Plan of Attack

19 March 2024 17:33

feature engg

↓

feature transformations → [maths]

- log
- square
- sqrt
- reciprocal
- Box-cox
- Yeo-Johnson

Dataset → case-study

April

- → xgboost
- → Lightgbm
- → Catboost
- → Adv hyper tun → hyperopt
- → Imbalance data

May - MLOps

Why do we need transformations?

21 March 2024 14:08

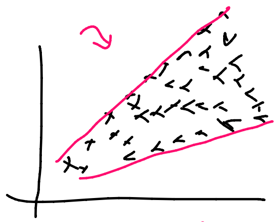
variable trans → target

[Why?]

[To improve the model performance]- as these transformations conform to model assumptions and which in turn amplifies model's predictive power and increases the quality of the model

1. It can even out the variance
2. To make the feature more normal
3. It can reduce the skew
4. It can linearize the relationship between the feature and target
5. Reduce the impact of outliers

liner reg



parametric

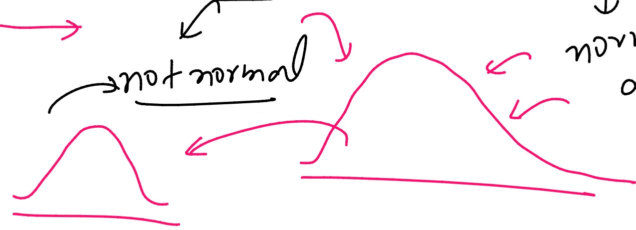
↓

assumption

model assumption

$\boxed{f_1}$ f_2 f_3 → [linear reg] ↓ normally dist

not normal



right skewed

feature

$\boxed{f_1}$ f_2 f_3

math trans

23 → sq. $(23)^2$

24 $(24)^2$

28 $(28)^2$

log trans

log(23)

log 24

log 28

f_1 | y

→ f_1^2 → y

linear

feature trans

linear model

[tree based models]

What are Feature Transformations

21 March 2024 14:24

Feature transformation involves applying mathematical operations to modify the original data features in a way that enhances their representation for machine learning models. These transformations can help in improving model accuracy, meeting algorithm assumptions.

1. Power Transformations
2. Log Transformations
3. Complex Transformations

Problems after transformation

1. Interpretation
2. Finding the best transformation is tricky
3. Additional step in the pipeline

age | salary

x^2

linear

$salary = \beta_0 + \beta_1 age$

Power transform

- x^2
- $x^{1/2}$
- x^{-1}

log trans

- $\log(x)$

complex

- Box cox
- Yeo Johnson

$salary = \beta_0 + \beta_1 (age)^2$

Log Transformation

16 March 2024 10:15

log

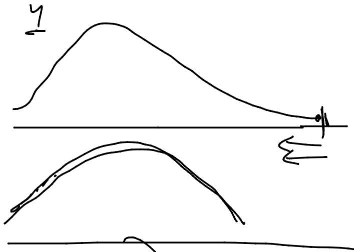
1 − 0 → 1
10 − 1 → 9
linear scale

100 − 2 → 98
1000 − 3 → 997
10000 − 4 → 9996

1 10 100 1000

log scale → large val → more compress

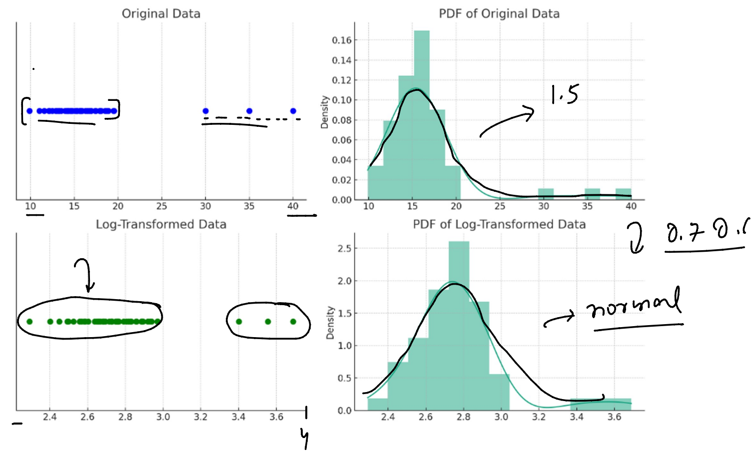
0 1 2 3 4



right skewed moderate

How to check for Normality

1. PDF
2. Skew
3. QQ Plot
4. Statistical test



Algorithms that benefit from Log Transform:

