Dt:- 3/4/22
11:30pm

$\boxed{\text{"1"}}$ ⇒ **Feature Transformation** → Only For **Continous Data**

Feature processing : Transformation.

↓

columns/variables/Features (same)

Ex:

x
2
4
9
12
25

$x^{0.5}$ $\sqrt{\ }$

$x^{1/2}$

$x^{1/3}$

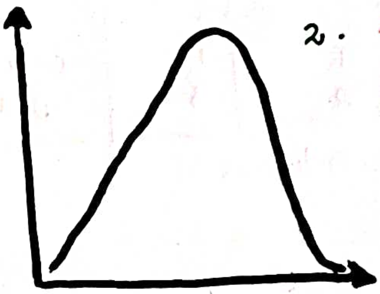
$x^{1/4}$

1. $[x]^{1/2}, [x]^{1/3}, [x]^{1/4}$
2. np.log

Right skewed

+1.5
+2
+6.9 .....

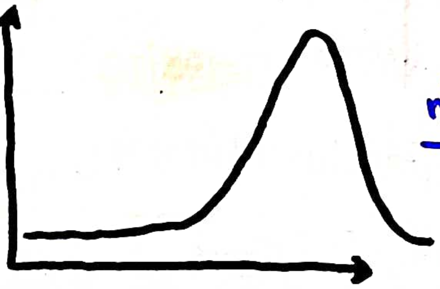
$\xrightarrow[\log(x)]{n^{th} \text{ root (or)}}$



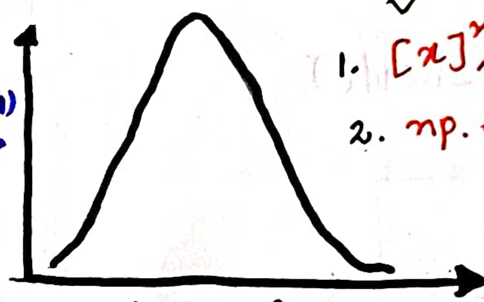
-1 to +1 (normal Distribution)

1. $[x]^2, [x]^3, [x]^4$.
2. np.exp

Left skewed



$\xrightarrow[\exp]{n^{th} \text{ power (or)}}$



-1 to 1

1. Root Transformations
2. log Transformations
3. Reciprocal Transformat
4. Boxcox Transformatio

import Liberaris/packages.

```
# import numpy as np
# Import pandas as pd
# Import matplotlib.pyplot as plt
# % matplotlib inline
# Import Seaborn as Sns
```

titanic = pd. read_csv ("titanic_csv")

Location of csv

titanic

	Passenger Id	Survied	Pclass	Name	Sex	Age	Sibsp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Owen Harris	male	22.0	1	0	A/52771	7.2500	NAN	S
1	2	1	1	cumring	Female	38.0	1	0	.	71.2833	C85	C
2	3	1	3	Laina	Female	26.0	0	0	.	7.9250	NAN	S
3	4	1	1	Futrelle	Female	35.0	1	0	.	53.1000	C123	S
	⋮										⋮	
889	890	1	1	Behr	male	26.0	0	0	.	30.0000	C148	C
890	891	0	3	Dooley	male	32.0	0	0	.	7.7500		

891 rows x 12 columns.

titanic = pd.read_csv("titanic.csv", Usecols= ["Fare"])

titanic_head()

$\boxed{\text{Out}}$

	Fare
0	7.2500
1	71.2833
2	7.9250
3	53.1000
4	8.0500

Particular column

Usecols = ["Fare"]

