3. Data Pre-processing - Handling missing values, outliers, Normalization, Scaling

November 15, 2022

1 Handling Missing Values and Outliers

Data Cleaning is the process of finding and correcting the inaccurate/incorrect data that are present in the dataset. One such process needed is to do something about the values that are missing in the dataset.For filling missing values, there are many methods available. For choosing the best method, you need to understand the type of missing value and its significance, before you start filling/deleting the data.

1.1 Replacing Missing values by Median

[6]:
```
import pandas as pd
import numpy as np
df=pd.read_csv('missing_values.csv')
print("BEFORE DATA IMPUTATION")

df.head(10)
```

```
BEFORE DATA IMPUTATION
```

[6]:

	RollNo	Name	MARKS1	MARKS2	MARKS3	GRADE	Gender
0	100	'Akash'	NaN	80.0	90.0	'Excellent'	'Male'
1	101	'Ajay'	70.0	NaN	70.0	'Good'	'Male'
2	102	'Rakesh'	45.0	50.0	NaN	'Fair'	'Male'
3	103	'Chandra'	85.0	80.0	70.0	'Good'	'Male'
4	104	'Ramu'	NaN	76.0	90.0	'Fair'	'Male'
5	105	'Devi'	90.0	NaN	70.0	'Good'	'Female'
6	106	'Lakshmi'	82.0	87.0	NaN	'Excellent'	'Female'
7	107	'Neha'	90.0	60.0	70.0	'Excellent'	'Female'
8	108	'Jyothi'	NaN	75.0	80.0	'Excellent'	'Female'
9	109	'Anjali'	85.0	NaN	75.0	'Good'	'Female'

[7]:
```
df['MARKS1']=df['MARKS1'].fillna(df['MARKS1'].mean())
df['MARKS2']=df['MARKS2'].fillna(df['MARKS2'].mean())
df['MARKS3']=df['MARKS3'].fillna(df['MARKS3'].mean())

print("AFTER DATA IMPUTATION")
```

```
df.head(10)
```

AFTER DATA IMPUTATION

[7]:

	RollNo	Name	MARKS1	MARKS2	MARKS3	GRADE	Gender
0	100	'Akash'	75.6875	80.000000	90.0000	'Excellent'	'Male'
1	101	'Ajay'	70.0000	67.333333	70.0000	'Good'	'Male'
2	102	'Rakesh'	45.0000	50.000000	68.1875	'Fair'	'Male'
3	103	'Chandra'	85.0000	80.000000	70.0000	'Good'	'Male'
4	104	'Ramu'	75.6875	76.000000	90.0000	'Fair'	'Male'
5	105	'Devi'	90.0000	67.333333	70.0000	'Good'	'Female'
6	106	'Lakshmi'	82.0000	87.000000	68.1875	'Excellent'	'Female'
7	107	'Neha'	90.0000	60.000000	70.0000	'Excellent'	'Female'
8	108	'Jyothi'	75.6875	75.000000	80.0000	'Excellent'	'Female'
9	109	'Anjali'	85.0000	67.333333	75.0000	'Good'	'Female'

1.2 Filling with most Fixed Values

[9]:
```
import pandas as pd
import numpy as np
df=pd.read_csv('missing_values.csv')
print("BEFORE DATA IMPUTATION:")
df.head(10)
```

BEFORE DATA IMPUTATION

[9]:

	RollNo	Name	MARKS1	MARKS2	MARKS3	GRADE	Gender
0	100	'Akash'	NaN	80.0	90.0	'Excellent'	'Male'
1	101	'Ajay'	70.0	NaN	70.0	'Good'	'Male'
2	102	'Rakesh'	45.0	50.0	NaN	'Fair'	'Male'
3	103	'Chandra'	85.0	80.0	70.0	'Good'	'Male'
4	104	'Ramu'	NaN	76.0	90.0	'Fair'	'Male'
5	105	'Devi'	90.0	NaN	70.0	'Good'	'Female'
6	106	'Lakshmi'	82.0	87.0	NaN	'Excellent'	'Female'
7	107	'Neha'	90.0	60.0	70.0	'Excellent'	'Female'
8	108	'Jyothi'	NaN	75.0	80.0	'Excellent'	'Female'
9	109	'Anjali'	85.0	NaN	75.0	'Good'	'Female'

