Distributions

__/__/____

- A data distribution refers to the way in which data values are spread (or) distribute across a data set.

Key characteristics of a Data Distributions :

1. **Cardinality** (nunique) and **Unique** (For Discrete columns)
2. **Count, Min and Max** (For Numerical columns)
3. **Central Tendency** : This indicates the "center" of the data. common measures of central tendency include the mean (average), median (middle value), and mode (most frequently occuring value).
4. **Dispersion (or) Spread** : This describes how spread out the data points are. Measures of dispersion include Range, Variance, and Stand deviation.
5. **Shape** : This refers to the overall pattern (or) Form of the distribution. Common shapes include normal (bell-shaped), skewed (lopsided), and uniform (Evenly distributed).

Data Distributions

How to Analyse the data distribution ?

- Non-visual → understand the properties of distribution
- visual → Shape, Spread, Outliers.

__/__/____

Agenda

To do probability Distribution

Discreate probability Distribution

Continous probability Distribution

categorical & discreate Numerical

Numerical (OR) Real continous

Bernoulli Distribution

Exactly 2 categories in a column

2 cardinalities. (unique)

Binomial Distribution

More than 2 categories

Advanced dist.

poission Distribution

uniform Distribution

Gaussion Distribution

Normal Distribution.

Exponential Distribution

Advanced dist.

Log - Normal

pareto / power law.

__/__/____

Why Study Data Distributions ?

* Understanding the distribution of data is crucial in a wide range of fields, including Statistics, Data Science, Economics, Finance, Heath care, and more.
* It informs decision-making, helps in selecting appropriate Statistical tests (or) models, and provides insights into the underlying patterns and characteristics of a data set.

* Following are the benifits of studying data distributions :

(1) Outliers and Missing values : Outliers are data points that are significantly different from the rest of the data. knowledge of Data Distribution helps perform a better outlier and missing value treatment.

(2) Advance Data Analysis : Knowledge of data distributions and their PMFs / PDFs can helps us with better insight Generation from the given data and take data driven decisions.

(3) Inferential Statistics : Knowledge of Distribution can help us select the appropriate Statistical Tests.

(4) ML and DL : It also helps us identify appropriate Algorithems for Data Modelling in ML and DL.

__/__/____

Discrete Distributions

① Bernoulli Distribution: (cardinality = 2)

The Bernoulli distribution models a random experiment with two possible outcomes, often labeled as "Success" and "Failure". It is characterized by a single parameter, p, which represents the probability of Success.

For eg:

Loan Approval - Suppose a bank is considering a loan application. The outcome can be either "Approved" (Success) (OR) "Rejected" (failure) based on certain criteria.

Let's say probability of loan approval (p) is 0.8.

- **What is the probability of Rejection?**

  Ans: $1-p = 1-0.8 = 0.2$.

- **Data Follows Bernoulli distribution, we Represent like:**

$$X \sim B(1, p)$$

- **probability Mass Function:**

$$f(x) = \begin{cases} p & \text{if } x=1 \\ 1-p & \text{if } x=0 \end{cases}$$

- **code:**

```
def bernoulli_distribution_generator(p, size):
    # define the no.of trails (n) and probability of success (p)
    return np.random.binomial(n=1, p=p, size=size)

bernoulli_dist = bernoulli_distribution_generator(p=0.30, size=10000)
```

__/__/____

```
from scipy import stats
# using scipy.stats to generate the Bernoulli distribution
prob_success = 0.30
var = stats.bernoulli.rvs(p=prob_success, size=10,000)

# calculating PMF:

print("F(k=0):", stats.bernoulli.pmf(k=0, p=prob_success))
print("F(k=1):", stats.bernoulli.pmf(k=1, p=prob_success))
```

② Binomial Distribution:

* The Binomial distribution models the number of successes (x) in a fixed number of Independent Bernoulli trails (i.e; n), each with the same probability of success (p).

For Eg:

* Number of loan Approvals a day - Assuming the Bank receives 20 loan applications in a day (i.e; n), and each application has an independent probability of approval of $p = 0.8$.

- What is the probability of approving exactly 15 loans?