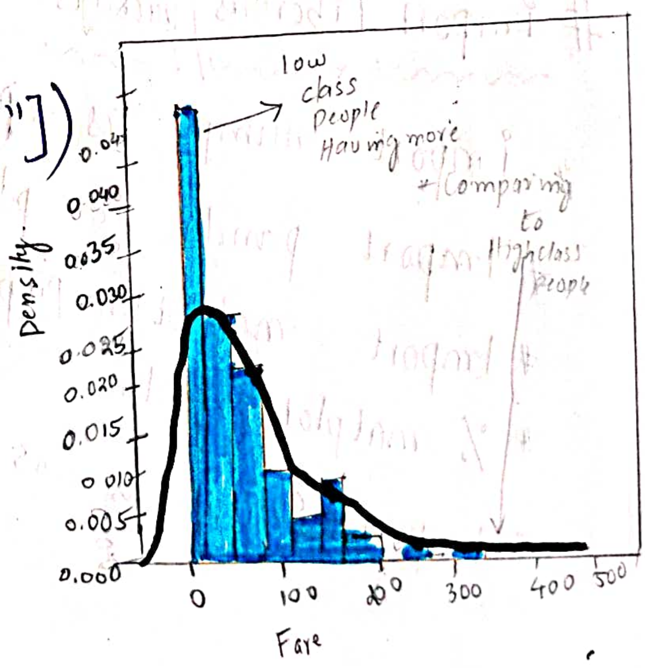
col. Name.

Sns. distplot (titanic ["Fare"])

plt. show()

titanic ["Fare"]. skew()

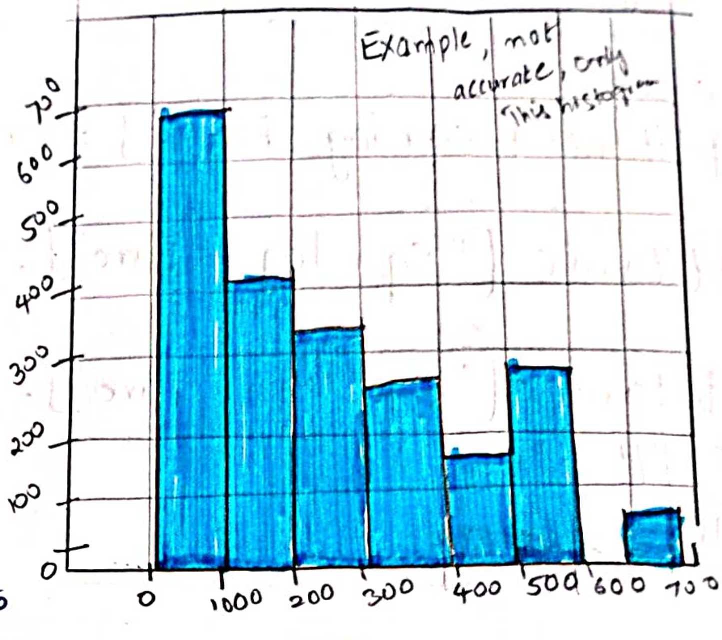
$\boxed{\text{Out :}}$ + 4.7873



another way

```
t - ["Fare"].hist()
plt.show()
titanic["Fare"].skew()
```

Out :- +4.7873



Values

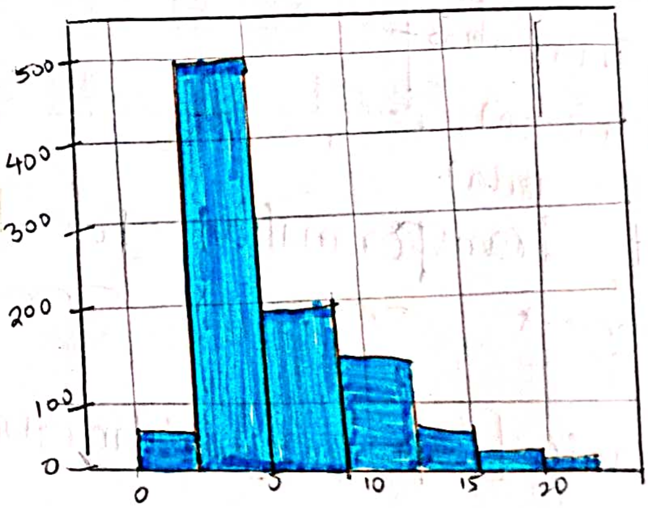
skew = +2.085 [2th root]
skew = +0.5 (4th root)
skew = +0.21 (5th root) — consider, This which is close To "0".
skew = -0.95 (6th root)

