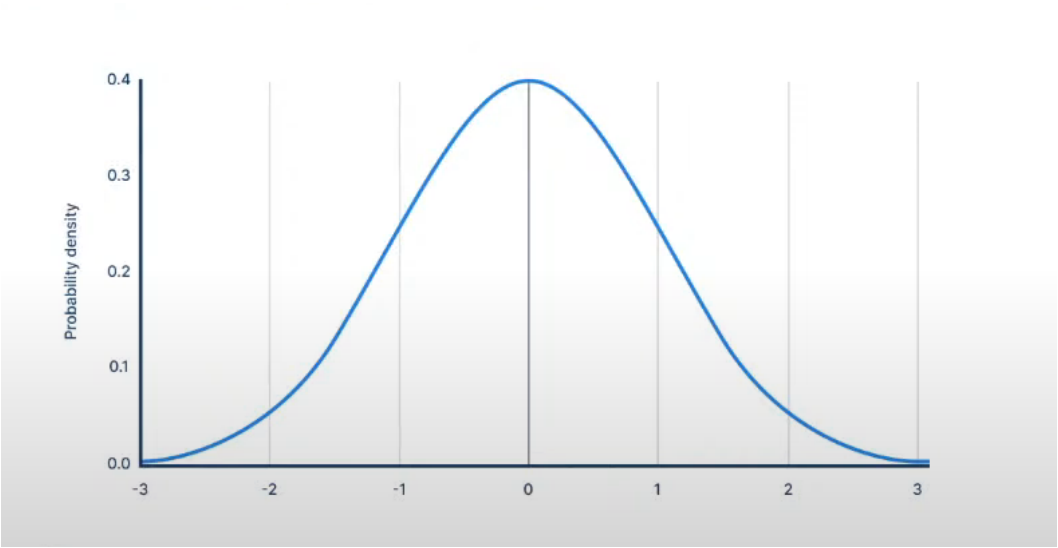
Normal Distribution

- Also known as Gaussian distribution
- continuous probability distribution that is symmetrical around the mean, with a bell-shaped curve.

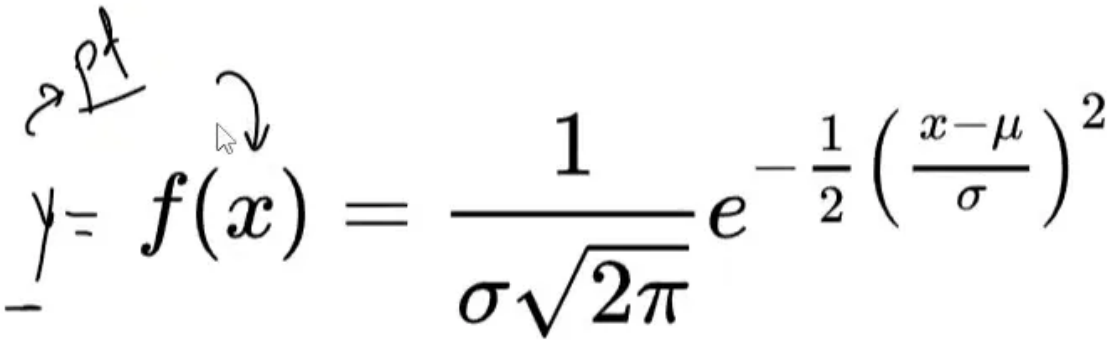


- Lots of points near the mean and very few far away.
- You can create the graph if you hAve mean & SD
- The mean represents the centre of the distribution, while the standard deviation represents the spread of the distribution.

Denoted as → $X \sim N(\mu,\sigma)$

PDF Equation of Normal Distribution

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

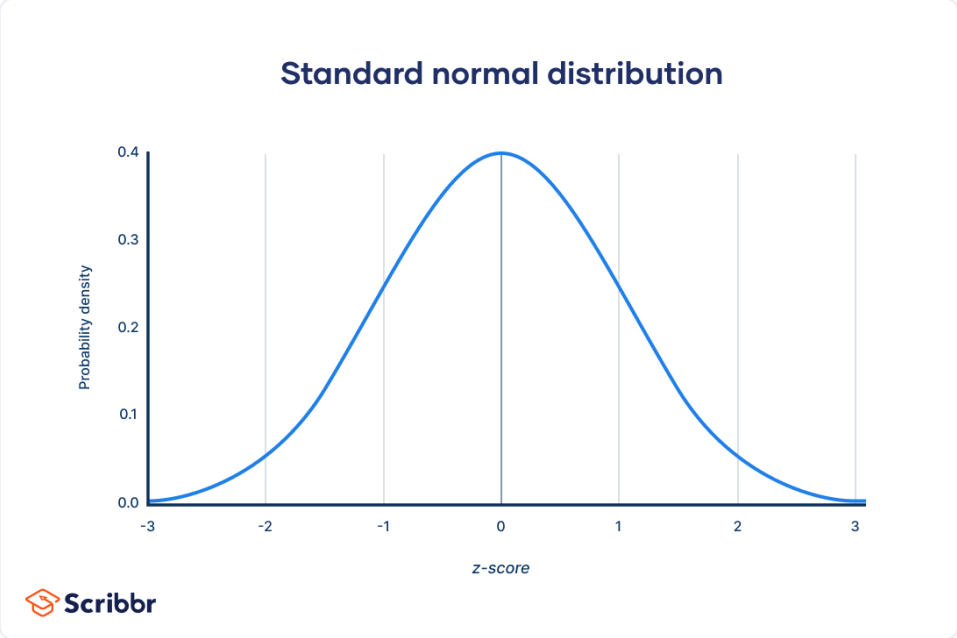


Standard Normal Distribution vs Normal Distribution

- **The standard normal distribution, also called the z-distribution, is a special normal distribution where the mean is 0 and the standard deviation is 1.**
- Any normal distribution can be converted into the standard normal distribution by turning the individual values into **z-scores**.