1. Linear Models →
2. ANOVA →
3. Time series analysis → ARIMA
4. K-Means →
5. PCA →
6. Gaussian Naïve Bayes →
7. Training of Neural Networks → fast/stable

(moderate (1−3))

[1 − 3/4] → log transform

[0 − 1] [3/4 →]

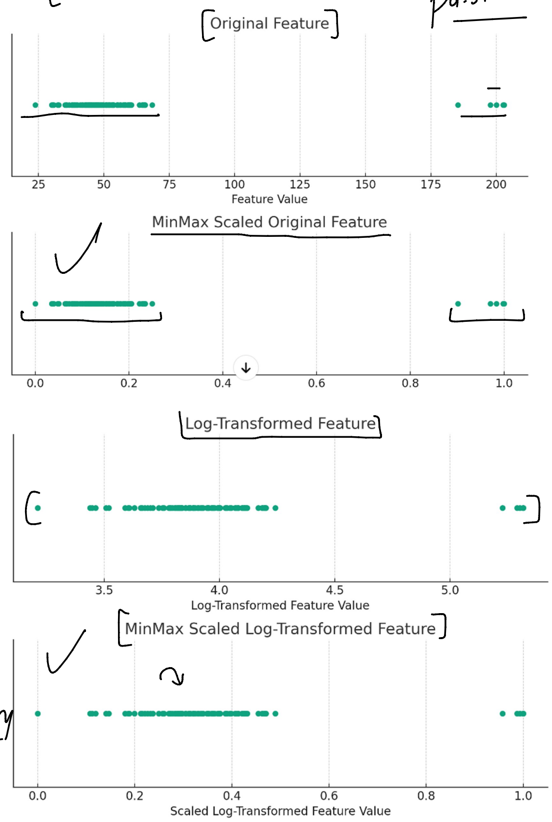
pd.skew() 0 − 1

1 − 3/4
↓
log trans

3/4 >

When to use?

1. When you have right skewed data
2. When your data contains outliers
3. Reduces Heteroskedasticity



When not to use?

1. With negative values/zero values
2. With normal or uniform distribution
3. Interpretation

minmax scaler

robust scaler

log

Inverse Transform

invertible

$\log(x) \rightarrow e^x \rightarrow \boxed{X}$

robust scaler

log transform with minmax scaler

age | salary

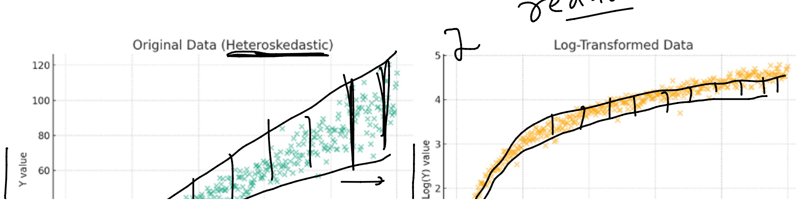
log(age) vs salary

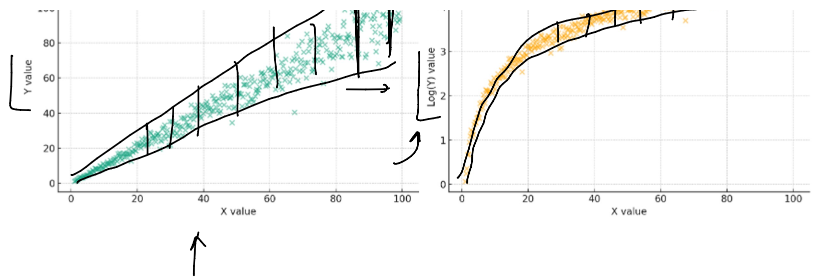
$Y = \beta_0 + \beta_1 \log(x)$ → percent_change →

