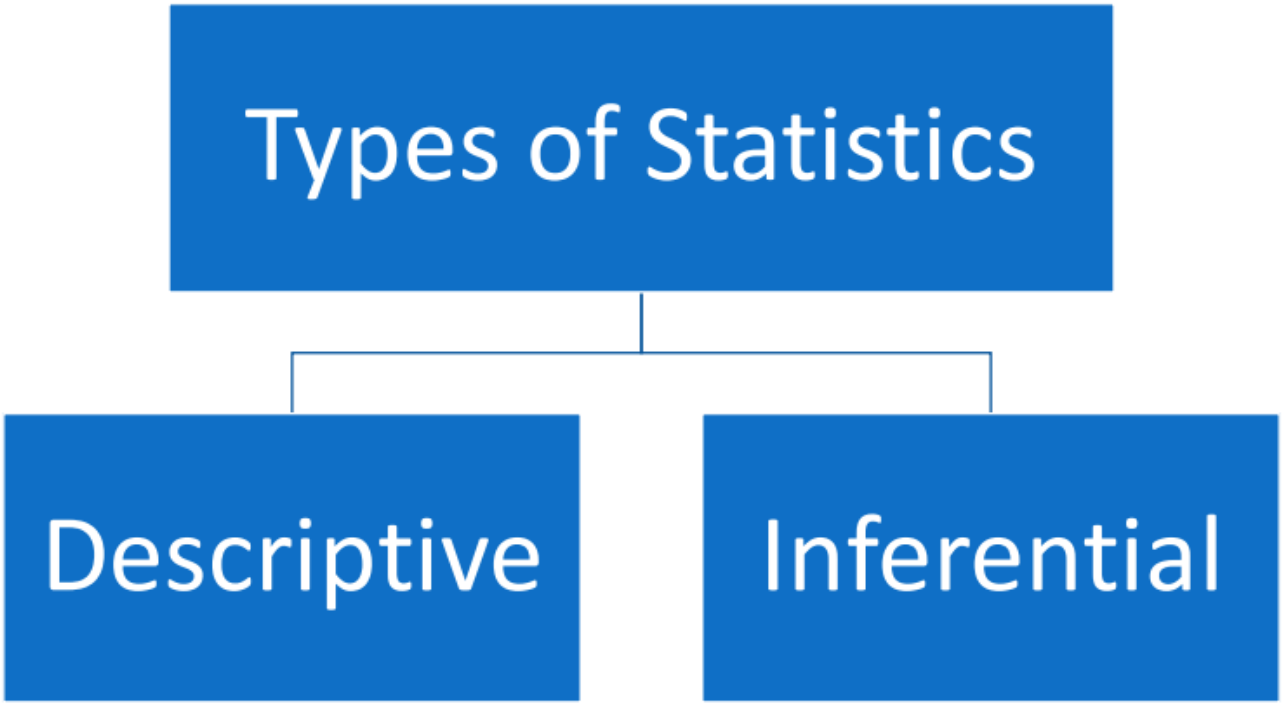
STATISTICS

STATISTICS is the discipline that concerns the collection, organization, analysis, interpretation, and presentation of Data.



DESCRIPTIVE STATISTICS

Descriptive statistics consists of methods for organizing, displaying, and describing data by using tables, graphs, and summary measures.

TOOLS

1. Measures of Central Tendency
    - Mean
    - Mode
    - Median
2. Measures of Dispersion
    - Range
    - Standard Deviation
3. Frequency Distributions
4. Histograms

INFERENTIAL STATISTICS

Inferential statistics consists of methods that use sample results to help make decisions or predictions about a population

TOOLS

1. Hypothesis testing
2. ANOVA
3. Chi-Squared Tests
4. Regression

Importing Libraries

In [117…
```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
import seaborn as sns
from scipy import stats
import matplotlib.pyplot as plt
import math
import random
import statistics as st
random.seed = 42
```

Dataset Details

In [118…
```
dataset = pd.read_csv('/content/data.csv')
dataset.head()
```

Out[118…

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_HH_Income
0	5000	8000	3	2000	64200
1	6000	7000	2	3000	79920
2	10000	4500	2	0	112800
3	10000	2000	1	0	97200
4	12500	12000	2	3000	147000

In [119…
```
dataset.info()
dataset.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   Mthly_HH_Income         50 non-null     int64 
 1   Mthly_HH_Expense        50 non-null     int64 
 2   No_of_Fly_Members       50 non-null     int64 
 3   Emi_or_Rent_Amt         50 non-null     int64 
 4   Annual_HH_Income        50 non-null     int64 
 5   Highest_Qualified_Member  50 non-null     object
 6   No_of_Earning_Members   50 non-null     int64 
dtypes: int64(6), object(1)
memory usage: 2.9+ KB
```

Out[119…
```
(50, 7)
```

In [120…
```
dataset.describe()
```

Out[120…

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_HH_Inco

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_HH_Inco
count	50.000000	50.000000	50.000000	50.000000	5.000000e-
mean	41558.000000	18818.000000	4.060000	3060.000000	4.900190e-
std	26097.908979	12090.216824	1.517382	6241.434948	3.201358e-
min	5000.000000	2000.000000	1.000000	0.000000	6.420000e-
25%	23550.000000	10000.000000	3.000000	0.000000	2.587500e-
50%	35000.000000	15500.000000	4.000000	0.000000	4.474200e-
75%	50375.000000	25000.000000	5.000000	3500.000000	5.947200e-
max	100000.000000	50000.000000	7.000000	35000.000000	1.404000e-

Mean

The mean is the most widely spread measure of central tendency. It is the simple average of the dataset.

Simply we can say that Sum of all observations by total no of Observations

Note: easily affected by **outliers**

The formula to calculate the Mean is:

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n}$$

or

$$\mu = \frac{x_1 + x_2 + x_3 + x_4 + ... + x_n}{n}$$

Where

μ is denoted by Mean

n is total number of Observations.

To Understand **Mean** in much better let's take an example

1. Mean of 104, 98, 90, 104, 104
2. $\frac{(104+98+90+104+104)}{5} = 100$

In **python Mean** is calculated using

=> *np.mean(dataset[´x´])*

where x is the column name of the dataset.