Standard Normal Variate (Z/Z-score) / Standard Normal Distribution

- It is a standardized form of the normal distribution with mean = 0 and standard deviation = 1.



How to convert normal distr. → Std. Normal Distr.?

$$Z = (X - \mu)/\sigma$$

- **Z** is the Z-score
- **X** is the original value
- **μ** is the population mean
- **σ** is the population standard deviation

💡 **Standard Normal Variate (Z-score) tells you how many standard deviations (SD) a data point is away from the mean.**

Example:

Let's say you have a population with a mean of 70 and a standard deviation of 10. If you have a value of 85, the Z-score would be:

```
Z = (85 - 70) / 10 = 1.5
```

This means that the value of 85 is 1.5 standard deviations above the mean.

Key Points:

- A Z-score of 0 indicates that the data point is equal to the mean.
- A positive Z-score indicates that the data point is above the mean.
- A negative Z-score indicates that the data point is below the mean.

Use Cases

1. **Outlier Detection**:
    - Z-scores outside [−3,3] are often considered outliers.

      [−3,3][-3, 3]
2. **Feature Scaling**:
    - Standardization is used in machine learning models like SVMs and linear regression.
3. **Data Analysis**:
    - SNV simplifies data comparison by removing units and scaling differences.
4. It allows us to compare different distributions with each other, and to calculate probabilities using standardized tables or software.

PDF equation of a standard normal distribution:

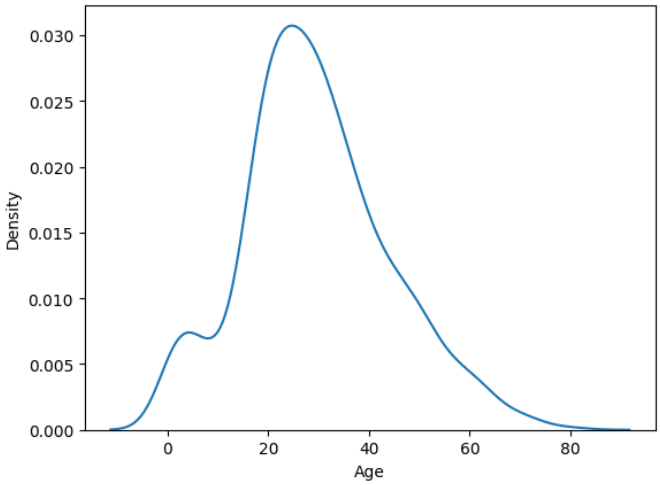
$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

- $f(x)$ is the probability density at point x,
- $\frac{1}{\sqrt{2\pi}}$ is a normalization factor to ensure the total area under the curve equals 1,
- e is Euler's number (approximately 2.71828),
- x is the value at which you're evaluating the PDF (typically a data point),
- $\frac{1}{2}x^2$ is the exponent that controls the bell-shaped curve.

Let’s standardize the age column of titanic dataset:

```
titanic = pd.read_csv('https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv')
```

```
sns.kdeplot(data=titanic,x='Age')
```

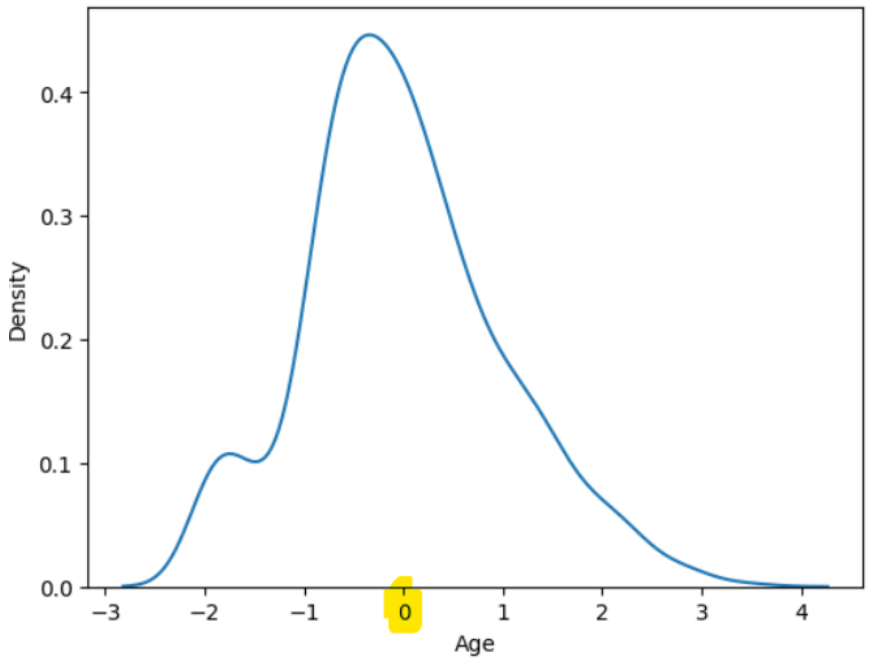


```
titanic['Age'].mean()

Output: 29.69911764705882
```

```
z = (titanic['Age'] - titanic['Age'].mean())/titanic['Age'].std()
```