→ not good

output

reduce





Y value
X value
Log(Y) value
X value

Square Root Transformation

21 March 2024 14:17

graph

sqrt ~ log transforms

1 10 100 1000

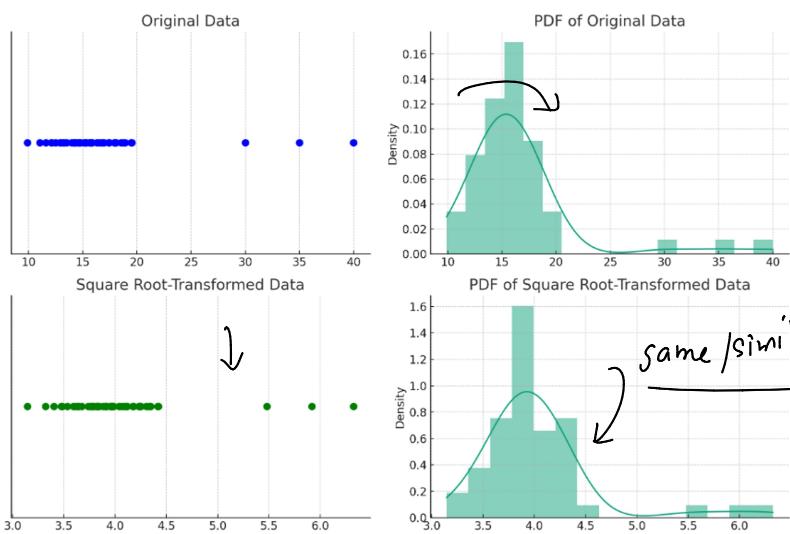
[0 1 2 3]

← moderate (1-3) right skew

1 3. 10 31.6

mild right skewed [0-1] sqrt

← compress



[0-1 → sqrt weak ✓
1-3/4 → log strong
>4/3 → ?]

same/similar [log]

[mild left skewed data]

square trans

mild right skewed dist

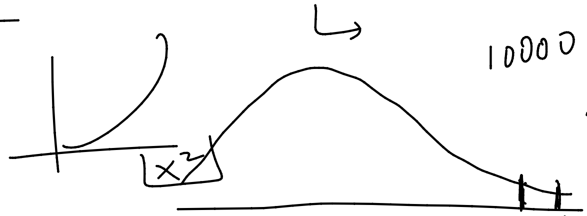
[-1 to 0]