[19]:
```
df['MARKS1']=df['MARKS1'].fillna(df['MARKS1'].value_counts().index[0])
df['MARKS2']=df['MARKS2'].fillna(df['MARKS2'].value_counts().index[0])
df['MARKS3']=df['MARKS3'].fillna(df['MARKS3'].value_counts().index[0])

print("AFTER DATA IMPUTATION:")
df.head()
```

AFTER DATA IMPUTATION:

[19]:

	RollNo	Name	MARKS1	MARKS2	MARKS3	GRADE	Gender
0	100	'Akash'	75.6875	80.000000	90.0000	'Excellent'	'Male'
1	101	'Ajay'	70.0000	67.333333	70.0000	'Good'	'Male'
2	102	'Rakesh'	45.0000	50.000000	68.1875	'Fair'	'Male'
3	103	'Chandra'	85.0000	80.000000	70.0000	'Good'	'Male'
4	104	'Ramu'	75.6875	76.000000	90.0000	'Fair'	'Male'

1.3 Using Sklearn Simple Imputer

[17]:
```
from sklearn.impute import SimpleImputer
df=pd.read_csv('missing_values.csv')
print("BEFORE DATA IMPUTATION:")
df.head(10)
```

BEFORE DATA IMPUTATION

[17]:

	RollNo	Name	MARKS1	MARKS2	MARKS3	GRADE	Gender
0	100	'Akash'	NaN	80.0	90.0	'Excellent'	'Male'
1	101	'Ajay'	70.0	NaN	70.0	'Good'	'Male'
2	102	'Rakesh'	45.0	50.0	NaN	'Fair'	'Male'
3	103	'Chandra'	85.0	80.0	70.0	'Good'	'Male'
4	104	'Ramu'	NaN	76.0	90.0	'Fair'	'Male'
5	105	'Devi'	90.0	NaN	70.0	'Good'	'Female'
6	106	'Lakshmi'	82.0	87.0	NaN	'Excellent'	'Female'
7	107	'Neha'	90.0	60.0	70.0	'Excellent'	'Female'
8	108	'Jyothi'	NaN	75.0	80.0	'Excellent'	'Female'
9	109	'Anjali'	85.0	NaN	75.0	'Good'	'Female'

[20]:
```
#step 1: create an object for imputer imputer class
imputer=SimpleImputer(strategy='mean',missing_values=np.NaN)

#step 2: fit the data imputation strategy for a dataset
imputer=imputer.fit(df[['MARKS1']])
df['MARKS1']=imputer.transform(df[['MARKS1']])

imputer=imputer.fit(df[['MARKS2']])
df['MARKS2']=imputer.fit_transform(df[['MARKS2']])

imputer=imputer.fit(df[['MARKS3']])
df['MARKS3']=imputer.fit_transform(df[['MARKS3']])

print("AFTER DATA IMPUTATION:")
df.head()
```

AFTER DATA IMPUTATION:

[20]:

	RollNo	Name	MARKS1	MARKS2	MARKS3	GRADE	Gender
0	100	'Akash'	75.6875	80.000000	90.0000	'Excellent'	'Male'
1	101	'Ajay'	70.0000	67.333333	70.0000	'Good'	'Male'
2	102	'Rakesh'	45.0000	50.000000	68.1875	'Fair'	'Male'
3	103	'Chandra'	85.0000	80.000000	70.0000	'Good'	'Male'
4	104	'Ramu'	75.6875	76.000000	90.0000	'Fair'	'Male'

2 Normalization

Normalization means adjusting all the values measured in the different scales, in a common scale. In statistics, normalization is the method of rescaling data where we try to fit all the data points between the range of 0 to 1 so that the data points can become closer to each other.