- Here, we put the values in Z-score formula.

```
sns.kdeplot(z)
```



Question

Q. Suppose the heights of adult males in a certain population follow a normal distribution with a **mean of 68 inches and a standard deviation of 3 inches**. ***What is the probability that a randomly selected adult male from this population is taller than 72 inches***?

```
mean = 68
std_dev = 3
data_point = 72
z_score = (data_point - mean) / std_dev
z_score

Outtput: 1.3333333333333333
```

- We got z score 1.33

- To find out the percentile, we can see the z-table
  - https://math.arizona.edu/~rsims/ma464/standardnormaltable.pdf

STANDARD NORMAL DISTRIBUTION: Table Values Represent AREA to t

Z	.00	.01	.02	.03	.04	.05	.06
0.0	.50000	.50399	.50798	.51197	.51595	.51994	.52392
0.1	.53983	.54380	.54776	.55172	.55567	.55962	.56356
0.2	.57926	.58317	.58706	.59095	.59483	.59871	.60257
0.3	.61791	.62172	.62552	.62930	.63307	.63683	.64058
0.4	.65542	.65910	.66276	.66640	.67003	.67364	.67724
0.5	.69146	.69497	.69847	.70194	.70540	.70884	.71226
0.6	.72575	.72907	.73237	.73565	.73891	.74215	.74537
0.7	.75804	.76115	.76424	.76730	.77035	.77337	.77637
0.8	.78814	.79103	.79389	.79673	.79955	.80234	.80511
0.9	.81594	.81859	.82121	.82381	.82639	.82894	.83147
1.0	.84134	.84375	.84614	.84849	.85083	.85314	.85543
1.1	.86433	.86650	.86864	.87076	.87286	.87493	.87698
1.2	.88493	.88686	.88877	.89065	.89251	.89435	.89617
1.3	.90320	.90490	.90658	.90824	.90988	.91149	.91309
1.4	.91924	.92073	.92220	.92364	.92507	.92647	.92785
1.5	.93319	.93448	.93574	.93699	.93822	.93943	.94062

- We get 0.90824 →**90.824% (Population left to 72)**
- 100- **90.824 = 9.17%**
    - **9.17%** ***probability that a randomly selected adult male from this population is taller than 72 inches***?

By using python:

```
from scipy.stats import norm

# Given Z-score
z_score = 1.33

# Cumulative probability
percent = norm.cdf(z_score) * 100

print(f"Cumulative Percentage: {percent:.2f}%")

Output: Cumulative Percentage: 90.82%
```

- 100- **90.824 = 9.17%**

> 💡 **If you need the reverse (finding the z-score from a given percentile), you can use `norm.ppf()`**

Alt way: Look at negative z value

STANDARD NORMAL DISTRIBUTION: Table Values Represent AREA to the LEFT of the Z score.

Z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
-3.9	.00005	.00005	.00004	.00004	.00004	.00004	.00004	.00004	.00003	.00003
-3.8	.00007	.00007	.00007	.00006	.00006	.00006	.00006	.00005	.00005	.00005
-3.7	.00011	.00010	.00010	.00010	.00009	.00009	.00008	.00008	.00008	.00008
-3.6	.00016	.00015	.00015	.00014	.00014	.00013	.00013	.00012	.00012	.00011
-3.5	.00023	.00022	.00022	.00021	.00020	.00019	.00019	.00018	.00017	.00017
-3.4	.00034	.00032	.00031	.00030	.00029	.00028	.00027	.00026	.00025	.00024
-3.3	.00048	.00047	.00045	.00043	.00042	.00040	.00039	.00038	.00036	.00035
-3.2	.00069	.00066	.00064	.00062	.00060	.00058	.00056	.00054	.00052	.00050
-3.1	.00097	.00094	.00090	.00087	.00084	.00082	.00079	.00076	.00074	.00071
-3.0	.00135	.00131	.00126	.00122	.00118	.00114	.00111	.00107	.00104	.00100
-2.9	.00187	.00181	.00175	.00169	.00164	.00159	.00154	.00149	.00144	.00139
-2.8	.00256	.00248	.00240	.00233	.00226	.00219	.00212	.00205	.00199	.00193
-2.7	.00347	.00336	.00326	.00317	.00307	.00298	.00289	.00280	.00272	.00264
-2.6	.00466	.00453	.00440	.00427	.00415	.00402	.00391	.00379	.00368	.00357
-2.5	.00621	.00604	.00587	.00570	.00554	.00539	.00523	.00508	.00494	.00480
-2.4	.00820	.00798	.00776	.00755	.00734	.00714	.00695	.00676	.00657	.00639
-2.3	.01072	.01044	.01017	.00990	.00964	.00939	.00914	.00889	.00866	.00842
-2.2	.01390	.01355	.01321	.01287	.01255	.01222	.01191	.01160	.01130	.01101
-2.1	.01786	.01743	.01700	.01659	.01618	.01578	.01539	.01500	.01463	.01426
-2.0	.02275	.02222	.02169	.02118	.02068	.02018	.01970	.01923	.01876	.01831
-1.9	.02872	.02807	.02743	.02680	.02619	.02559	.02500	.02442	.02385	.02330
-1.8	.03593	.03515	.03438	.03362	.03288	.03216	.03144	.03074	.03005	.02938
-1.7	.04457	.04363	.04272	.04182	.04093	.04006	.03920	.03836	.03754	.03673
-1.6	.05480	.05370	.05262	.05155	.05050	.04947	.04846	.04746	.04648	.04551
-1.5	.06681	.06552	.06426	.06301	.06178	.06057	.05938	.05821	.05705	.05592
-1.4	.08076	.07927	.07780	.07636	.07493	.07353	.07215	.07078	.06944	.06811
-1.3	.09680	.09510	.09342	.09176	.09012	.08851	.08691	.08534	.08379	.08226
-1.2	.11507	.11314	.11123	.10935	.10749	.10565	.10383	.10204	.10027	.09853