[square trans] → use case

non-linear

↓

square



x^2

10000 12100

→ skew = 0.7

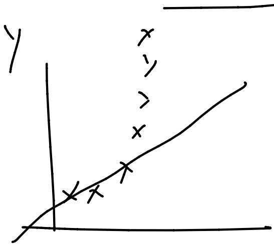
Skew →

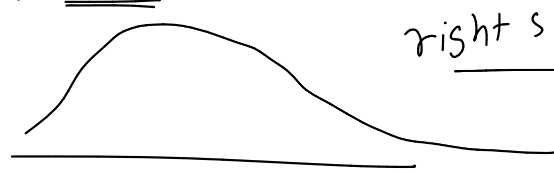
3./4. /5

100 110

↳ square

right skewed





$x \rightarrow y$

$[x, x^2 \rightarrow y]$

square

→ mild left skewed data x^3 / x^4

→ linearize non-linear relationships → x^3 / x^4

Reciprocal Transformation

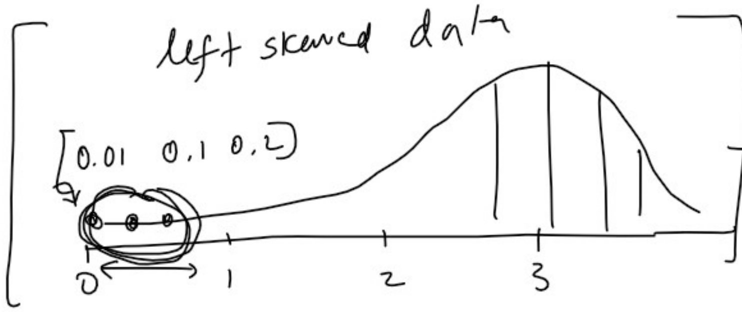
21 March 2024 14:17

$$\left[y = \frac{1}{x} \right]$$

very large

reciprocal

left skewed data

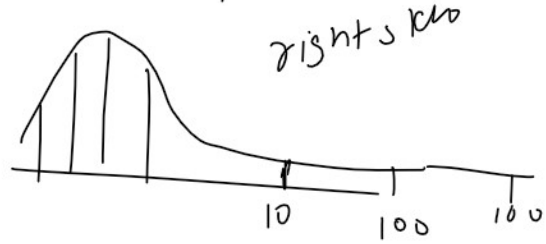


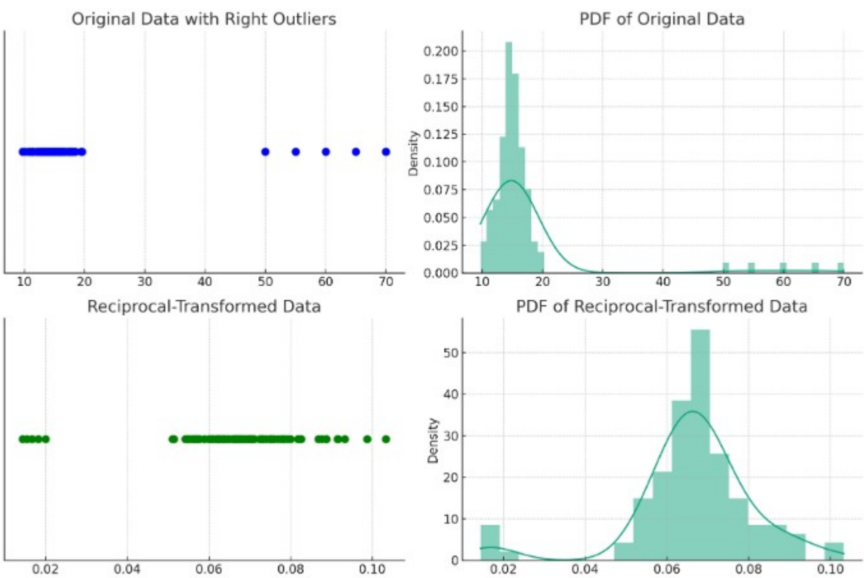