Without Libraries

In [121…

```
for column in dataset.columns:
    k=0
```

```
  meann=0
  new=list(dataset[column])
  for i in range(len(new)):
    if type(new[i]) == str:
      k=1
  if k!=1:
    for i in range(len(new)):
      meann=meann+new[i]
    meann=meann/len(new)
    print(column + "---> Mean is", meann)
```

```
Mthly_HH_Income---> Mean is 41558.0
Mthly_HH_Expense---> Mean is 18818.0
No_of_Fly_Members---> Mean is 4.06
Emi_or_Rent_Amt---> Mean is 3060.0
Annual_HH_Income---> Mean is 490019.04
No_of_Earning_Members---> Mean is 1.46
```

With Libraries

In [122…

```
for column in dataset.columns:
  k=0
  new=list(dataset[column])
  for i in range(len(new)):
    if type(new[i]) == str:
      k=1
  if k!=1:
    print(column + "---> Mean is", np.mean(dataset[column]))
```

```
Mthly_HH_Income---> Mean is 41558.0
Mthly_HH_Expense---> Mean is 18818.0
No_of_Fly_Members---> Mean is 4.06
Emi_or_Rent_Amt---> Mean is 3060.0
Annual_HH_Income---> Mean is 490019.04
No_of_Earning_Members---> Mean is 1.46
```

Median

The median is the midpoint of the ordered dataset. It is not as popular as the mean, but is often used in academia and data science. That is since it is not affected by outliers.

In an ordered dataset, the median is the number at position $\frac{n+1}{2}$

If this position is not a whole number, it, the median is the simple average of the two numbers at positions closest to the calculated value i.e mean of $(\frac{n}{2})^{th} value$ and $(\frac{n+1}{2})^{th} value$

To Understand **Median** in much better let's take an example

example-1:

1. Median of 104, 98, 90, 104, 104
2. Arranged in ascending order: => $90, 98, 104, 104, 104$
3. Here 104 is the middle value of the ordered dataset, So mode is **104**

Example-2:

1. Median of 104, 98, 90, 104, 104, 85
2. Arranged in ascending order: => $85, 90, 98, 104, 104, 104$
3. $\frac{98+104}{2} = \frac{202}{2} = 101$

In **python Median** is calculated using

$$\Rightarrow np.median(dataset[\acute{}x\acute{}])$$

where x is the column name of the dataset.

Without Libraries

In [123…
```
k=len(dataset['No_of_Fly_Members'])
dataset['No_of_Fly_Members'].sort_values()
if k % 2 == 0:
   mediann = ((dataset['No_of_Fly_Members'][k/2]) + (dataset['No_of_Fly_Members'][(k/2)+1
else:
   mediann = dataset['No_of_Fly_Members'][k/2]
print("median without libraries:",mediann)
```

```
median without libraries: 4.0
```

In [123…
```

```

With Libraries

In [124…
```
np.median(dataset['No_of_Fly_Members'])
```

Out[124…
```
4.0
```

Mode

The mode is the value that occurs most often. A dataset can have 0 modes, 1 mode or multiple modes. The mode is calculated simply by finding the value with the highest frequency

To understand **Mode** in much better let's take an example

1. Mode of 104, 98, 90, 104, 104
2. Here 104 is occurred three times in the data
3. So, Mode=104

In **python Mode** is calculated using

$$\Rightarrow statistics.mode(dataset[\acute{}x\acute{}])$$

where x is the column name of the dataset.

Without Libraries

In [125…
```
for column in dataset.columns:
  new=list(dataset[column])
  maxx=-10
  modee=0
  for i in range(len(new)):
    count=0
    for j in range(i,len(new)):
      if new[i]==new[j]:
        count+=1
    if count > maxx:
      modee=new[i]
      maxx=count
```

```
  print(column + "---> Mode is", modee)
  new.clear()
```

```
Mthly_HH_Income---> Mode is 45000
Mthly_HH_Expense---> Mode is 25000
No_of_Fly_Members---> Mode is 4
Emi_or_Rent_Amt---> Mode is 0
Annual_HH_Income---> Mode is 590400
Highest_Qualified_Member---> Mode is Graduate
No_of_Earning_Members---> Mode is 1
```

With Libraries

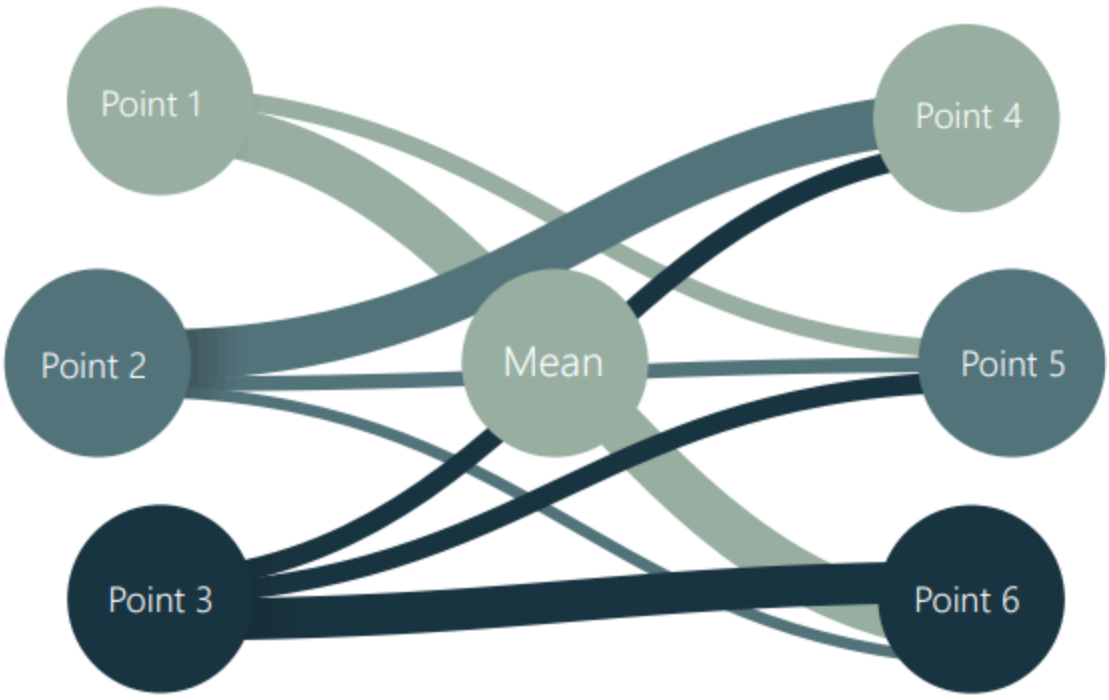
In [126…

```
for column in dataset.columns:
  print(column + "---> Mode is", st.mode(dataset[column]))
```

```
Mthly_HH_Income---> Mode is 45000
Mthly_HH_Expense---> Mode is 25000
No_of_Fly_Members---> Mode is 4
Emi_or_Rent_Amt---> Mode is 0
Annual_HH_Income---> Mode is 590400
Highest_Qualified_Member---> Mode is Graduate
No_of_Earning_Members---> Mode is 1
```