Q. For a Normal Distribution X~(u,std) what percent of population lie between mean and 1
standard deviation, 2 std and 3 std?

```
from scipy.stats import norm

# CDF values at various z-scores
cdf_1 = norm.cdf(1)   # CDF at z = 1 (mean to 1 std deviatio
n)
cdf_2 = norm.cdf(2)   # CDF at z = 2 (mean to 2 std deviation
s)
cdf_3 = norm.cdf(3)   # CDF at z = 3 (mean to 3 std deviation
s)
```

```
# Percentages between the intervals
mean_to_1_std = cdf_1 - norm.cdf(0)   # Between mean (0) and
1 std deviation
one_to_two_std = cdf_2 - cdf_1       # Between 1 std and 2 st
d deviations
two_to_three_std = cdf_3 - cdf_2     # Between 2 std and 3 st
d deviations

# Display results
print(f"Percentage of population between mean and 1 std: {mea
n_to_1_std * 100:.2f}%")
print(f"Percentage of population between 1 std and 2 std: {on
e_to_two_std * 100:.2f}%")
print(f"Percentage of population between 2 std and 3 std: {tw
o_to_three_std * 100:.2f}%")
```

Output:

Percentage of population between mean and 1 std: 34.13%
Percentage of population between 1 std and 2 std: 13.59%
Percentage of population between 2 std and 3 std: 2.14%

- z=1 corresponds to 34.13%.
- z=2 corresponds to 47.72%
- z=3 corresponds to 49.87%.

We can find out the same with

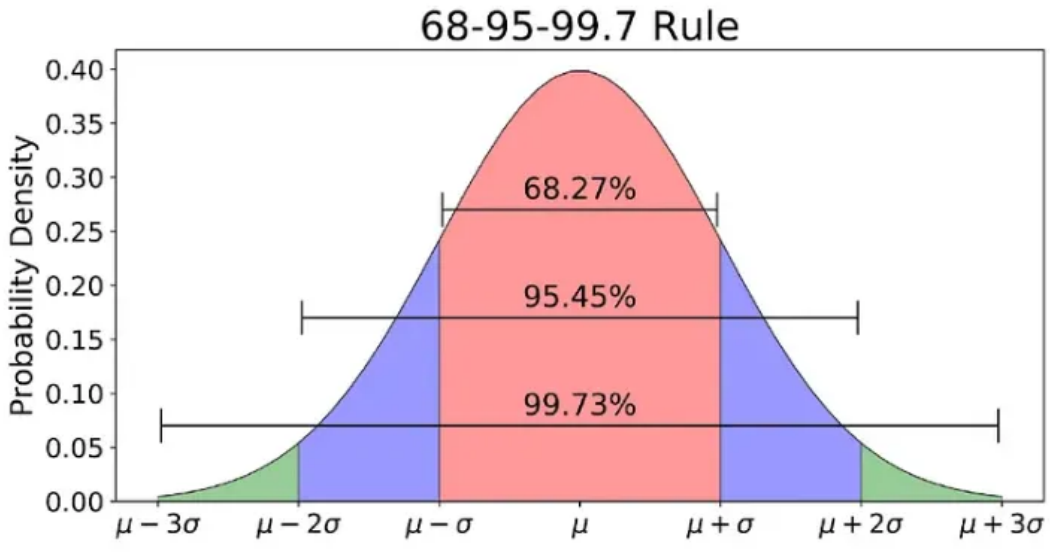
```
from scipy.stats import norm

# Mean and Standard Deviation
mean = 0  # Standard normal distribution
std = 1
```

```
# Z-scores for 1σ, 2σ, and 3σ
z1 = 1
z2 = 2
z3 = 3

# Cumulative probabilities
p1 = norm.cdf(z1) - norm.cdf(0)  # Mean to 1σ
p2 = norm.cdf(z2) - norm.cdf(0)  # Mean to 2σ
p3 = norm.cdf(z3) - norm.cdf(0)  # Mean to 3σ

# Convert to percentages
print(f"Mean to 1σ: {p1 * 100:.2f}%")
print(f"Mean to 2σ: {p2 * 100:.2f}%")
print(f"Mean to 3σ: {p3 * 100:.2f}%")
```