reciprocal

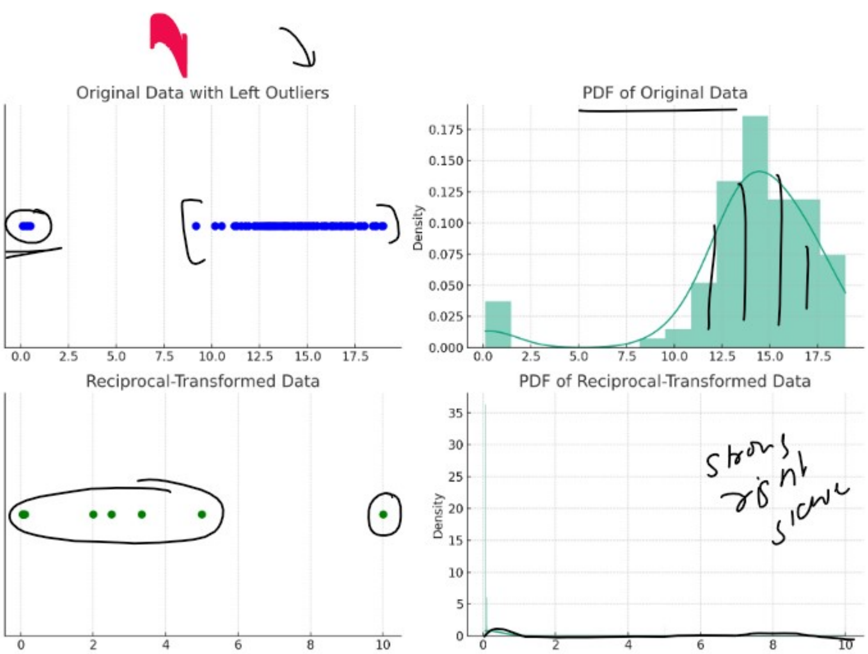
1/

strongly right

right skew







0.01

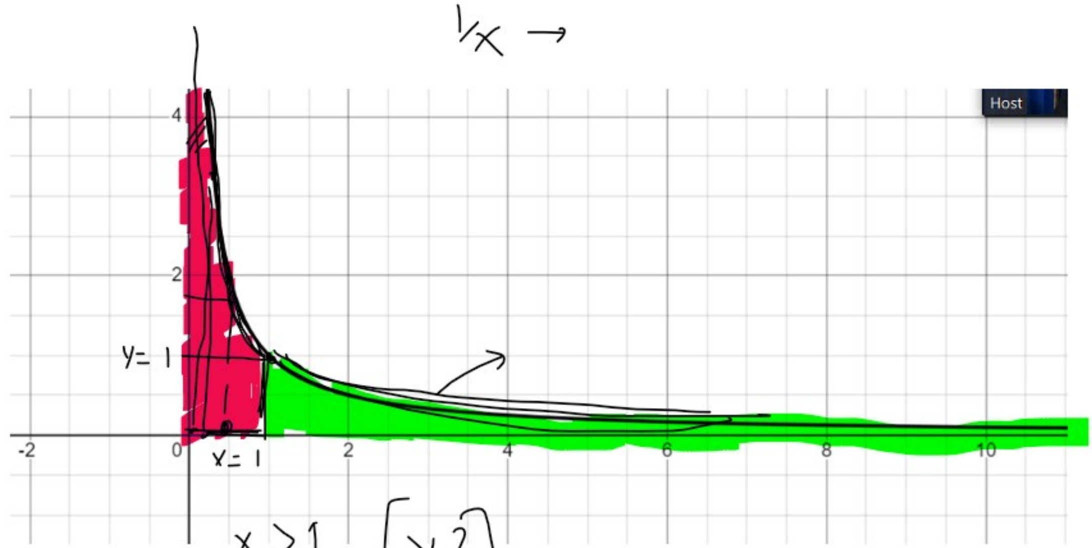
100

0.1

10

$1/x \rightarrow$

$x \simeq 0$



$x > 1$ [y?]

$y \rightarrow [0-1)$

$[x > 1]$

strong right skewed data

larger x → y 0-1

$\boxed{1/x}$

decrea

$\boxed{1/x} \rightarrow$

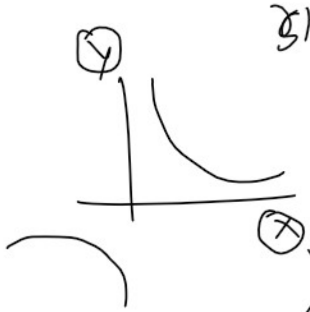
right skewed

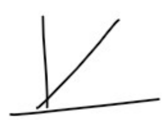
mild → 0-1 → sqrt

modirate → 1-3/4 → log

strong(x>1) → >4 → reciprocal

linear





speed	dist remaining
60	60
70	50
80	40

shifting

−2.89 → [2.89] → log

right

+ve

?