Variance and Standard Deviation

Variance and standard deviation measure the dispersion of a set of data points around its mean value. Standard Deviation can be represented as the square root of the variance and it denoted by the symbol "σ ".



There are different formulas for population and sample variance & standard deviation. This is due to the fact that the sample formulas are the unbiased estimators of the population formulas.

Sample Standard Deviation:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n-1}}$$

Population Standard Deviation:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n}}$$

In **python Standard Deviation** is calculated using

$$=> np.std(dataset[\acute{}x\acute{}])$$

where x is the column name of the dataset

To Understand **Standard Deviation** in much better let's take an example

Standard Deviation of 104, 98, 90, 104, 104

1. Find the average (mean) = 100
2. How far each item is from mean (104-100), (98-100), (90-100), (104-100), (104-100)
   $=> 4, -2, -10, 4, 4$
3. Take square of the distance from mean (4)2 , (-2)2 , (-10)2 , (4)2 , (4)2 $=> 16, 4, 100, 16, 16$
4. Take the mean of these squares (16+4+100+16+16)/5 = 30.4 (This is the Variance)
5. Square root of variance is the standard deviation

$$\sqrt{(30.4)} = 5.51$$

Without Libraries

In [127…

```
for column in dataset.columns:
  k=0
  meann=0
  summ=0
  new=list(dataset[column])
  for i in range(len(new)):
    if type(new[i]) == str:
      k=1
  if k!=1:
    for i in range(len(new)):
      meann=meann+new[i]
    meann=meann/len(new)
    for i in range(len(new)):
      summ=summ + ((new[i]-meann)**2)
    summ=summ/(len(new))
    sd=math.sqrt(summ)
    print(column + "---> Standard deviation is", sd)
```

```
Mthly_HH_Income---> Standard deviation is 25835.611779092826
Mthly_HH_Expense---> Standard deviation is 11968.704023410388
No_of_Fly_Members---> Standard deviation is 1.502131818450032
Emi_or_Rent_Amt---> Standard deviation is 6178.705366013175
Annual_HH_Income---> Standard deviation is 316918.26531451027
No_of_Earning_Members---> Standard deviation is 0.72691127381545
```

With Libraries

```
  k=0
  new=list(dataset[column])
  for i in range(len(new)):
    if type(new[i]) == str:
      k=1
  if k!=1:
    print(column + "---> Standard Deviation is", np.std(dataset[column]))
```

```
Mthly_HH_Income---> Standard Deviation is 25835.611779092826
Mthly_HH_Expense---> Standard Deviation is 11968.704023410388
No_of_Fly_Members---> Standard Deviation is 1.502131818450032
Emi_or_Rent_Amt---> Standard Deviation is 6178.705366013175
Annual_HH_Income---> Standard Deviation is 316918.26531451027
No_of_Earning_Members---> Standard Deviation is 0.72691127381545
```

Correlation

Correlation is a measure of the joint variability of two variables. Unlike covariance, correlation could be thought of as a standardized measure. It takes on values between -1 and 1, thus it is easy for us to interpret the result.

1. A correlation of 1, known as perfect positive correlation, means that one variable is perfectly explained by the other.
2. A correlation of 0 means that the variables are independent.
3. A correlation of -1, known as perfect negative correlation,means that one variable is explaining the other one perfectly, but they move in opposite directions.

Correlation:

$$r_{xy} = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

In other words:

$$r_{xy} = \frac{Covarience\ of\ X\ and\ Y}{(Standard\ Deviation\ of\ X) * (Standard\ Deviation\ of\ Y)}$$

Sample correlation formula:

$$r = \frac{S_{xy}}{S_x S_y}$$

Population correlation formula:

$$r = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

In **python Correlation** is calculated using

$$\mathbf{=> np.corrcoef(dataset[\acute{}x\acute{}], dataset[\acute{}y\acute{}])[0, 1]}$$

Where x and y are column names of the dataset.

Without Libraries

In [129…

```
mean_1=np.mean(dataset['Mthly_HH_Expense'])
                                come'])
```

```
x = list(dataset['Mthly_HH_Expense'])
y = list(dataset['Annual_HH_Income'])
num,den_1,den_2 = 0,0,0
for i in range(len(x)):
   num = num + ((x[i]-mean_1)*(y[i]-mean_2))
   den_1 = den_1 + ((x[i]-mean_1)**2)
   den_2 = den_2 + ((y[i]-mean_2)**2)
   den = math.sqrt((den_1)*(den_2))
r=num/den
print("Correlation is ", r)
```

```
Correlation is  0.5912216295468027
```

With Libraries

In [130…
```
np.corrcoef(dataset['Mthly_HH_Expense'],dataset['Annual_HH_Income'])[0,1]
```

Out[130…
```
0.5912216295468026
```

Normal Distrubution

A normal distribution is an arrangement of a data set in which most values cluster in the middle of the range and the rest taper off symmetrically toward either extreme. Height is one simple example of something that follows a normal distribution pattern: Most people are of average height