Ans:

- Data Follows Binomial distribution:

$$X \sim B(n, p)$$

- probability Mass Function:

$$f(x) = \binom{n}{x} (p)^x (1-p)^{n-x}$$

$$\binom{n}{x} = \frac{n!}{x!(n-x)!}$$

__/__/____

code:

```
def binomial_distribution_generator(n, p, size):
    #define the no.of trials (n) & the probability of success (p)
    return np.random.binomial(n=n, p=p, size=size)

binomial_dist = binomial_distribution_generator(n=10, p=0.3, size=10000)

# using Scipy.stats to generate the Binomial distribution
n_trails = 10
prob_Success = 0.30

var = stats.binom.rvs(n=n_trails, P=prob_Success, size=10000)

# computing PMF & CDF:
"probability of 0 Success out of 10 trials:" stats.binom.pmf(k=0,
                                    n=n_trails, p='prob_success')
"probability of 1 Success out of 10 trails:"
 Stats.binom.pmf(k=1, n=n_trails, P=prob_success.)
"probability of 2 Success out of 10 trails:"
 stats.binom.pmf(k=2, n=n_trails, p=prob_Success)
"probability of <=2 Success out of 10 trails:"
 stats.binom.cdf(k=2, n=n_trails, p=prob_Success.)
```

__/__/____

③ Poisson Distribution :

* The poisson distribution models the number of events that occur in a fixed interval of time (or) space, assuming events occur at a constant average rate (λ) and are independent of the time since the last event.

For eg:

- Customers Arrivals at a Bank Branch Every hour - Suppose, on average, a bank branch receives 30 customers per hour. Here $\lambda = 30$ (i.e; Average no. of customers arriving per hour)

- What is the probability of 40 (i.e; x) customers arriving in the Next hour ?

Ans :

- Data follows poisson distribution,

$$X \sim P(\lambda)$$

- probability Mass function :

$$f(x) = e^{-\lambda} \frac{\lambda^x}{x!}$$

- Typically, poisson Distribution has following characteristics :

① It should exhibit a right-skewed, unimodal shape.

② Variance is approximately equal to the mean.

code :

```
def poisson_distribution_generator(lam, size):
    # define expected no. of events occuring in a fixed-time
      interval (lam).
    return np.random.poisson(lam = lam, size = size)
```

__/__/____

poisson_dist = poisson_distribution_generator(lam = 30, size = 10,000)

# using scipy.stats to generate the poission Distribution
lambda_ = 30
var = stats.poisson.rvs(mu = lambda_, size = 10,000)

Computing PMF and CDF
"probability of 20 events happening, given that on an average 30 events happen:"
stats.poisson.pmf(k = 20, mu = lambda_)

"probability of 25 events happening, given that on an average 30 events happen:"
stats.poisson.pmf(k = 25, mu = lambda_)

"probability of 35 events happening, given that on an average 30 events happen:"
stats.poisson.pmf(k = 35, mu = lambda_)

"probability of <= 35 events happening, given that on an average 30 events happen:"
stats.poisson.cdf(k = 35, mu = lambda_)

__/__/____

Multinomial Distribution :

* If you have a discreate column in your data set with more than two distinct values (i.e; a cardinality greaterthan 2), the distribution of this column would be a Specific type of discreate probability distribution known as a Multinomial Distribution.
* In a multinomial distribution, each Observation falls into One of Several categories, with each category having a Specified probability.
* In your case, with three distinct values, you have 3 Categories.

Conclusion :

These examples illustrate how each of these distributions can be applied in different scenarios to model random processess with Specific characteristics.

* The Bernoulli distribution can model binary outcomes like pass/Fail.
* The Bernoulli distribution can model the number of Successess in a fixed number of trails (E.g; Success in a tutoring program)
* The poission distribution can model the Occurances of Events (E.g; Student arrives at a library.) over a Specific time frame.

__/__/____

Continous Distribution

① Uniform Distribution :

* In a uniform distribution, all outcomes are equally likely within a specified range. For, E.g;
  - Rolling a Fair Six-sided Die - When rolling a fair six-sided die, each face has an equal probability of 1/6. This is an example of a discrete uniform distribution.

