14. Feature Scaling Technique

Problem:

- Some data are too large in thousands and some data are too less in zeros, so ML algo will dominate the large data and neglect the small data
- To address this problem we introduce Feature scaling technique to bring both the both the datas in the same pitch
- The big data will reduce to bring equal to small data
- You should do feature scaling before training your data

Types of Feature Scaling:

1. Standardization
2. Normalization

Standardization (Z-score normalization)

- It is a very effective technique which re-scales a feature value so that it has distribution with 0 mean value and variance equals to 1

The formula for standardization is:

$$X_{new} = \frac{X_i - \mu}{\sigma}$$

where:

- (X_{new}) is the standardized value,
- (X_i) is the original value,
- (\mu) is the mean of the data,
- (\sigma) is the standard deviation of the data.

- By Scaling - outliers are not removed, though magnitude of outlier will be reduced, but will not affect it significantly

In [2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:
```
dataset = pd.read_csv('loan.csv')
dataset.head(3)
```

Out[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	

Data Cleaning (Identifying and Removing Null Value)

In [4]:
```
dataset.isnull().sum()
```

Out[4]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [5]:
```
dataset['ApplicantIncome'].fillna(dataset['ApplicantIncome'].mode()[0],inplace=True
```

In [6]:
```
dataset.isnull().sum()
```

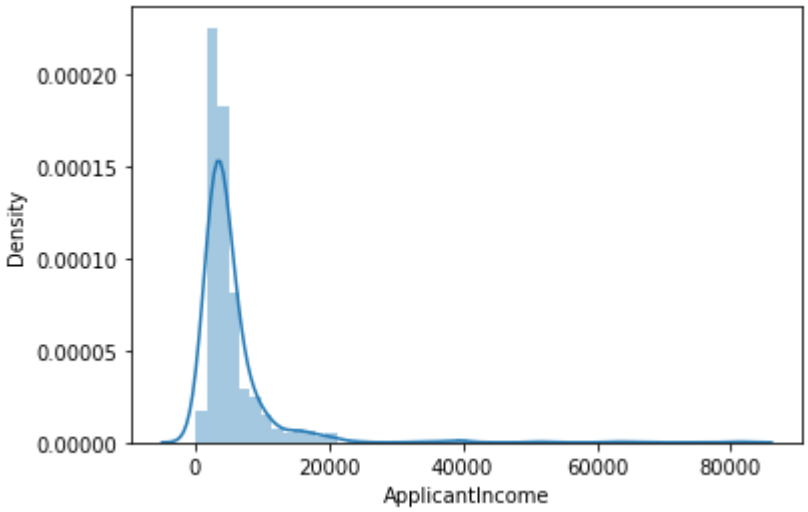
Out[6]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

Check the nature of Data (Outlier Detection)

data: dataset['ApplicantIncome']

In [16]:
```
sns.distplot(dataset['ApplicantIncome'])
plt.show()
```

```
C:\Users\rashi\AppData\Local\Temp/ipykernel_4156/1976060950.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(dataset['ApplicantIncome'])
```



Outlier is present as long tail is evident from the graph showing number of outliers present in the data

In [8]:
```
dataset.describe()
```

Out[8]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Histo
count	614.000000	614.000000	592.000000	600.00000	564.0000
mean	5403.459283	1621.245798	146.412162	342.00000	0.8421
std	6109.041673	2926.248369	85.587325	65.12041	0.3648
min	150.000000	0.000000	9.000000	12.00000	0.0000
25%	2877.500000	0.000000	100.000000	360.00000	1.0000
50%	3812.500000	1188.500000	128.000000	360.00000	1.0000
75%	5795.000000	2297.250000	168.000000	360.00000	1.0000
max	81000.000000	41667.000000	700.000000	480.00000	1.0000

14.1 Feature Scaling of Data by Standardization

In [9]:
```
from sklearn.preprocessing import StandardScaler
```

In [10]:
```
ss = StandardScaler()
ss.fit(dataset[['ApplicantIncome']])
```

Out[10]:
```
StandardScaler()
```

In [11]:
```
# Transform the data and stored in csv file in another column called ApplicantIncom
dataset['ApplicantIncome_ss'] = pd.DataFrame(ss.transform(dataset[['ApplicantIncome
dataset.head(3)
```

Out[11]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	

In [12]:
```
dataset.describe()
```

Out[12]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Histo
count	614.000000	614.000000	592.000000	600.00000	564.0000
mean	5403.459283	1621.245798	146.412162	342.00000	0.8421
std	6109.041673	2926.248369	85.587325	65.12041	0.3648
min	150.000000	0.000000	9.000000	12.00000	0.0000
25%	2877.500000	0.000000	100.000000	360.00000	1.0000
50%	3812.500000	1188.500000	128.000000	360.00000	1.0000
75%	5795.000000	2297.250000	168.000000	360.00000	1.0000
max	81000.000000	41667.000000	700.000000	480.00000	1.0000