Output:

Mean to 1σ: 34.13%
Mean to 2σ: 47.72%
Mean to 3σ: 49.87%

68-95-99.7 Rule:



The rule states that:

1. **68%** of the population lies within 1 standard deviation of the mean (μ±σ)
2. **95%** of the population lies within 2 standard deviations of the mean (μ±2σ).
3. **99.7%** of the population lies within 3 standard deviations of the mean (μ±3σ).

Properties of Normal Distribution

1. Symmetricity
2. Measures of Central Tendencies are equal - mean, median, mode
3. Empirical Rule - 68-95-99.7 Rule
4. The area under the curve =1

The Area Under the Curve (AUC)

- The **CDF** tells you this area, which is the percentage of the population that lies below the given z-score.
-

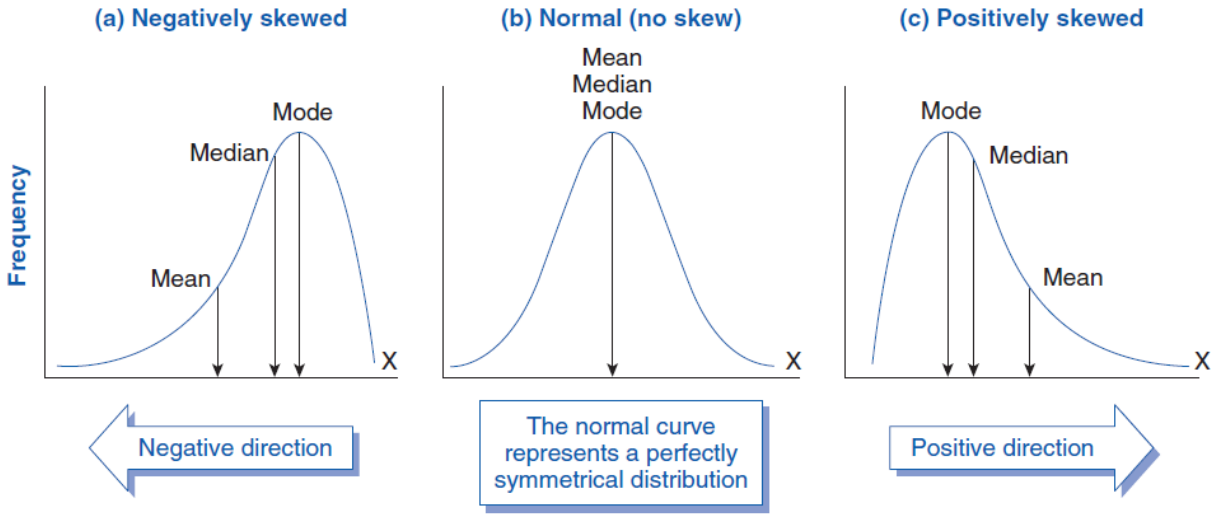
```
from scipy.stats import norm

# Calculate the CDF values
cdf_1_5 = norm.cdf(1.5)
cdf_0 = norm.cdf(0)

# Area under the curve between 0 and +1.5
area_between_0_and_1_5 = cdf_1_5 - cdf_0
print(f"Area under the curve between 0 and +1.5 standard devi
ations: {area_between_0_and_1_5 * 100:.2f}%")
```

Output:

Area under the curve between 0 and +1.5 standard deviations: 43.32%

Skewness

- **Skewness** is a measure of the asymmetry or the departure from symmetry in the distribution of a dataset. It describes the direction and degree of skew (or lopsidedness) in the data distribution.



Positive Skew (Right Skew):

- The right tail (larger values) is longer or fatter than the left tail.
- Most of the data points are concentrated on the left side of the mean, and fewer data points are located on the higher side.
- The mean is greater than the median (**mean > median**).
- Example: **Income distribution** where most people earn lower wages, but a few earn significantly higher wages.

- **Weakly Positive Skew**: Skewness between 0 and 0.5
- **Moderately Positive Skew**: Skewness between 0.5 and 1
- **Strongly Positive Skew**: Skewness greater than 1

Negative Skew (Left Skew):

- The left tail (smaller values) is longer or fatter than the right tail.
- Most of the data points are concentrated on the higher side, and fewer data points are on the lower side.
- The mean is less than the median (mean < median).
- Example: Test scores where most people score higher marks, but a few students score very low.

- **Weakly Negative Skew**: Skewness between 0 and -0.5
- **Moderately Negative Skew**: Skewness between -0.5 and -1
- **Strongly Negative Skew**: Skewness less than -1

Zero Skew (Symmetrical Distribution):

- The distribution is symmetric with respect to the mean.
- In this case, the data is perfectly balanced on either side of the mean.
- Example: A normal distribution (bell curve).
- The skewness value is **0**.

💡 In practice, skewness values generally fall within the range of **-3 to +3**, though values outside this range are possible in some cases.

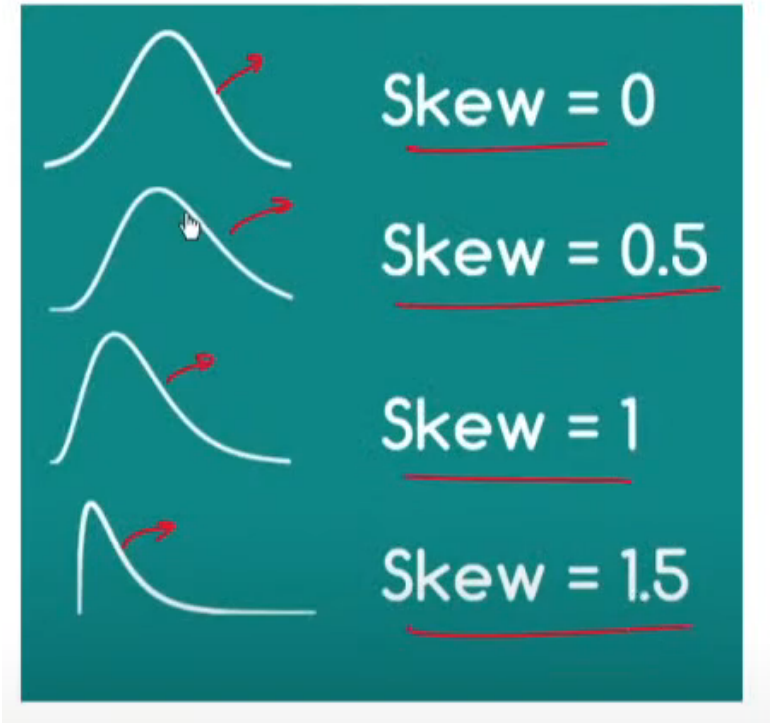
Skewness Formula:

The formula to calculate skewness for a dataset is:

$$\text{Skewness} = \frac{n}{(n-1)(n-2)} \sum_{i=1}^{n} \left(\frac{x_i - \mu}{\sigma} \right)^3$$

Where:

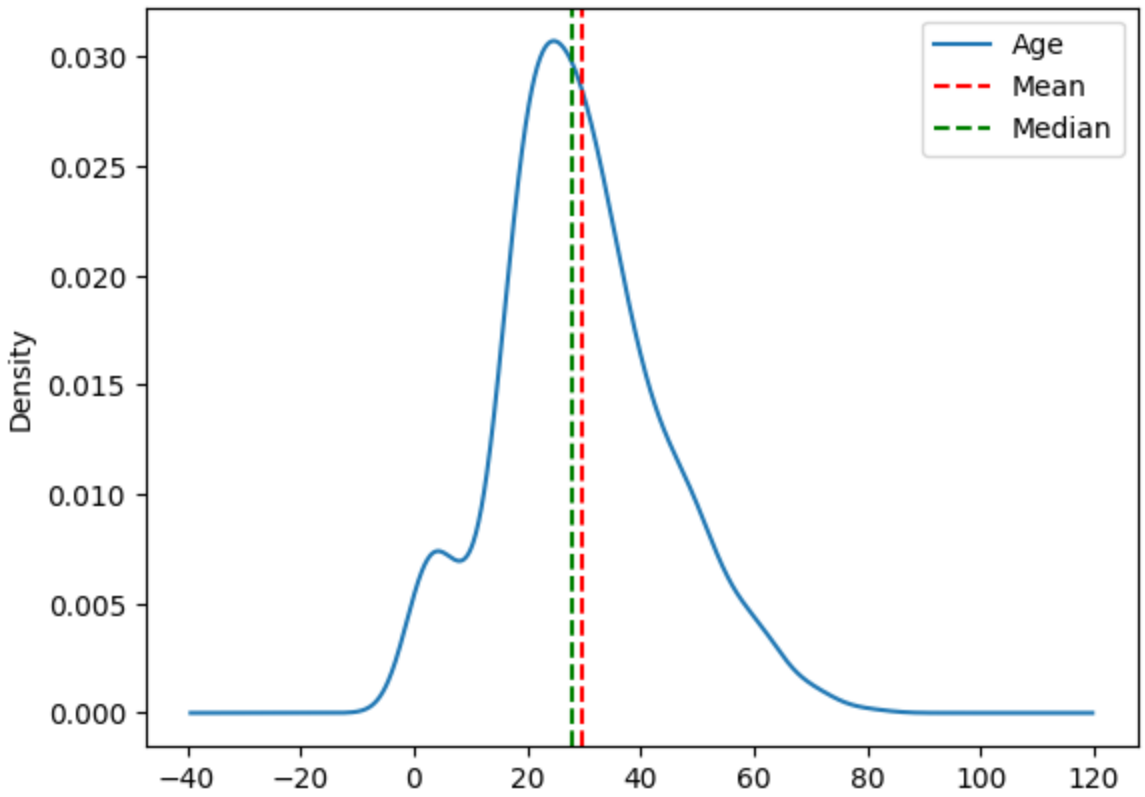
- x_i is each data point in the dataset
- μ is the mean of the dataset
- σ is the standard deviation of the dataset
- n is the number of data points




```
skewness = titanic['Age'].skew()
skewness

Output: 0.38910778230082704
```