shift

$-2.89 \rightarrow |2.89| \rightarrow 100$

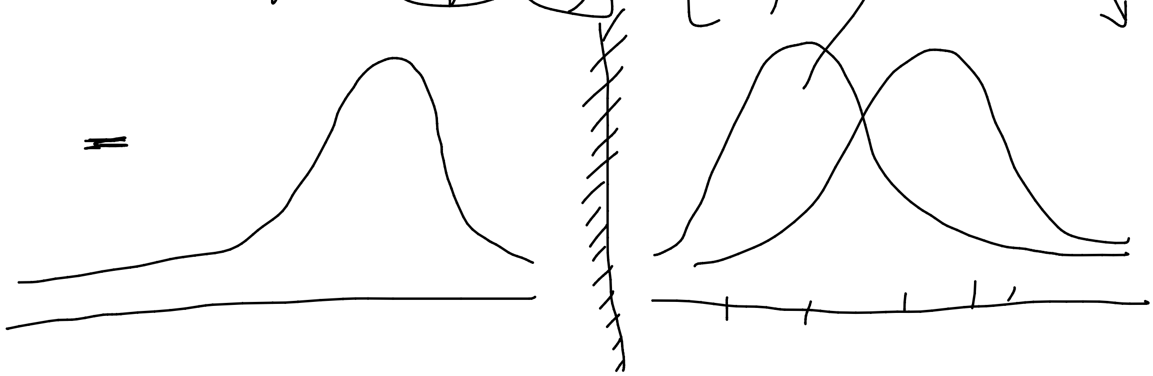
right

+ve

?

