```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
df = pd.DataFrame({
    'Income': [15000, 1800, 120000, 10000],
    'Age': [25, 18, 42, 51],
    'Department': ['HR','Legal','Marketing','Management']
})
df
```

	Income	Age	Department
0	15000	25	HR
1	1800	18	Legal
2	120000	42	Marketing
3	10000	51	Management

```
df_scaled = df.copy()
col_names = ['Income', 'Age']
features = df_scaled[col_names]
features
```

	Income	Age
0	15000	25
1	1800	18
2	120000	42
3	10000	51

MinMax Scalar

MinMax scaler scales all the data between 0 and 1. The formula for calculating the scaled value is-

x_scaled = (x – x_min)/(x_max – x_min)

Though (0, 1) is the default range, we can define our range of max and min values as well.

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

df_scaled[col_names] = scaler.fit_transform(features.values)
df_scaled
```

	Income	Age	Department
0	0.111675	0.212121	HR
1	0.000000	0.000000	Legal
2	1.000000	0.727273	Marketing
3	0.069374	1.000000	Management

```
#suppose we don't want the income or age to have values like 0. Let us take the range to be (5, 10)
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(5, 10))

df_scaled[col_names] = scaler.fit_transform(features.values)
df_scaled
```

	Income	Age	Department
0	5.558376	6.060606	HR
1	5.000000	5.000000	Legal
2	10.000000	8.636364	Marketing
3	5.346870	10.000000	Management

Standard Scaler

For each feature, the Standard Scaler scales the values such that the mean is 0 and the standard deviation is 1(or the variance).

x_scaled = x – mean/std_dev

Standard Scaler assumes that the distribution of the variable is normal. Thus, in case, the variables are not normally distributed, we either choose a different scaler or first, convert the variables to a normal distribution and then apply this scaler

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

df_scaled[col_names] = scaler.fit_transform(features.values)
df_scaled
```

	Income	Age	Department
0	-0.449056	-0.685248	HR
1	-0.722214	-1.218219	Legal
2	1.723796	0.609110	Marketing
3	-0.552525	1.294358	Management

```
df_scaled.describe()
```

	Income	Age
count	4.000000	4.000000e+00
mean	0.000000	-5.551115e-17
std	1.154701	1.154701e+00
min	-0.722214	-1.218219e+00
25%	-0.594947	-8.184910e-01
50%	-0.500791	-3.806935e-02
75%	0.094157	7.804217e-01
max	1.723796	1.294358e+00

Log Transform

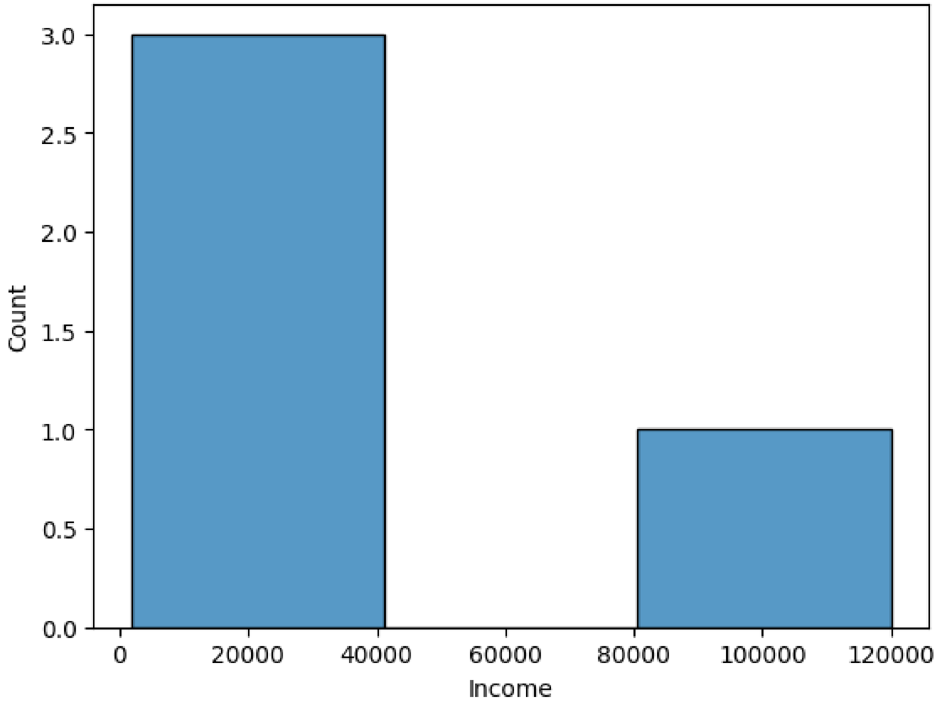
It is primarily used to convert a skewed distribution to a normal distribution/less-skewed distribution. In this transform, we take the log of the values in a column and use these values as the column instead.

the log operation had a dual role:

Reducing the impact of too-low values

Reducing the impact of too-high values.

```
# Histogram
import seaborn as sns
import matplotlib.pyplot as plt
sns.histplot(x=df['Income'], data=df )
plt.show()
```




```
#Apply Log
df['log_income'] = np.log(df['Income'])
df
# We created a new column to store the log values
```

	Income	Age	Department	log_income
0	15000	25	HR	9.615805
1	1800	18	Legal	7.495542
2	120000	42	Marketing	11.695247
3	10000	51	Management	9.210340

```
df['log_income'].plot.hist(bins = 5)
```

```
<Axes: ylabel='Frequency'>
```