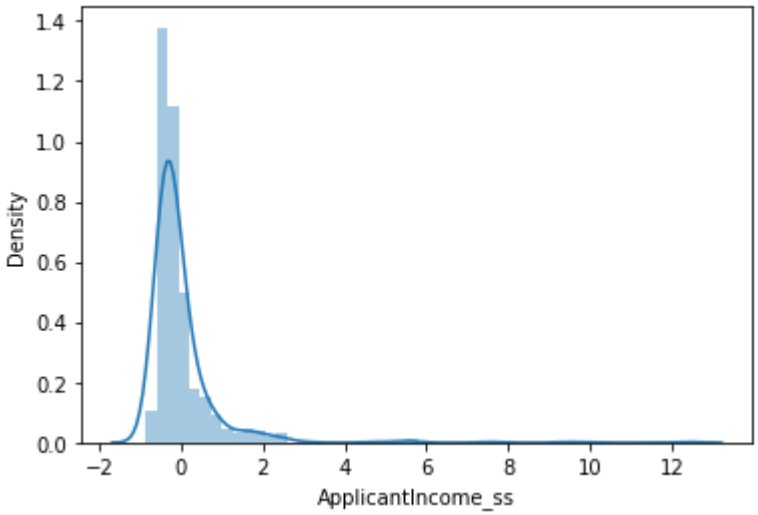
- varaiance is square of standard deviation

Check the nature of transformed Data (Outlier Detection)

data: dataset['ApplicantIncome_ss']

In [13]:
```
sns.distplot(dataset['ApplicantIncome_ss'])
plt.show()
```

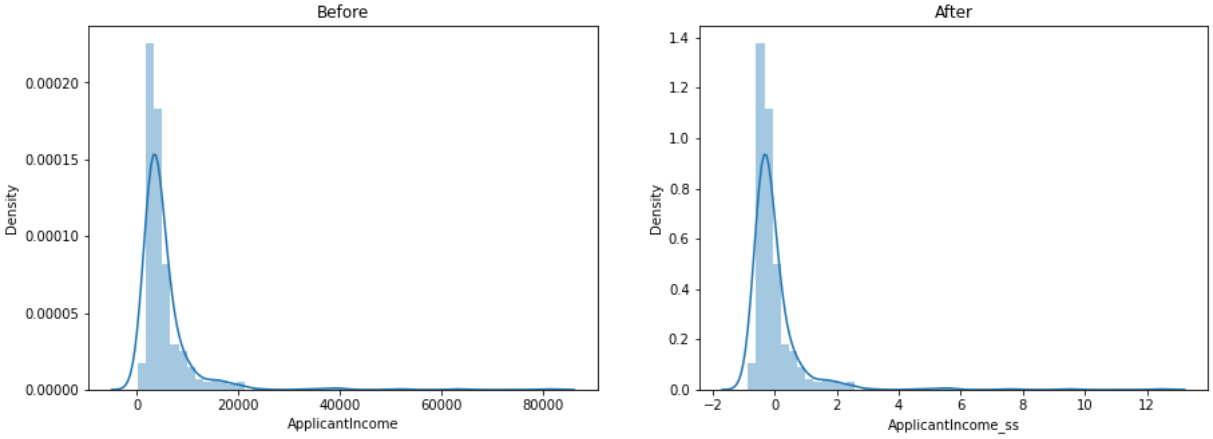
```
C:\Users\rashi\AppData\Local\Temp/ipykernel_4156/3877852283.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(dataset['ApplicantIncome_ss'])
```



- In order to compare both graphs, we will use subplot function to draw both graphs side by side
- subplot(number_of_rows,number_of_col, position)

In [15]:
```
plt.figure(figsize=(15,5))
# plot#1 plt.subplot: row=1, col=2, position=1
plt.subplot(1,2,1)
plt.title('Before')
sns.distplot(dataset['ApplicantIncome'])

# plot#2 plt.subplot: row=1, col=2, position=2
plt.subplot(1,2,2)
plt.title('After')
sns.distplot(dataset['ApplicantIncome_ss'])

plt.show()
```

```
C:\Users\rashi\AppData\Local\Temp/ipykernel_4156/2479568808.py:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(dataset['ApplicantIncome'])
C:\Users\rashi\AppData\Local\Temp/ipykernel_4156/2479568808.py:10: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(dataset['ApplicantIncome_ss'])
```



- So nature of data is not changed after scalling
- Only magnitude of big data is reduced
-

14.2 Feature Scaling of Data by Normalization (Min-Max Scaler)

- Data nature also remain same before and after scalling by Normalization
- Data will be reduced according to min and max values in the data
- Data range after scaling by Normalization is between 0 and 1

Normalization (Min-Max Scaling)

It is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1.

The formula for normalization is:

$$X_{new} = \frac{X_i - X_{min}}{X_{max} - X_{min}}$$

where:

- (X_{new}) is the normalized value,
- (X_i) is the original value,
- (X_{min}) is the minimum value of the feature,
- (X_{max}) is the maximum value of the feature.