- Data follows uniform distribution;

$$X \sim U(\alpha, \beta)$$

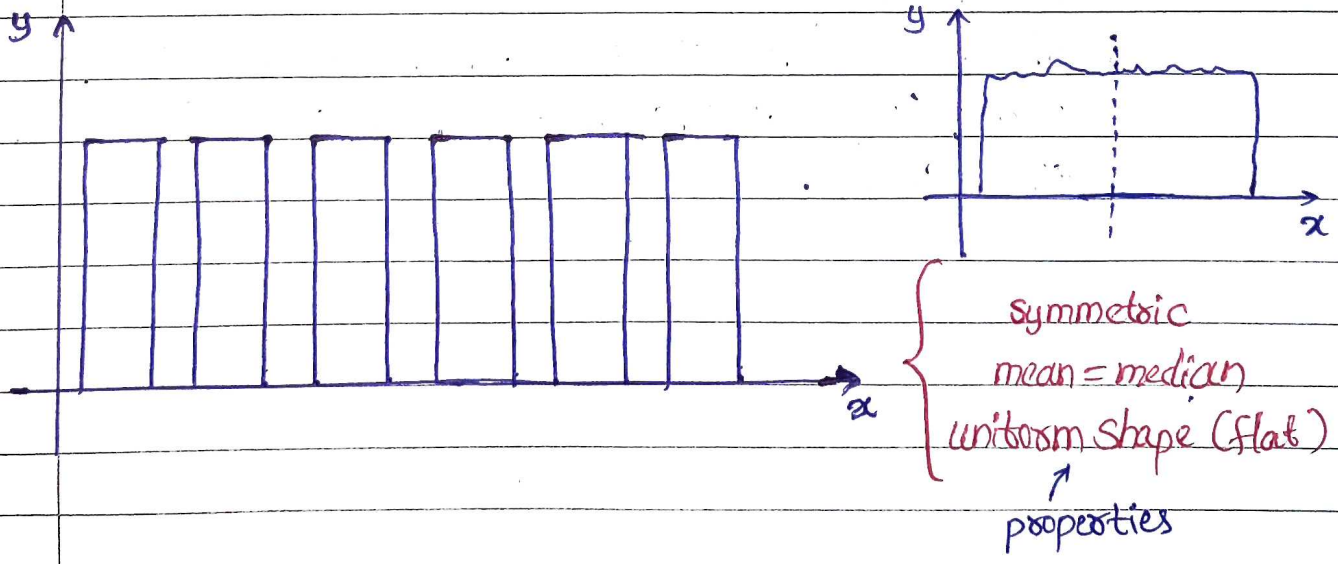
- probability Density Function :

$$f(x) = \begin{cases} \frac{1}{\beta - \alpha} & \text{if } \alpha \le x \le \beta \\ 0 & \text{otherwise} \end{cases}$$

Code:

```
def uniform_distribution_generator(min, max, size):
    return np.random.uniform(low=min, high=max, size=size)

uniform_dist = uniform_distribution_generator(min=10, max=40,
                                              size=10000)
```



__/__/____

Using scipy.stats to generate the uniform distribution :

```
min = 10
max = 40
vas = stats.uniform.rvs(loc = min, scale = max, size = 10,000)
```

computing PDF and CDF :

```
"F(x=0):", stats.uniform.pdf(x = 0, loc = min, scale = max)
"F(x=20):", stats.uniform.pdf(x = 20, loc = min, scale = max)
"F(x=30):", stats.uniform.pdf(x = 30, loc = min, scale = max)
"F(x<=30):", stats.uniform.cdf(x = 30, loc = min, scale = max)
```

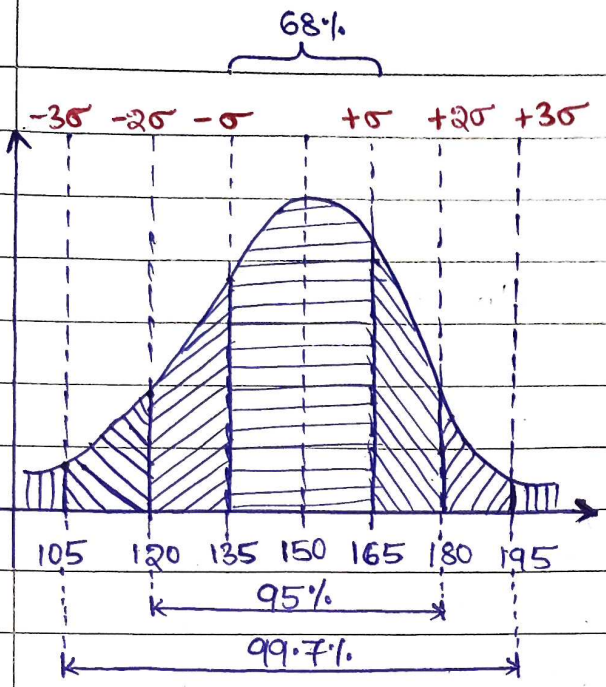
percent point Function (inverse of cdf) :

```
stats.uniform.ppf(q = 0.25, loc = min, scale = max)
```

② Gaussion / Normal Distribution : 68-95-99.7% Rule

* The Normal distribution is characterized by its Bell-shaped curve and is widely used in various fields to model a wide range of natural phenomena.
* This distribution can be described by its Mean and standard deviation, providing insights into the central tendency and spread of the data.
* Important — If a data follow normal distribution, it will follow 68-95-99.7% Rule. i.e; (68%) of total data will lie in (1 std) range, (95%) of the total data will lie in (2 std) range and (99.7%) the total data will lie in (3 std) range.

__/__/____



$\sigma = 15$

properties:

→ Symmetric

→ mean = median

→ Bell shape

Note: How do we identify the type of distribution?

↳ shape of the distribution

- Data follows Normal Distribution:

$X \sim N(\mu, \sigma)$

- probability Density Function:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

Code:

```
def normal_distribution_generator(mean, std, size):
    return np.random.random(loc=mean, scale=std, size=size)

normal_dist = normal_distribution_generator(mean=10., std=40, size=100)
```

__/__/____

using scipy.stats to generate the normal distribution :

```
mean = 10
std  = 40
vars = stats.norm.rvs(loc=mean, scale=std, size=10000)
```

computing PDF and CDF :

```
"F(x=0):", stats.norm.pdf(x=0 , loc=mean, scale=std)
"F(x=20):", stats.norm.pdf(x=20, loc=mean, scale=std)
"F(x=30):", stats.norm.pdf(x=30 , loc=mean, scale=std)
"F(x<=30):", stats.norm.cdf(x=10 , loc=mean, scale=std)
```

percent point function (Inverse of cdf) :

```
stats.norm.ppf(q=0.5, loc=min, scale=max)
```

__/__/____

③ Exponential Distribution :

The exponential distribution models the time until an event occurs in a poisson process, where events occur continously and independently at a constant average rate (λ).

For eg:

* Time between customer Arrivals.
* Modeling waiting times, such as the time between phone calls at a call center.
* Reliability engineering, eg; time until failure of a component.

- Data follows exponential distribution :

$X \sim E(\lambda)$

- probability Density Function :

$$f(x) = \begin{cases} \lambda \exp(-\lambda x) & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

code :

```
def exponential_distribution_generator(lam, size):
    # scale (beta) is inverse of lambda parameter
    return np.random.exponential(scale = 1/lam, size = size)

exponential_dist = exponential_distribution_generator(lam=5,size=1000)

# using scipy.stats to generate the normal distribution :
lambda_ = 5
var = stats.expon.rvs(scale = 1/lambda_, size = 1000)
```

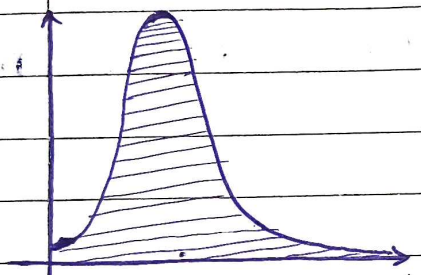
__/__/____

computing PDF and CDF :

```
"F(X=0.5):", stats.expon.pdf(x = 0.5 , scale = 1/lambda_)
"F(X=0.75):", stats.expon.pdf(x = 0.75 , scale = 1/lambda_)
"F(X=0.90):", stats.expon.pdf(x = 0.90 , scale = 1/lambda_)
"F(X<=1):", stats.expon.cdf(x = 1 , scale = 1/lambda_)
```

percent point function (inverse of cdf) :

```
stats.expon.ppf(q = 0.994, scale = 1/lambda_)
```



④ Log-Normal Distribution :

* It is right-skewed and typically used for variables that are positive and have a long right tail.
* The log-normal distribution describes a random variable whose logarithem is normally distributed.

* Applications :

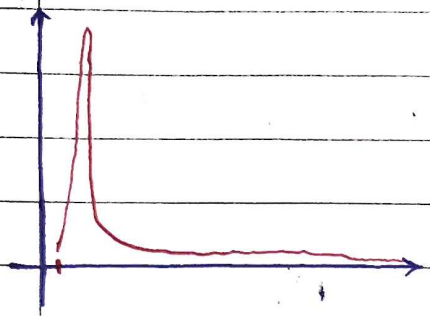
* Modeling variables like income, asset prices, and biological measurements that cannot be negative.
* In Finance, modeling stock prices.

code :

```
def lognormal_distribution_generator(mean, std , size ):
    return np.random.lognormal(mean=mean, sigma=std, size=size)
```

__/__/____

```
lognormal_dist = lognormal_distribution_generator(mean = 0 ,
                                                  std  = 2 ,
                                                  size = 10,000 )
```



* The way location, scale, and shape parameters work in Scipy for the Log-Normal distribution is confusing.

* Suppose a normally distributed random variable X has mean mu and standard deviation sigma.
* Then Y = exp(X) is lognormally distributed with s = sigma and scale = exp(mu).

using scipy.stats to generate the normal distribution :

```
mean = 0
std  = 2
var  = stats.lognorm.rvs(s = std, scale = np.exp(mean), size = 10000)
```

computing PDF & CDF :

```
F(X = 0.5): stats.lognorm.pdf(x = 0.5, s = std , scale = np.exp(mean))
F(X = 0.75): stats.lognorm.pdf(x = 0.75, s = std , scale = np.exp(mean))
F(X = 2): stats.lognorm.pdf(x = 2 , s = std , scale = np.exp(mean))
F(X<=2): stats.lognorm.cdf(x = 2 , s = std , scale = np.exp(mean))
```

```
stats.lognorm.ppf(q = 0.536, s = std , scale = np.exp(mean))
```

__/__/____

⑤ Pareto Distribution : [power-law] 80:20 distribution

* The power-law distribution, also known as a pareto distribution, is characterized by a long tail of rare events.
* It's a heavy-tailed distribution. It describes phenomina where large events are rare but have a substantial impact.
* It is also known as the "80-20 rule".

Applications:

* Modeling city sizes, where a few cities have a very large population while most have relatively few inhabitants.
* In social sciences, representing the distribution of income (or) wealth.

code:

```
def pareto_distribution_generator(alpha, size):
    return np.random.pareto(a=alpha, size=size)

pareto_dist = pareto_distribution_generator(alpha=2, size=10000)

# using scipy.stats to generate the pareto distribution.
alpha = 2
var = stats.pareto.rvs(b=alpha, size=10000)

# computing PDF and CDF :
F(X=0.5): stats.pareto.pdf(x=0.5, b=alpha)
F(X=0.75): stats.pareto.pdf(x=0.75, b=alpha)
F(X=2): stats.pareto.pdf(x=2, b=alpha)
F(X<=2): stats.pareto.cdf(x=2, b=alpha)
```

__/__/____

Verify the Type of Distribution - QQ plot :

Normal Distribution

```
stats.probplot(normal_dist, dist = stats.norm, plot = plt)
```

The above code will help you check observed distribution with Theoritical distribution.

uniform Distribution

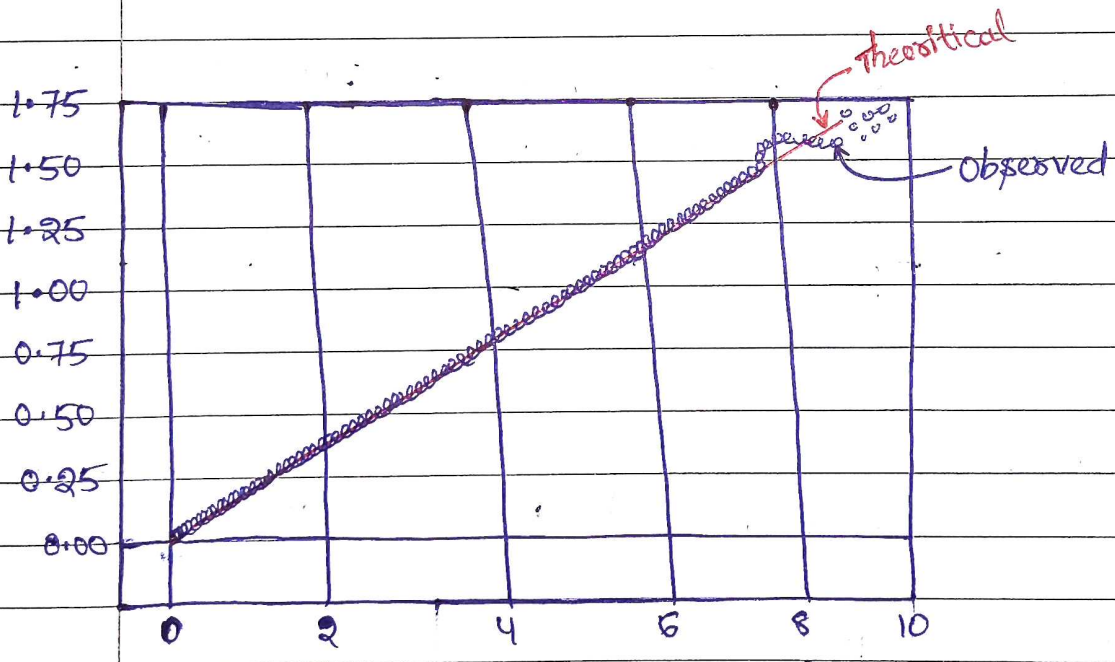
```
stats.probplot(uniform_dist, dist = stats.uniform, plot = plt)
```