* Root Transformation # For Right skewed

```
# titanic["sqr_Fare"] = titanic["Fare"]**(1/2)
# titanic["sqr-Fare"].skew()
# titanic["sqr-Fare"].hist()
```

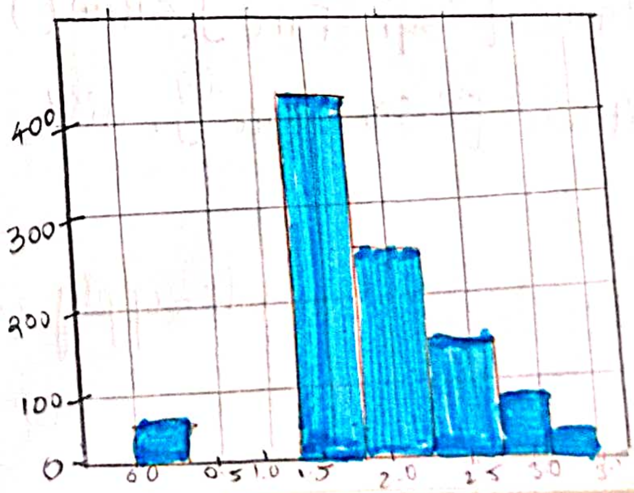
→ For Right Skewed we use n^{th} root $\sqrt{\ }, \sqrt[3]{\ }, \sqrt[4]{\ }, \sqrt[5]{\ }$

Out :- +2.085   More Than +1 or -1



* Try with [1/5],[1/6]...

```
# titanic["sqr_Fare"] = titanic["Fare"]**(1/5)
# titanic["sqr_Fare"].hist()
# titanic["sqr_Fare"].skew()
```



Out :- -0.212676   less Than -1 to +1

Final

→ **log Transformation**

Q. Is, This mandatory, To write Every time?
- No, Only if we "0" in The Data.

```
#titanic ["sqr log_"Fare"] = np.log(titanic["Fare"] +0.01)
#titanic ("sqr log_"Fare"].skew()
#titanic ("sqr log_"Fare"]. hist()
# plt. show()
```

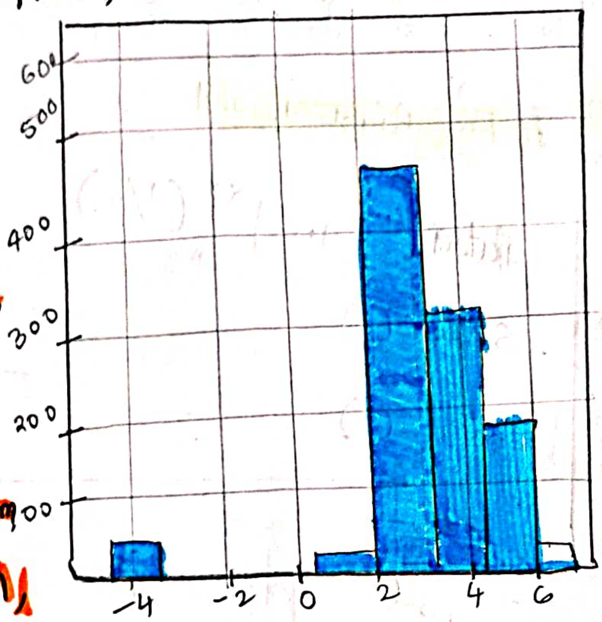
Why?

Because In Data "Fare" column, we have some records "0" values. So, If we calculate with log. it give ∞ (infinite) Value, so, we add +0.01 to "0" column.