- Positively skewed data
- Therefore, mean>median

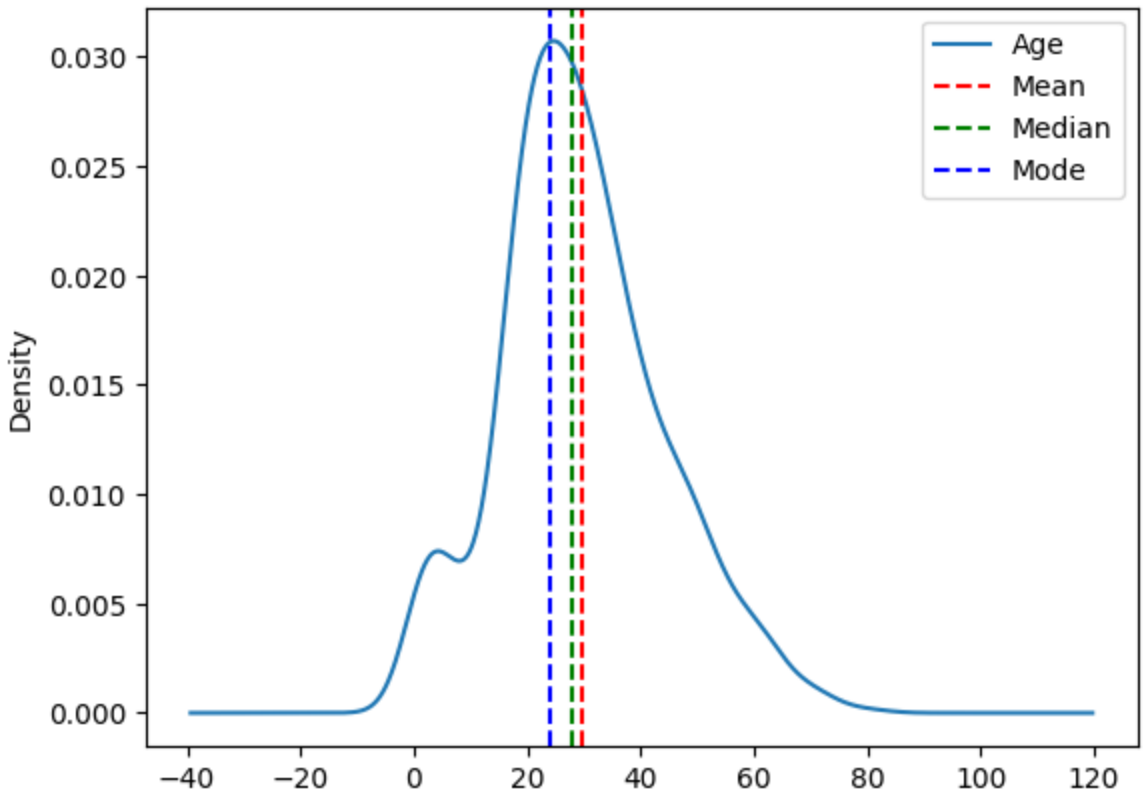
```
titanic['Age'].plot(kind='kde')
plt.axvline(titanic['Age'].mean(), color='red', linestyle='da
shed', label='Mean')
plt.axvline(titanic['Age'].median(), color='green', linestyle
='dashed', label='Median')
plt.legend()
```



- Mean is greater than median

- We can also plot the mode

```
titanic['Age'].plot(kind='kde')
plt.axvline(titanic['Age'].mean(), color='red', linestyle='da
shed', label='Mean')
plt.axvline(titanic['Age'].median(), color='green', linestyle
='dashed', label='Median')
plt.axvline(titanic['Age'].mode()[0], color='blue', linestyle
='dashed', label='Mode')
plt.legend()
```



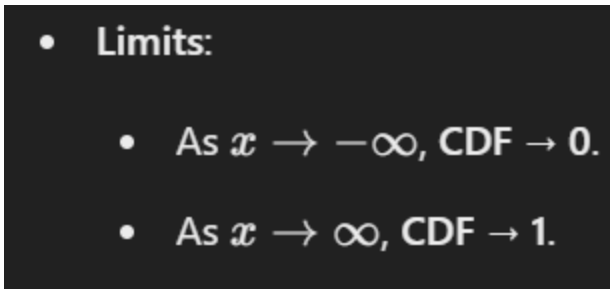
💡 **The greater the skew the greater the distance between mode, median and mean.**