uniform_dist ↑ observed distribution

stats.uniform ↑ Theoritical distribution

Exponential Distribution

```
stats.probplot(exponential_dist, dist = stats.expon, plot = plt)
```



__/__/____

Feature Engineering :

(*) Feature Scaling (OR) Rescaling ↓

* Feature Scaling is all about removing the scale difference in Numerical columns.
* Feature Scaling, is a process of transforming data into a common format (OR) Scale.
* This ensures that data from different sources (OR) with different units of measurement can be compared (OR) combined more effectively.

① Min-Max Scaling (Normalization) :

After Rescaling → min = 0
max = 1

$$\Rightarrow \frac{x - min}{max - min}$$

Code :

```
def min_max_scalar(data):
    min_value = np.min(data)
    max_value = np.max(data)
    return (data - min_value)/(max_value - min_value)
```

② Z-transformation (Standardization) :

After Rescaling → $\mu = 0$
$\sigma = 1$

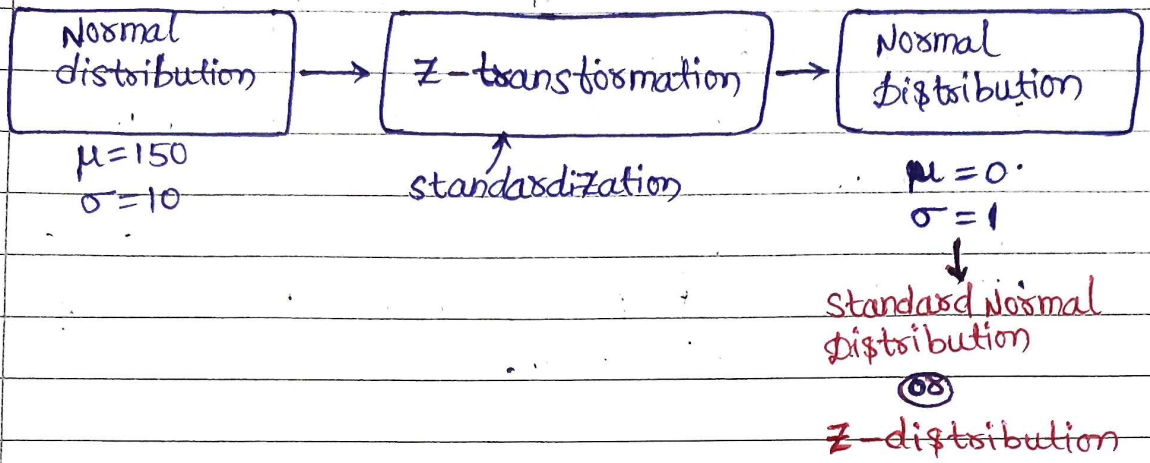
$$\Rightarrow \frac{x - \mu}{\sigma}$$

Code :

```
def Z_transformer(data):
    mean = np.mean(data)
    std  = np.std(data)
    return (data - mean)/(std)
```

__/__/____

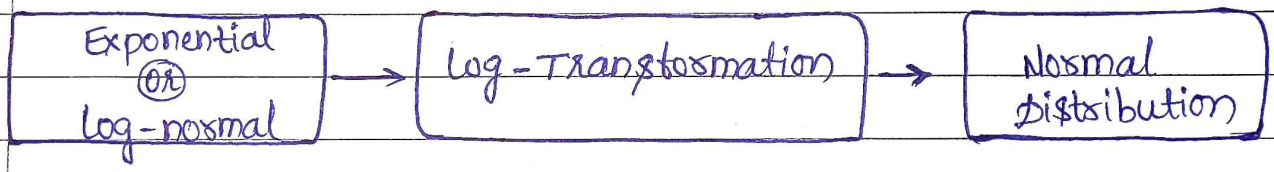
Note:



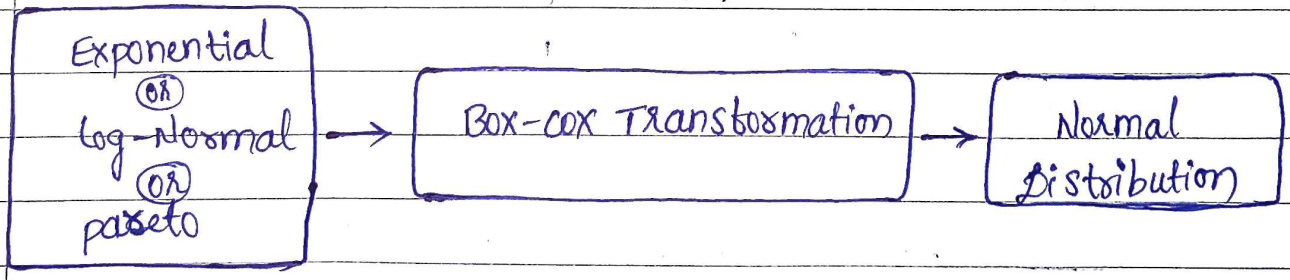
Data Transformation (Reshaping):

- It alters the shape of the distribution.

① Log-Transformation:



② Box-cox Transformation:



__/__/____

Case Study - Budget Allocation :

Scenario : Setting up a Garment Store

your friend is embarking on the journey of setting up a garment store. Launching a new business with limited funds demands meticulous planning, resourcefulness, and a sharp understanding of the market.
For the initial phase, your friend has decided to focus solely on the product - Jeans, and is investing 1,00,000 INR in inventory.

Question: How should the available budget be allocated for inventory?

Answer: Determining the optimal budget allocation can be a bit challenging without a thorough market analysis. A primary suggestion might be to allocate 50,000 INR for both male and Female customers.

However, what if there's a scarcity of male customers? Fortunately, your friend has you, a trusted best friend who also happens to be a successful Data Scientist.
Budget Allocation can relay on many factors like :

① **Gender split :** Since your friend is offering jeans, it's essential to consider the Gender Split in potential customers. This will help in determining how to Allocate the Budget effectively.

② **sizing considerations :** Different body types and sizes should be taken into account. It might be beneficial to allocate a portion of the budget for a diverse range of sizes.

__/__/____

Task - Data driven Budget Allocation :

Let's assume you with the help of an expert got the following information about jeans sizes and Height ranges for both the genders. Consider the following chart for better Budget Allocation.

Group	small	Medium	Large
Female Heights	[120, 136.61]	[136.61 to 153.40]	[154.41 to 170]
Male Heights	[99.87, 128.87]	[128.97 to 160.07]	[160.07 to 190.17]

Given data :

① Import libraries : Numpy, pandas, matplotlib.pyplot, seaborn

② Load data : df = pd.read_csv('data/height-gender-data.csv')

③ Description about data :

The data set containing 2 columns (Height & Gender) and 10,000 rows.

df.columns - Height, Gender

df.shape - (10000, 2).

⇒ Gender Split - Budget Allocation for Male & Female :

Total Budget - ₹1,00,000/-

Non-visual Analysis :

df['gender'].value_counts()

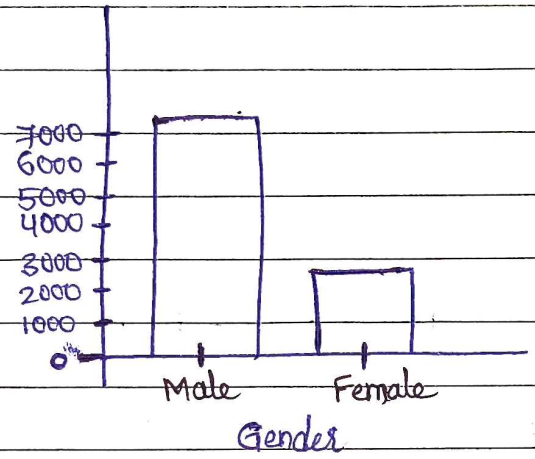
male 7114

Female 2886

__/__/____

Visual Analysis :

```
fig, ax = plt.subplots(figsize=(3,3), facecolor='light yellow', constrained_layout=True)
sns.countplot(data=df, x='gender', ax=ax)
```



- Based on visual & non-visual Analysis Budget Allocation for Male & Female Accordingly.
  - Male - 7,114 out of 10,000 → ₹ 71,140/- ~ 71000/-
  - Female - 2886 out of 10,000 → ₹ 28860/- ~ 29000/-

⇒ **Budget Allocation for Small, Medium & Large Sizes :**

Groupby Gender :

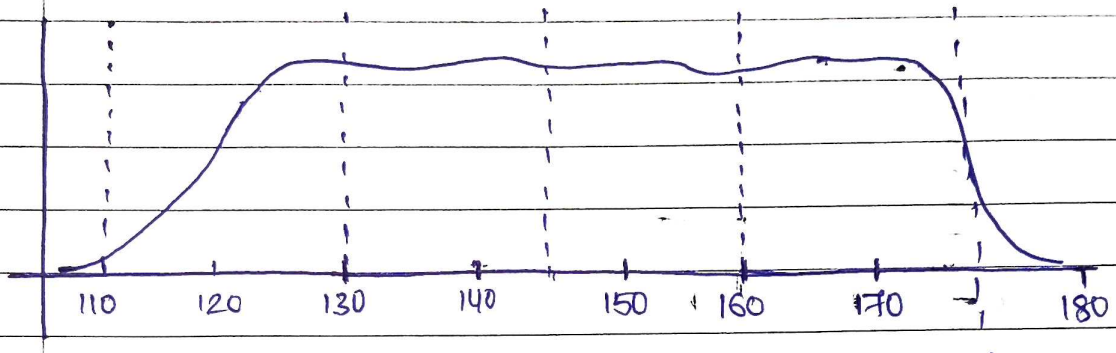
```
grouped_df = df.groupby('gender')

female_height = grouped_df.get_group('Female')['Height']
male_height = grouped_df.get_group('Male')['Height']
```

__/__/____

- Female :

```
fig , axs = plt.subplots (figsize = (8,3), facecolor = 'lightyellow',
                          constrained_layout = True )
axs.set_title ('Histogram plot')
sns.kdeplot (data = female_height, ax=axs, fill = True )
```



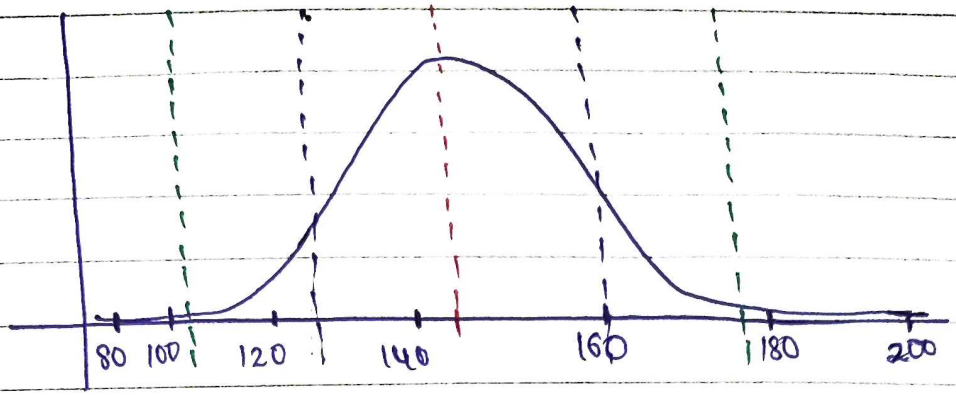
⇒ Female Height follows the uniform distribution -
Budget = ₹ 29,000/-

- Based on this it's better to allocate Equal sum of all the sizes.
  - Small - 29000/3 = 9666/-
  - Medium - 29000/3 = 9666/-
  - Large - 29000/3 = 9666/-

- Male :

```
fig, axs = plt.subplots (figsize = (8,3), facecolor = 'lightyellow',
                         constrained_layout = True )
axs.set_title ('kde plot')
sns.kdeplot (data = male_height, ax = axs, fill = True )
axs.axvline (110, ls = '--', color = 'green')
axs.axvline (130, ls = '--', color = 'orange')
axs.axvline (145, ls = '--', color = 'red' )
axs.axvline (160, ls = '--', color = 'orange')
axs.axvline (175, ls = '--', color = 'green' )
```

__/__/____



⇒ Male Height follows Gaussion distribution - Budget = ₹71,000/-

- Based on this, it can follow 68-95-99.7% Rule
  - Small - 15.85% of 71000 = ₹11,253.5/-
  - Medium - 68% of 71000 = ₹ 48,280/-
  - Large - 15.85% of 71000 = ₹ 11,253.5/-