$\leftarrow -1 - 0 - 1 - 3 - - - - -$

sq | sqrt | log | reciprocal



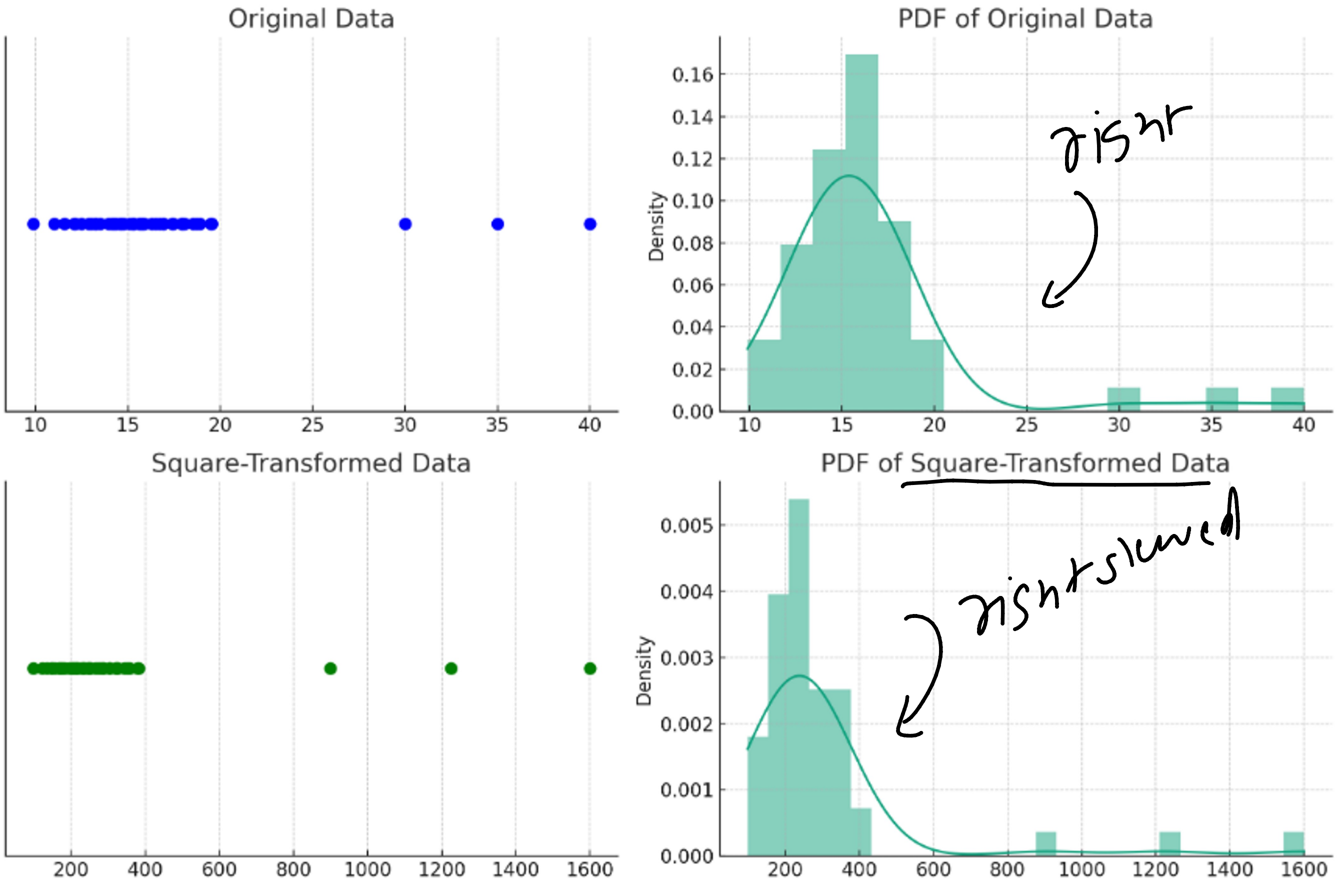
reflect

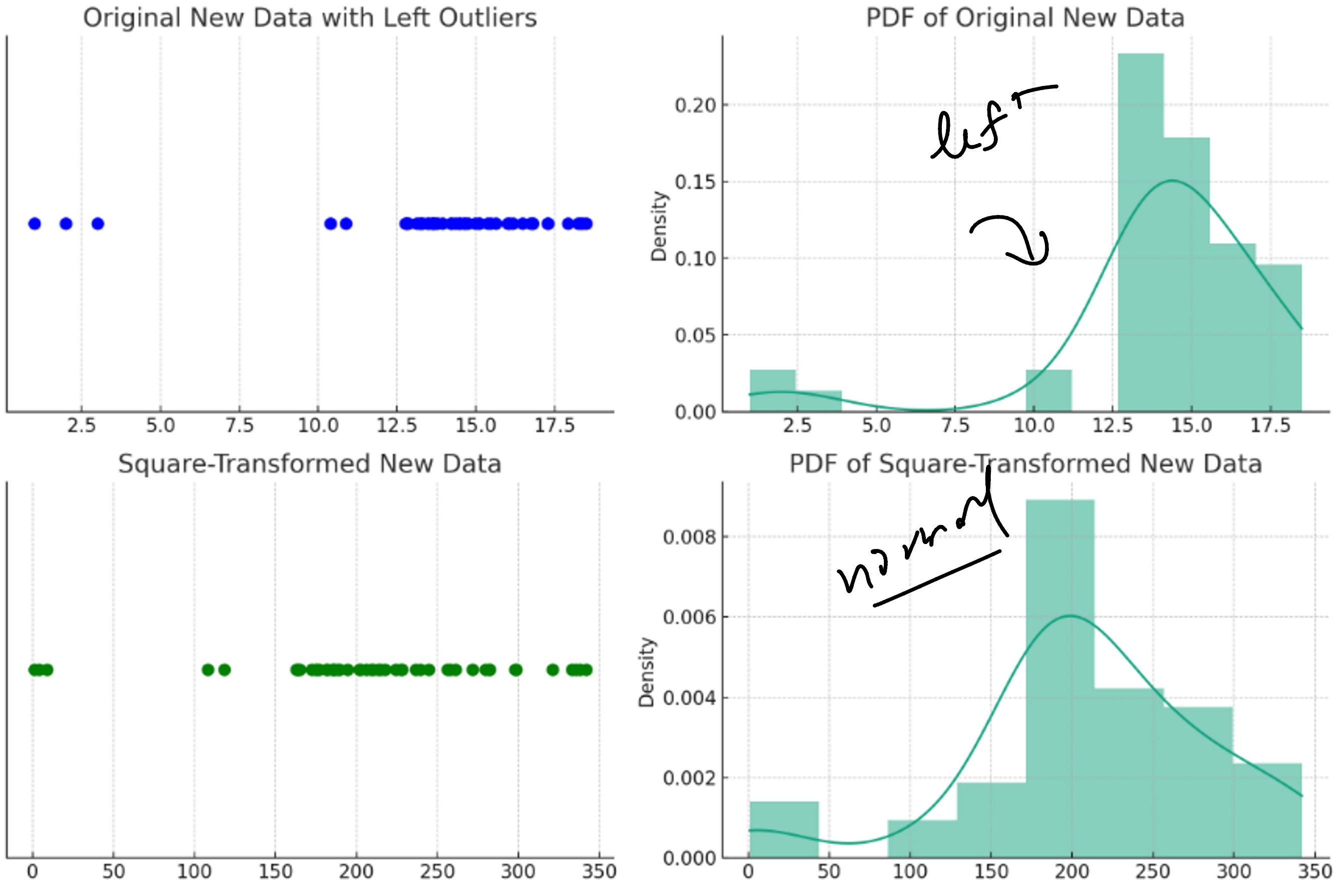
left skewed

↳ reflect → shift → log

Square Transformation

21 March 2024 14:17


Original Data
PDF of Original Data
Density
right
Square-Transformed Data
PDF of Square-Transformed Data
Density
right skewed


Original New Data with Left Outliers
2.5
5.0
7.5
10.0
12.5
15.0
17.5
PDF of Original New Data
Density
0.00
0.05
0.10
0.15
0.20
left
Square-Transformed New Data
0
50
100
150
200
250
300
350
PDF of Square-Transformed New Data
Density
0.000
0.002
0.004
0.006
0.008
normal

Box-Cox Transform

20 March 2024 00:46

$f_1 \rightarrow$ norma

MLE -1 $\frac{1}{2}$

The Box-Cox transformation is a family of power transformations that are applied to data to stabilize variance, make the data more normally distributed, or improve the skewness of both the right and left skewed data. It's particularly useful when dealing with non-normal data that violates assumptions of many statistical tests and models.

λ

$[\lambda = 2]$

$[2.3 \quad . \; -3.2]$

$$Y'(\lambda) = \begin{cases} \frac{Y^{\lambda}-1}{\lambda} & \text{if } \lambda \neq 0, \\ \log(Y) & \text{if } \lambda = 0, \end{cases}$$

$\lambda = -5$ to 5

$f_1 \quad f_2 \quad f_3 \quad y$

2   27 $\lambda = -3$

$\lambda = ?$

$$\boxed{\frac{Y^2 - 1}{2}}$$

MLE

optimization $\lambda = ?$

$$\frac{Y^{-3}-1}{-3}$$

1.4

$$\left[\frac{27^{-3}-1}{-3}\right]$$

$$\frac{Y^{1.4}-1}{1.4}$$

crim$^{-0.11}$

When to use:

1. Handle Skewness
2. Handle non-normal data
3. Heteroscedasticity
4. Can act as general case to many other transformations