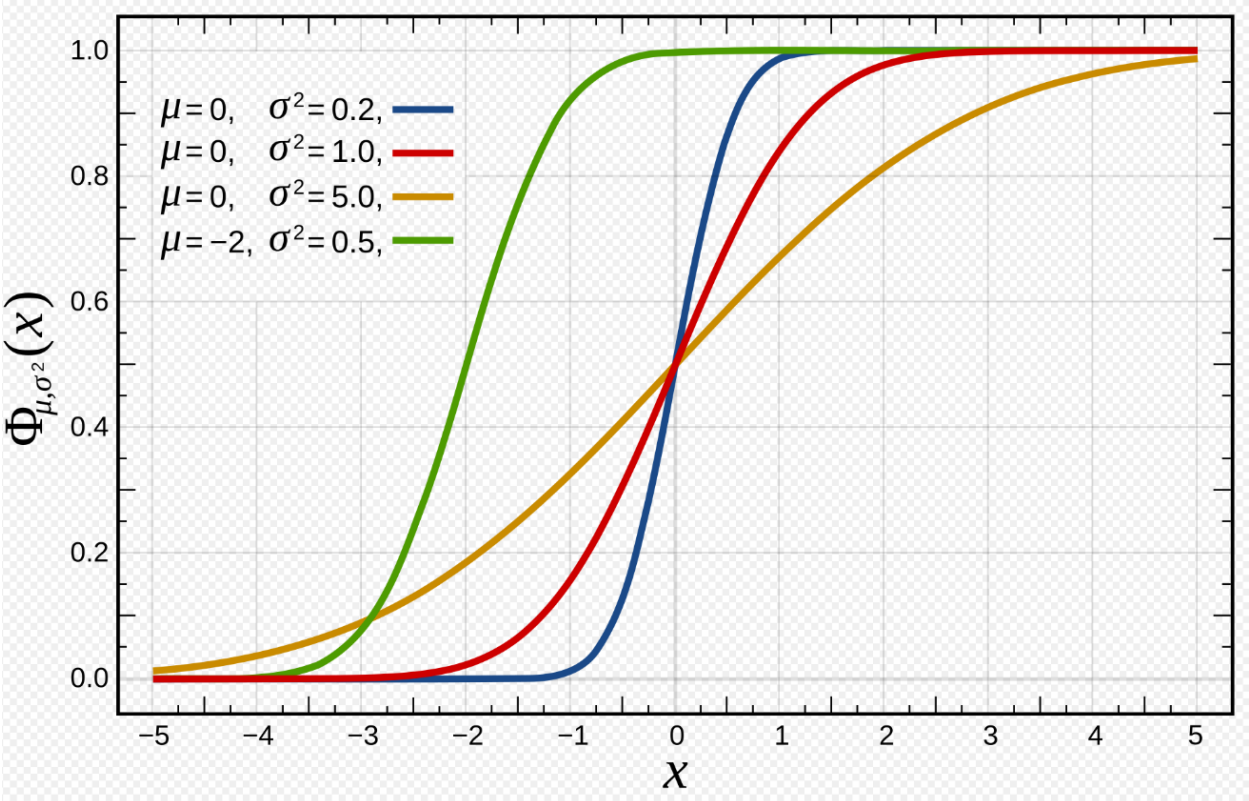
Note:

- This is also symmetric graph & its skewness will be close to 0.

CDF of Normal Distribution

- **Range**: The CDF value is always between **0 and 1** (or 0% to 100%).
- **Symmetry**: The CDF of a **standard normal distribution** N(0,1) satisfies: ***P(X≤0)=0.5***
    - This means that the probability of a value being **less than the mean** is always **50%**





Mathematical Definition

The CDF $F(x)$ for a normal distribution is:

$$F(x) = P(X \leq x) = \int_{-\infty}^{x} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt$$

For the **Standard Normal Distribution** ($\mu = 0, \sigma = 1$), denoted by $\Phi(z)$:

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{t^2}{2}} dt$$

- We perform integration of pdf, we get cfd.

In Python, you can use `scipy.stats.norm.cdf()` to calculate the CDF of a normal distribution.

```
from scipy.stats import norm

# Given values
mean = 100
std = 15
x = 110  # Value we want CDF for

# Calculate CDF
prob = norm.cdf(x, mean, std)

print(f"P(X ≤ {x}) = {prob:.4f}")  # Convert to percentage by
multiplying by 100
```

Output: P(X ≤ 110) = 0.7475

- You don't need to explicitly specify `loc=mu` and `scale=sigma` (`mean, std`) when μ **= 0 and** σ **= 1**, because these are the default parameters for the standard normal distribution in `scipy.stats.norm`.

Use in Data Science

- Outlier detection
- Assumptions on data for ML algorithms → Linear Regression and GMM
- Hypothesis Testing
- Central Limit Theorem

Outliers are considered the points above 3 SD

$$Outliers = mean + 3SD$$

- it's not a strict or universal rule for all data sets

```
titanic['Age'].mean() + 3*titanic['Age'].std()
```

Output: 73.27860964406094

- The ages above the above values will be outliers

```
titanic['Age'].mean() - 3*titanic['Age'].std()
```

- The ages below the above values will be outliers
  - But the can't be negative number
- So, find out people having age more than 75 years

```
titanic[titanic['Age'] > 73]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
630	631	1	1	Barkworth, Mr. Algernon Henry Wilson	male	80.0	0	0	27042	30.000	A23	S
851	852	0	3	Svensson, Mr. Johan	male	74.0	0	0	347060	7.775	NaN	S