In [19]:
```
dataset = pd.read_csv('loan.csv')
dataset.head(3)
```

Out[19]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	

In [21]:
```
dataset.isnull().sum()
```

Out[21]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

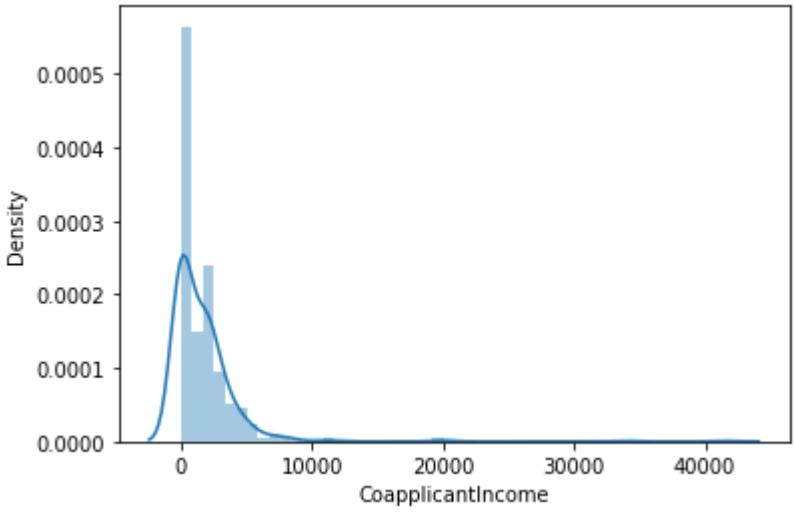
In [22]:
```
dataset.describe()
```

Out[22]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Histo
count	614.000000	614.000000	592.000000	600.00000	564.0000
mean	5403.459283	1621.245798	146.412162	342.00000	0.8421
std	6109.041673	2926.248369	85.587325	65.12041	0.3648
min	150.000000	0.000000	9.000000	12.00000	0.0000
25%	2877.500000	0.000000	100.000000	360.00000	1.0000
50%	3812.500000	1188.500000	128.000000	360.00000	1.0000
75%	5795.000000	2297.250000	168.000000	360.00000	1.0000
max	81000.000000	41667.000000	700.000000	480.00000	1.0000

In [23]:
```
sns.distplot(dataset['CoapplicantIncome'])
plt.show()
```

```
C:\Users\rashi\AppData\Local\Temp/ipykernel_4156/3783729653.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(dataset['CoapplicantIncome'])
```



In [24]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [27]:
```
ms = MinMaxScaler()
ms.fit(dataset[['CoapplicantIncome']])
```

Out[27]:
```
MinMaxScaler()
```

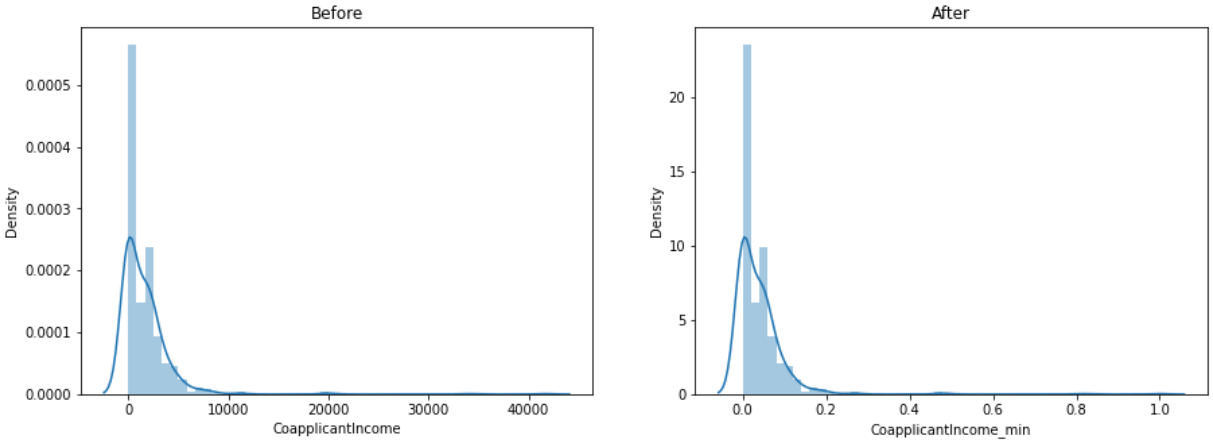
In [30]:
```
dataset['CoapplicantIncome_min'] = pd.DataFrame(ms.transform(dataset[['CoapplicantI
dataset.head(3)
```

Out[30]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	

In [31]:
```
plt.figure(figsize=(15,5))
# plot#1 plt.subplot: row=1, col=2, position=1
plt.subplot(1,2,1)
plt.title('Before')
sns.distplot(dataset['CoapplicantIncome'])

# plot#2 plt.subplot: row=1, col=2, position=2
plt.subplot(1,2,2)
plt.title('After')
sns.distplot(dataset['CoapplicantIncome_min'])

plt.show()
```

```
C:\Users\rashi\AppData\Local\Temp/ipykernel_4156/710565463.py:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(dataset['CoapplicantIncome'])
C:\Users\rashi\AppData\Local\Temp/ipykernel_4156/710565463.py:10: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(dataset['CoapplicantIncome_min'])
```



So you can see the nature of data is not changed through scalling of data by normalization also

In [ ]: