```



0.5 Data Reduction

[12]:
```
import seaborn as sns

corr = df.corr()
sns.heatmap(corr, cmap="Blues")
df.columns
```

[12]:
```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'PRICE'],
      dtype='object')
```



LSTAT contributes very less to the PRICE so can be eliminated

[13]:
```
simplified_df = df.drop("LSTAT", axis=1)
simplified_df.describe()
```

[13]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	\
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	
mean	3.537289	11.323320	11.140123	0.069170	0.554318	6.286315	
std	8.344714	23.154645	6.857791	0.253994	0.114938	0.702649	
min	0.009060	0.000000	0.460000	0.000000	0.385000	3.561000	
25%	0.082268	0.000000	5.190000	0.000000	0.449000	5.887250	
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.209000	
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	

	AGE	DIS	RAD	TAX	PTRATIO	B	\
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	
mean	68.629605	3.786801	9.549407	408.188933	18.458261	356.751510	
std	28.051530	2.095925	8.707268	168.337003	2.161832	91.307441	
min	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	
25%	45.175000	2.100175	4.000000	279.000000	17.400000	375.607500	
50%	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	
75%	94.075000	5.118000	24.000000	666.000000	20.200000	396.225000	

```
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000
```

```
             PRICE
count  506.000000
mean    22.532806
std      9.197104
min      5.000000
25%     17.025000
50%     21.200000
75%     25.000000
max     50.000000
```

0.6 Data Descritization

[14]:
```
pd.qcut(df['ZN'], q=40, duplicates="drop").value_counts()
```

[14]:
```
(-0.001, 12.5]    382
(55.625, 80.0]     24
(12.5, 20.0]       23
(33.875, 40.0]     14
(20.0, 22.0]       13
(22.0, 28.0]       13
(40.0, 55.625]     12
(82.5, 100.0]      12
(28.0, 33.875]     11
(80.0, 82.5]        2
Name: ZN, dtype: int64
```