Out

-2.41004

Here, The Value, in This Have more, so we don't use log Transform For This col (Data)



Root Transformation For (**Left Skewed**)

→ n^{th} power → $(x)^2, (x)^3, (x)^4$ → left skewed

- n^{th} root $(x)^{1/2}, (x)^{1/3}, (x)^{1/n}$ → Right skewed

we use "n" power

Fare data is not Left skewed, But To understand, Left skewed

```
titanic ["sqr_Fare"] = titanic["Fare"]**(2)
titanic ["sqr- Fare"].skew()
titanic ["sqr- Fare"]. hist()
```

it can be any Value (2) (3) (4) ....

* when The Data is left skewed.

Out :

* We apply $\boxed{n^{th}\text{ power}}$ To The Data (or) column.

* Exponential

```
titanic ["sqr-exp Fare"] = np.exp(titanic ("Fare"))
titanic ["sqr-exp Fare"].skew()
titanic ["sqr-exp Fare"].hist()
plt.show
```

another method For Left Skewed Distribution

* Evaluates $\boxed{e^x}$ For Each Element in The given Input.

* **Reciprocal Transformation**

```
titanic ["Rec-  r"] = 1 / [titanic ["Fare"] + 0.01)
titanic ["Rec-Fare"].skew()
titanic ["Rec-Fare"].hist()
plt.show()
```

"Reciprocal" is that We $\boxed{\text{divide}}$ Every value with (1) / [column] +0.0

of The data

to Elimi "0" value In The Data.

* **Box cox Transformation**

Fare
7.250
71.283
-
-
-
-
-

$7.250^{\lambda}-1$

$71.283^{\lambda}-1$

$$T[x] = \boxed{\frac{x^{\lambda}-1}{\lambda}}$$

∴ λ [lamba] Varies From -5 to +

∴ Where x is The response Variable and

⇒ In This Transformation, all Values o "λ" are Considered and The optimal Value for a given Variables are Selected.

from scipy import stats

If we don't write This Value
Error: Data must be +ve
+0.01 (or) +1 (User defined)

```
# stats.boxcox (titanic["Fare"] +0.01
                                +1 )
```

Out :- (array([2.3844558, 6.43357655, 2.512739737. .
2.607914, 5.7648427, 4.06768781..
. . . .])

Which ever gives best result

From help

we see Two return. values

1. array values (boxcox)
2. optimal Lamda parameter (max log)

Fitting data, Fitting Lambda

```
# titanic ["Fare_boxcox"], param = stats.boxcox (titanic ["Fare"]+1)
```

Multiple Variable assignment

out :- a, b = _ _

Stores

1

2

```
# print ("λ=", param) (or) param
```

Out :- -0.09

```
# titanic ["Fare_boxcox"]. skew ()
```

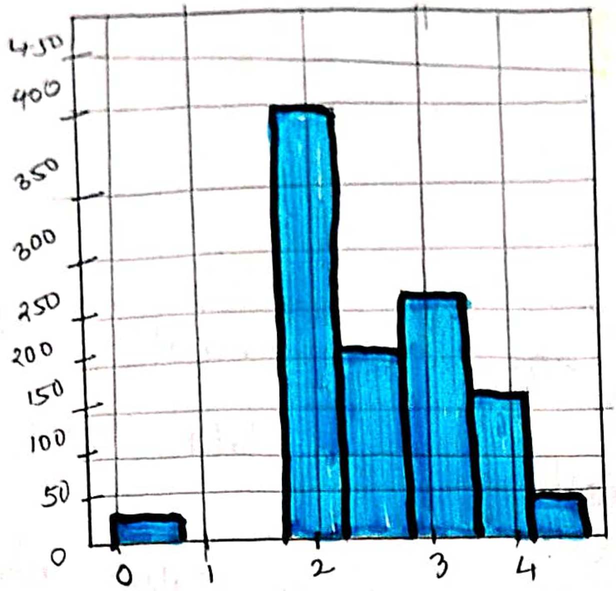
Out :- -0.04

lowest values while compare with other Transformations

Code

```
# titanic ["Fare_boxcox], Param = stats. boxcox (titanic.Fare +1)
# titanic ["Fare_boxcox]. hist ()
# titanic [" Fare_boxcox]. skew()
```

Out :- -0.04



4/4/22

5:30pm.