When not to use:

1. Negative values
2. Interpretability is a concern
3. Data is already normally distributed
4. Categorical Data

0 / -ve values

Yeo Johnson Transform

21 March 2024 14:17

improvement → Box-Cox → -ve values

The Yeo-Johnson transformation is a modification of the Box-Cox transformation, designed to handle both positive and negative data. While the Box-Cox transformation is only defined for positive data, the Yeo-Johnson transformation extends its applicability to all real numbers, making it more versatile in practical applications.
The Yeo-Johnson transformation is defined as follows:

$f_1 \to \lambda_1$, $f_2 - \lambda_2$

-ve handle

For $y \in \mathbb{R}$ and $\lambda \in \mathbb{R}$, the transformed variable $y'(\lambda)$ is given by:

$$y'(\lambda) = \begin{cases} [(y+1)^{\lambda} - 1]/\lambda & \text{if } \lambda \neq 0, y \geq 0 \\ \log(y+1) & \text{if } \lambda = 0, y \geq 0 \\ -[(-y+1)^{2-\lambda} - 1]/(2-\lambda) & \text{if } \lambda \neq 2, y < 0 \\ -\log(-y+1) & \text{if } \lambda = 2, y < 0 \end{cases}$$

The purpose of the Yeo-Johnson transformation is to stabilize the variance and make the data more closely follow a normal distribution, which is a common assumption in many statistical techniques and models. This transformation is particularly useful when dealing with real-world data that can contain both negative and positive values, ensuring that a wider range of datasets can be normalized for analysis.

Just like the Box-Cox transformation, the parameter is usually determined by maximizing the log-likelihood function, which effectively finds the best that makes the data as normal as possible after the transformation.