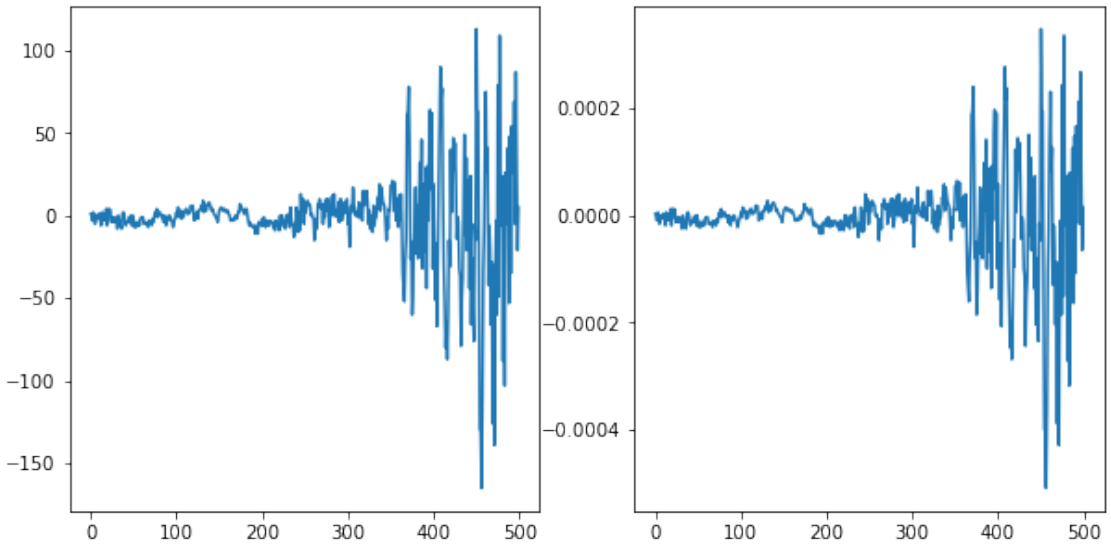
[22]:
```
#read a wav file
import scipy
import scipy.io.wavfile as wav
import matplotlib.pyplot as plt
from sklearn import preprocessing
%matplotlib inline

fs,a=wav.read("normalising_sample_voice_clip.wav")
normalized_a=preprocessing.normalize([a])

print(fs)
print(a)

fig,(ax1,ax2)=plt.subplots(1,2,figsize=(10,5))
ax1.plot(a[500:1000])
ax2.plot(normalized_a[0][500:1000])
plt.show()
```

```
16000
[ 4  4 -1 … -2  2  0]
```



3 Scaling

Scaling of the data comes under the set of steps of data pre-processing when we are performing machine learning algorithms in the data set. scaling of the data makes it easy for a model to learn and understand the problem.

[23]:
```
import numpy as np
from sklearn.preprocessing import MinMaxScaler
```

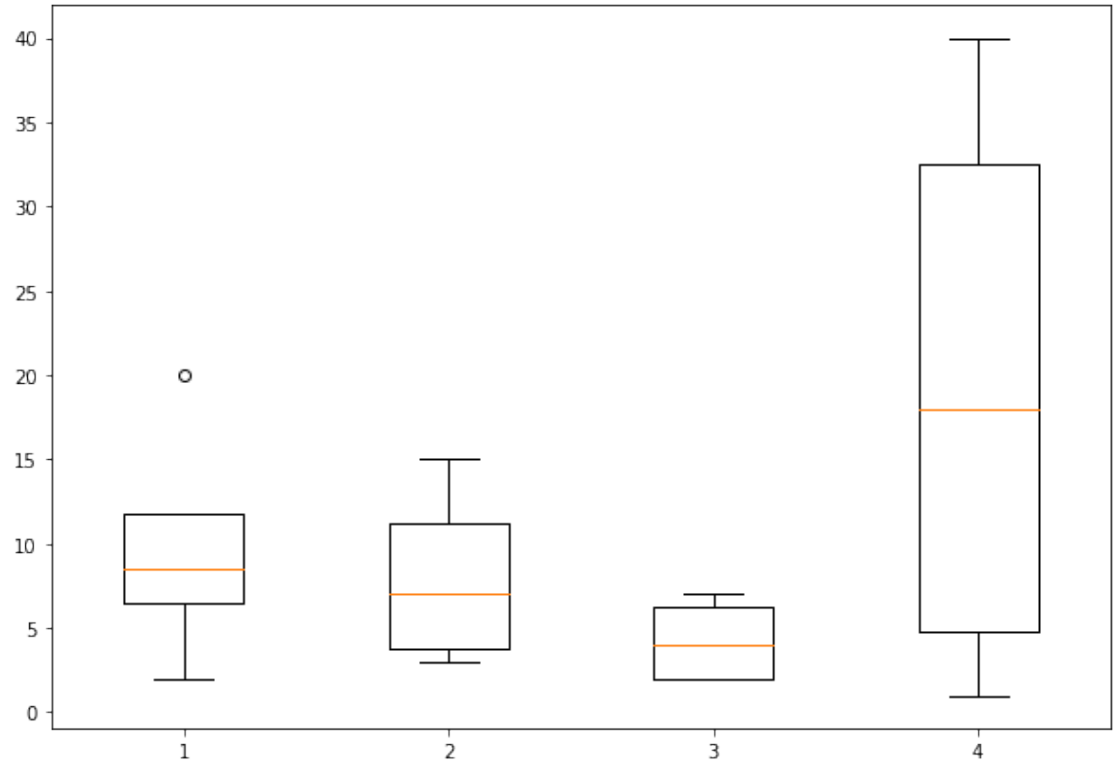
[27]:
```
#Defining an array

df=np.array([[2, 3, 7, 30],
                  [9, 4, 6, 1],
                  [8, 15, 2, 40],
                  [20, 10, 2, 6]])
print(df)
```

```
[[ 2  3  7 30]
 [ 9  4  6  1]
 [ 8 15  2 40]
 [20 10  2  6]]
```