Normal Distribution is also known as **Parametric Distribution** because it is dependent on Mean μ and Variance σ^2

It is represented as $X\backsim$

Normal Distribution is calculated by $y = \frac {1}{\sigma \sqrt {2 \pi}}e^\frac {-(x - \mu)^2}{2 \sigma^2}$

In python **Normal Distribution** is calculated using

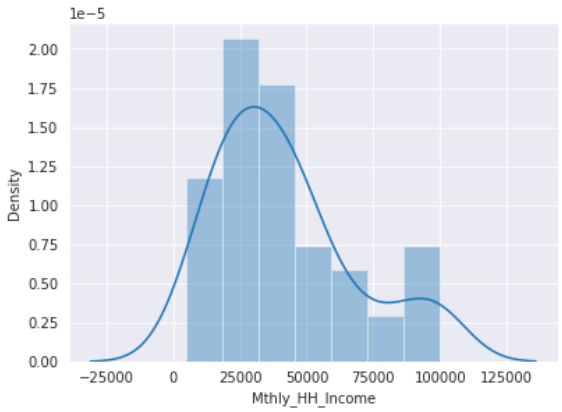
=>sns.distplot(dataset['x'])

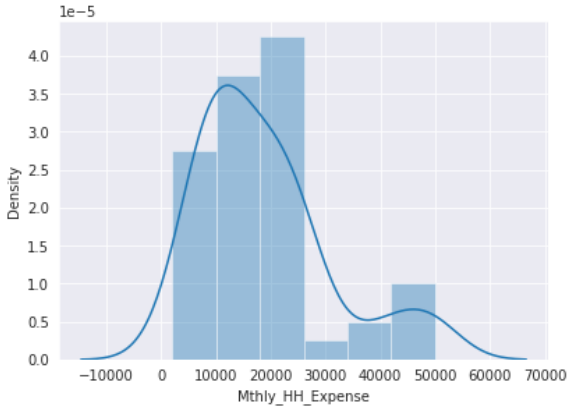
To understand this concept in detail lets go through the Normal Distribution graphs then we will go through the properties.so we can verify the graphs whether it is following Normal Distribution or not

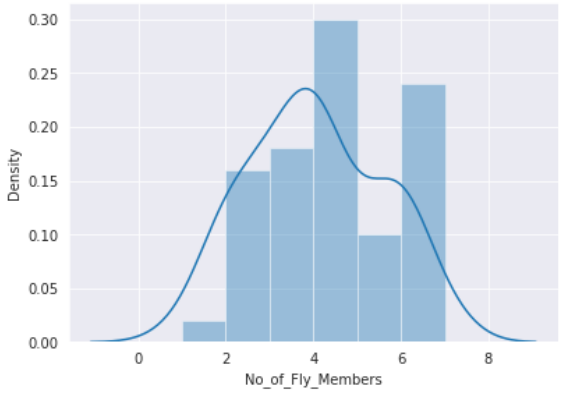
Univariate Normal Distribution Plots

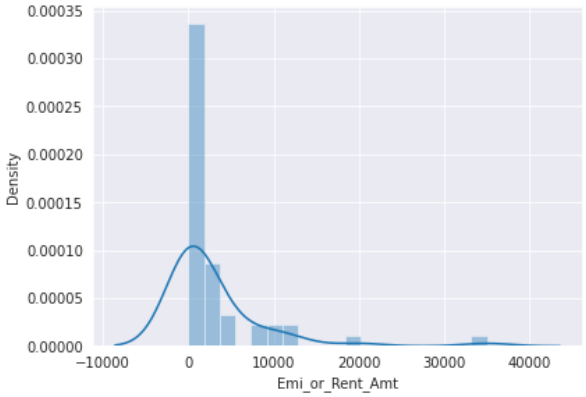
In [146…
```
sns.set_style("darkgrid")
plt.figure(figsize=(17, 17))
plt.subplots_adjust(hspace=0.4, wspace=0.7)
plt.subplot(3, 2, 1)
sns.distplot(dataset['Mthly_HH_Income'])
plt.subplot(3, 2, 2)
sns.distplot(dataset['Mthly_HH_Expense'])
plt.subplot(3, 2, 3)
sns.distplot(dataset['No_of_Fly_Members'])
plt.subplot(3, 2, 4)
sns.distplot(dataset['Emi_or_Rent_Amt'])
plt.subplot(3, 2, 5)
sns.distplot(dataset['Annual_HH_Income'])
plt.subplot(3, 2, 6)
Members'])
```

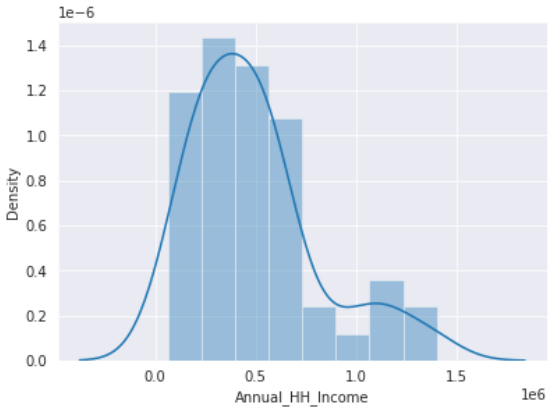
Out[146… <matplotlib.axes._subplots.AxesSubplot at 0x7f7c69fa4e90>









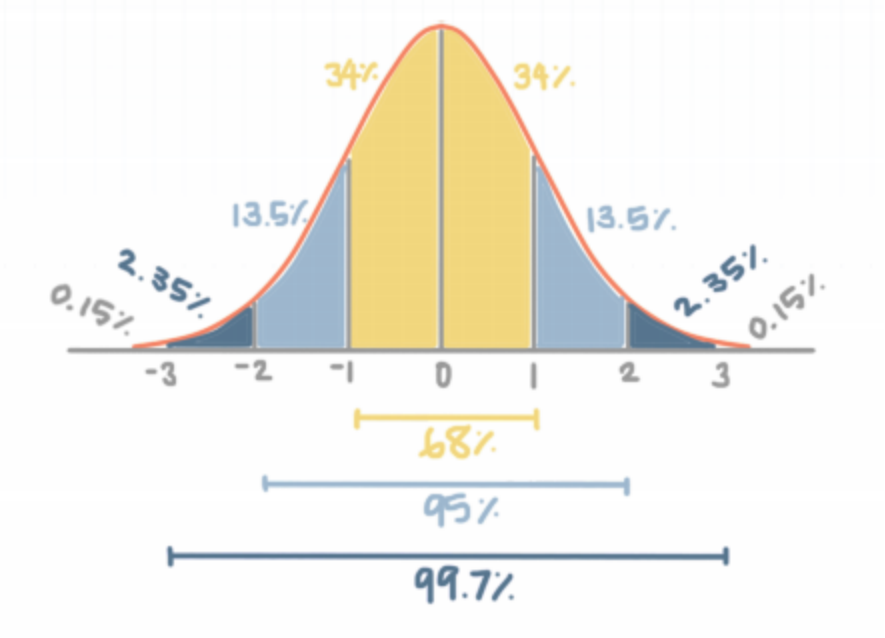


Features Of Normal Distribution

The Empirical Rule

Normal distributions follow the empirical rule, also called the 68-95-99.7 rule. The rule tells us that, for a normal distribution, there’s a

1. 68 % chance a data point falls within 1 standard deviation of the mean
2. 95 % chance a data point falls within 2 standard deviations of the mean
3. 99.7 % chance a data point falls within 3 standard deviations of the mean In other words, if we want to show this graphically



Bell Shaped Curve

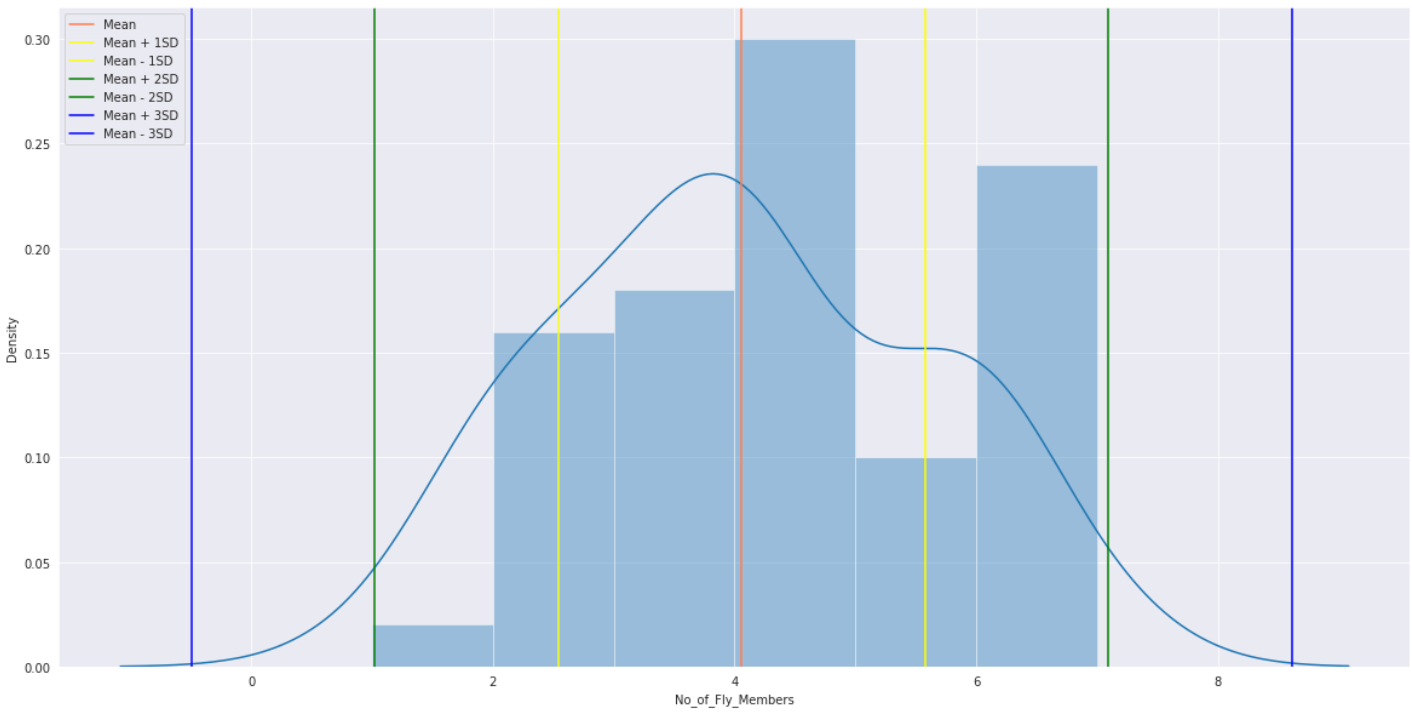
Mean and Standard Deviation defines the shape of the curve. It should be symmentrical.And MEAN=MODE=MEDIAN

In [132…
```
mu_normal_10000 = dataset['No_of_Fly_Members'].mean()

sigma_normal_10000 = dataset['No_of_Fly_Members'].std()

print(mu_normal_10000, sigma_normal_10000)
```

```
4.06 1.5173822786601394
```

In [133…
```
one_std_right = mu_normal_10000 + (1 * sigma_normal_10000)

one_std_left = mu_normal_10000 - (1 * sigma_normal_10000)

two_std_right = mu_normal_10000 + (2 * sigma_normal_10000)

two_std_left = mu_normal_10000 - (2 * sigma_normal_10000)

three_std_right = mu_normal_10000 + (3 * sigma_normal_10000)

three_std_left = mu_normal_10000 - (3 * sigma_normal_10000)
```

In [134…
```
plt.figure(figsize=(20,10))
sns.set_style("darkgrid")
sns.distplot(dataset['No_of_Fly_Members'])

plt.axvline(mu_normal_10000, color='coral', label='Mean')

plt.axvline(one_std_right, color='yellow', label='Mean + 1SD')
plt.axvline(one_std_left, color='yellow', label='Mean - 1SD')
plt.axvline(two_std_right, color='green', label='Mean + 2SD')
plt.axvline(two_std_left, color='green', label='Mean - 2SD')
plt.axvline(three_std_right, color='blue', label='Mean + 3SD')
plt.axvline(three_std_left, color='blue', label='Mean - 3SD')
plt.legend();
```



65 - 95 - 99.7 Rule

In [135…
```
type(dataset['No_of_Fly_Members'])
```

Out[135…
```
pandas.core.series.Series
```

In [136…
```
dataset['No_of_Fly_Members'] < one_std_right
```

Out[136…
```
0      True
1      True
2      True
3      True
4      True
5      True
6      True
7      True
8      True
9      True
10     True
11    False
12     True
13    False
14     True
15     True
16     True
17    False
18     True
19     True
20     True
21     True
22    False
23     True
24     True
25     True
26     True
27     True
28     True
29     True
30    False
31    False
32     True
33     True
```

```
35    False
36     True
37     True
38    False
39     True
40    False
41     True
42     True
43     True
44     True
45    False
46     True
47    False
48     True
49    False
Name: No_of_Fly_Members, dtype: bool
```

In [137…
```
(dataset['No_of_Fly_Members'] < one_std_right).sum()
```

Out[137…
```
38
```

In [138…
```
((one_std_left < dataset['No_of_Fly_Members']) & (dataset['No_of_Fly_Members'] < one_std_
```

Out[138…
```
29
```

In [139…
```
((one_std_left < dataset['No_of_Fly_Members']) & (dataset['No_of_Fly_Members'] < one_std_
```

Out[139…
```
0.58
```

In [140…
```
((two_std_left < dataset['No_of_Fly_Members']) & (dataset['No_of_Fly_Members'] < two_std_
```

Out[140…
```
0.98
```

In [141…
```
((three_std_left < dataset['No_of_Fly_Members']) & (dataset['No_of_Fly_Members'] < three_
```

Out[141…
```
1.0
```

NOTE: From the above Analysis it fails the property of Normal Distribution i.e it is not satysfying **Empirical Rule**. So it is **Not** following **Normal Distribution**

Skewness

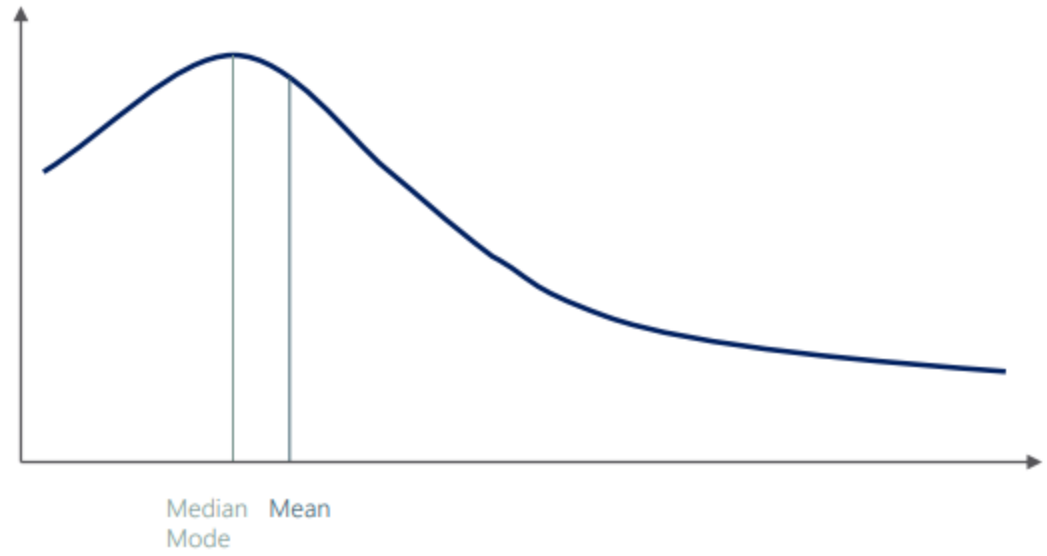
Skewness is a measure of asymmetry that indicates whether the observations in a dataset are concentrated on one side.

Usually, you will use software to calculate skewness

Positively Skewed Normal Distribution

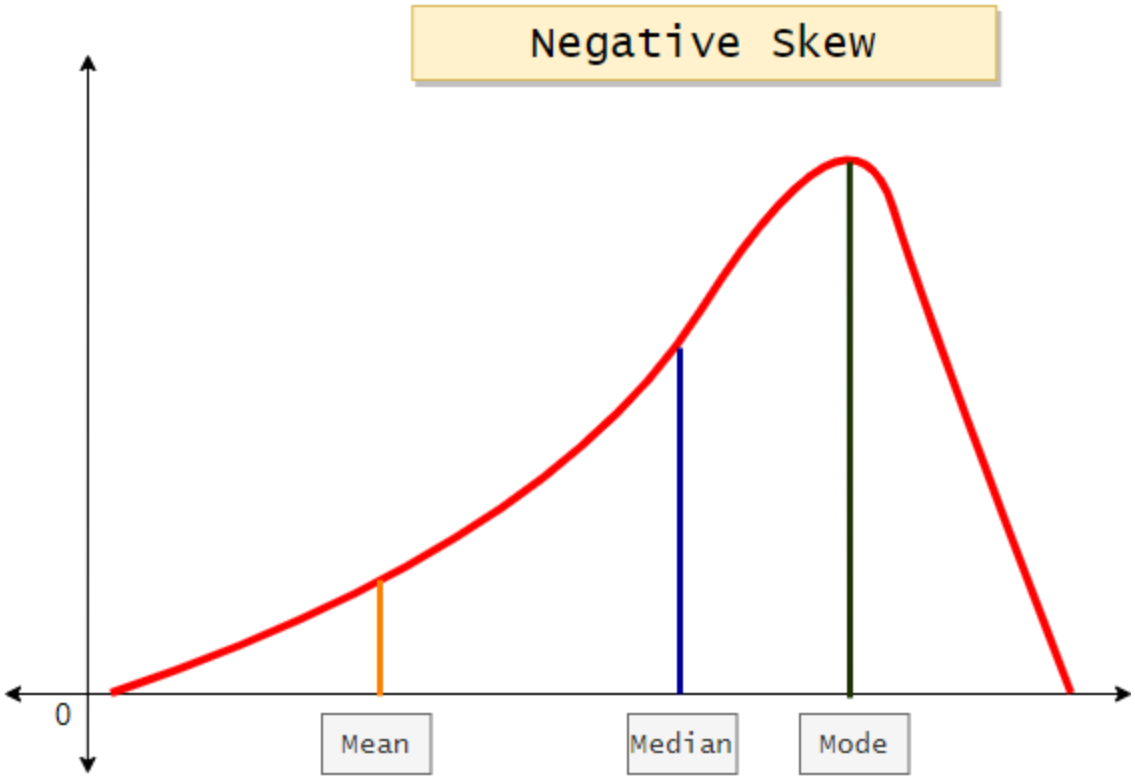
Right (positive) skewness looks like the one in the graph. It means that the outliers are to the right (long tail to the right).

MEAN > MEDIAN > MODE



Negatively Skewed Normal Distribution

Left (negative) skewness means that the outliers are to the left(long tail to the left) MODE > MEDIAN > MODE



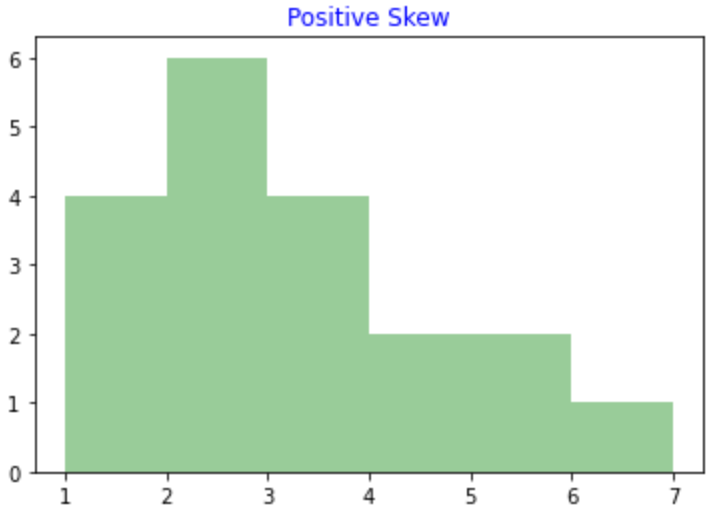
Effect on Mean, Median and Mode due to Skewness

To Understand Effect on mean, medain and mode in much better let's take an example

Positive Skew

lets we have dataset containing 1,1,1,1,2,2,2,2,2,2,3,3,3,3,4,4,5,5,7

This data set can be represented by following histogram. Each interval has width one, and each value is located in the middle of an interval.



The above data is not symmetrical

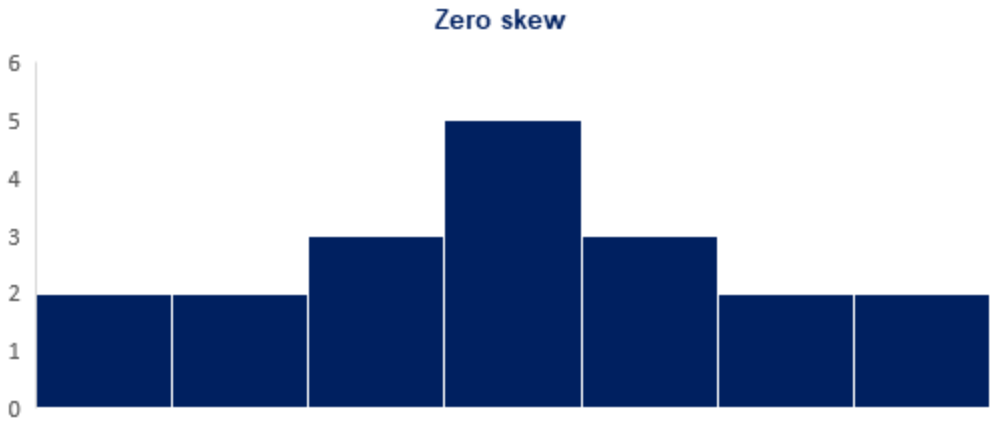
This Dataset is having Mean=2.79, Mode=2 and Median=2

This type of distribution is called **Positive Skew**

Note: Here mean is the largest than mode and median.

Zero Skew

Lets we have dataset containing 1,1,2,2,3,3,3,4,4,4,4,4,5,5,5,6,6,7,7



The above data is symmetrical

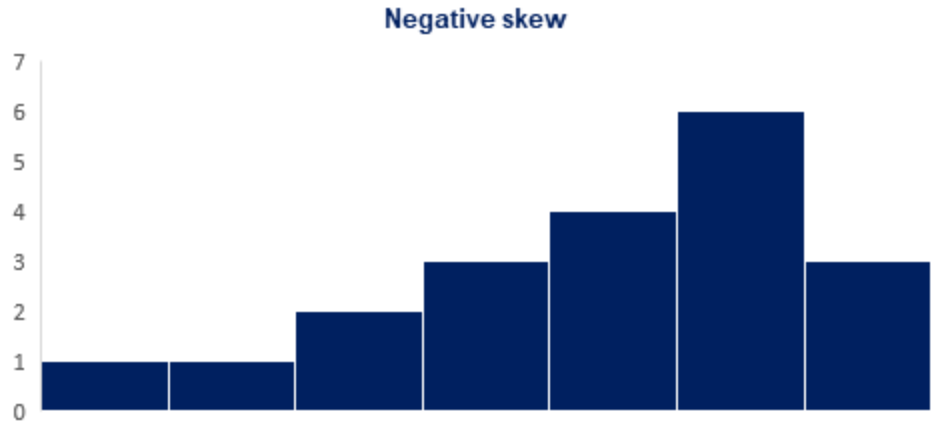
This Dataset is having Mean=4, Mode=4 and Median=4

This type of distribution is called **Zero Skew**

Note: Here mean, mode and medain are equal

Negative Skew

Lets we have dataset containing 1,2,3,3,4,4,4,5,5,5,5,6,6,6,6,6,6,7,7.



The above data is not symmetrical

This Dataset is having Mean=4.90, Median=5, Mode=6

This type of distribution is called **Negative Skew**

Note: Here Mode is the largest than Median than Mean.

QQ Plot

QQ Plot is also known as Quantile-Quantile Plot.

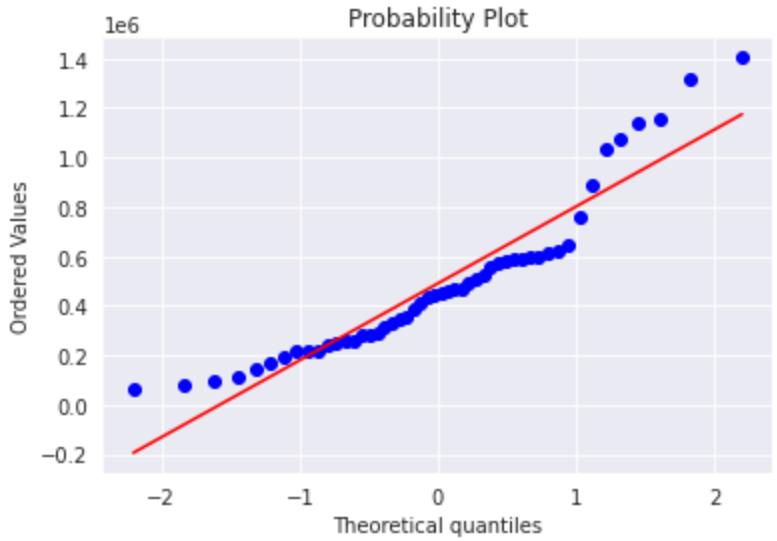
QQ plot is a way to test the normality of Distribution

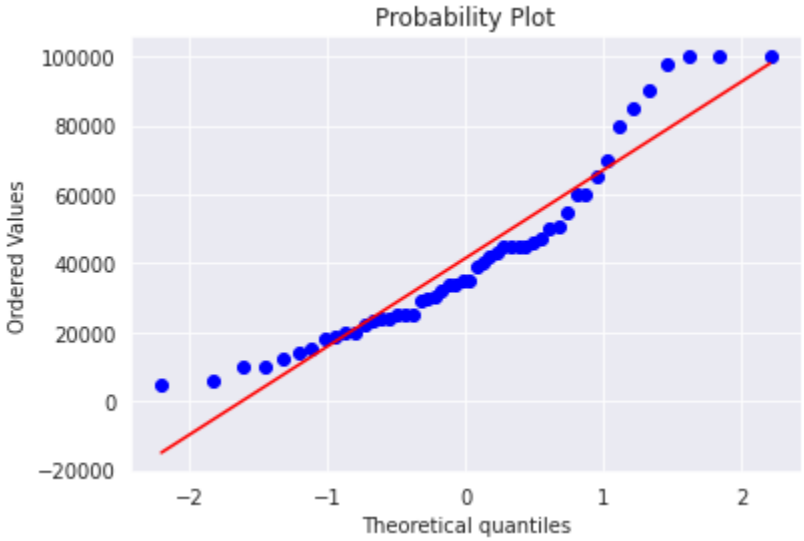
If Observed values is same as Theoritical Quantities then it follows standard normal distribution

Standard Normal Distribution means it follows normal distribution with mean=0 and Variance=1 $X \backsim N(0 , 1)$

In [142…

```
sns.set_style("darkgrid")
stats.probplot(dataset['Annual_HH_Income'], dist="norm", plot=plt)
plt.show()
stats.probplot(dataset['Mthly_HH_Income'], dist="norm", plot=plt)
plt.show()
```





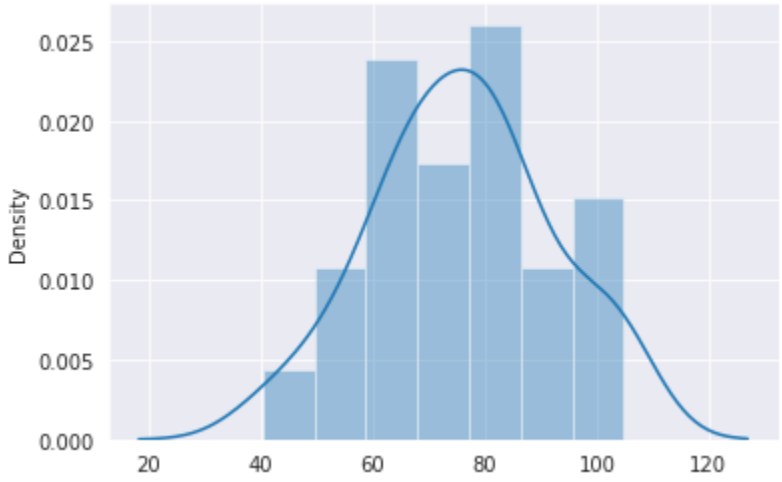
Box-Cox Transformation

It is used to transform data into Normal Distribution

For Lognormal and Pareto Distribution we use Box-Cox Transformation is get Normal Distribution.

NOTE: Generally Box-Cox Transformation is used only for Log Normal and Pareto Distribution.But it is used for other distributions, but it is not satisfying the properties of Normal Distribution in all the times

In [143…
```
norm,_ = stats.boxcox(dataset['Mthly_HH_Income'])
sns.distplot(norm, kde=True)
plt.show()
```



Pareto Distribution

The Pareto distribution is a skewed, heavy-tailed distribution or slowlying decaying tails, that is sometimes used to model the distribution of incomes and other financial variables.

In [144…
```
pareto_rv=np.random.pareto(a=4,size=6999)
sns.distplot(pareto_rv)
```

Out[144… `<matplotlib.axes._subplots.AxesSubplot at 0x7f7c75910610>`