[28]:
```
#Visualizing the array.

import matplotlib.pyplot as plt
fig = plt.figure(figsize =(10, 7))
plt.boxplot(df)
plt.show()
```



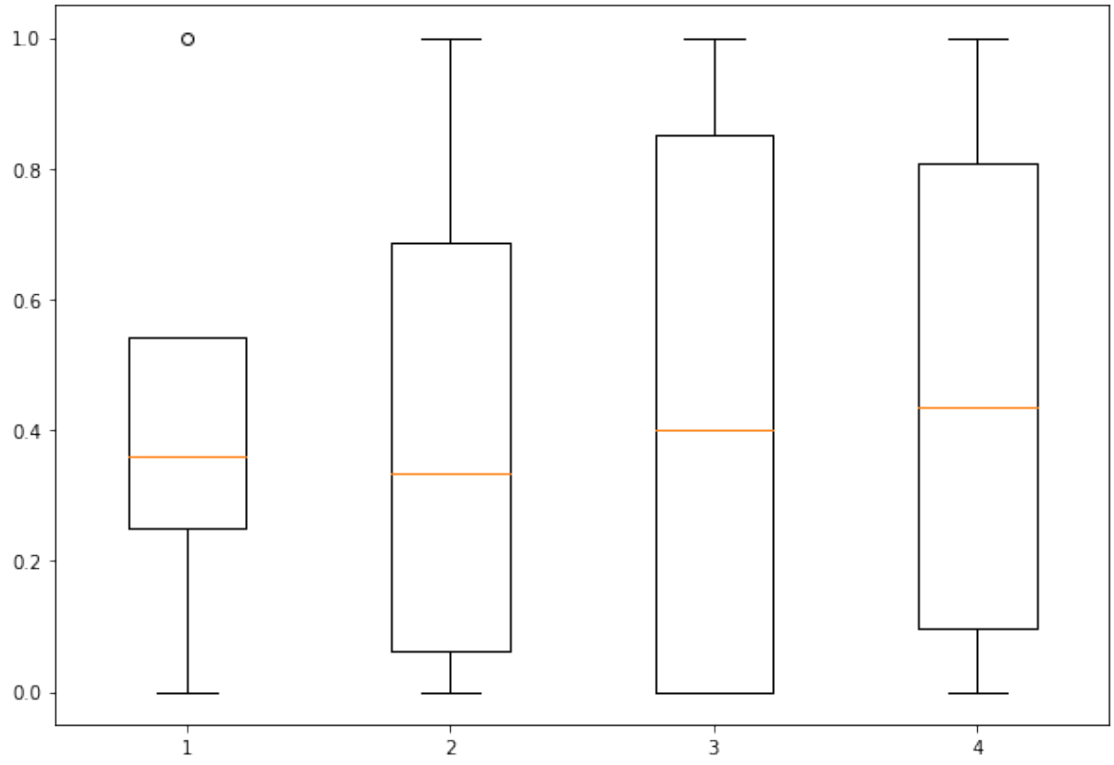
[29]:
```
#Normalizing the array.

scaler = MinMaxScaler()
scaler.fit(df)
scaled_features = scaler.transform(df)
print(scaled_features)
```

```
[[0.         0.         1.         0.74358974]
 [0.38888889 0.08333333 0.8        0.        ]
 [0.33333333 1.         0.         1.        ]
 [1.         0.58333333 0.         0.12820513]]
```

[30]:
```
#Visualizing scaled data:

fig = plt.figure(figsize =(10, 7))
plt.boxplot(scaled_features)
plt.show()
```


1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
4