Advanced Machine learning Mastering Course

Introduced by

George Samuel

Master in computer science
Cairo University

Innovisionray.com

2024

Advanced Machine Learning 2024 by George Samuel

Machine Learning Diploma

1- Statistics
2- Pandas

Agenda:

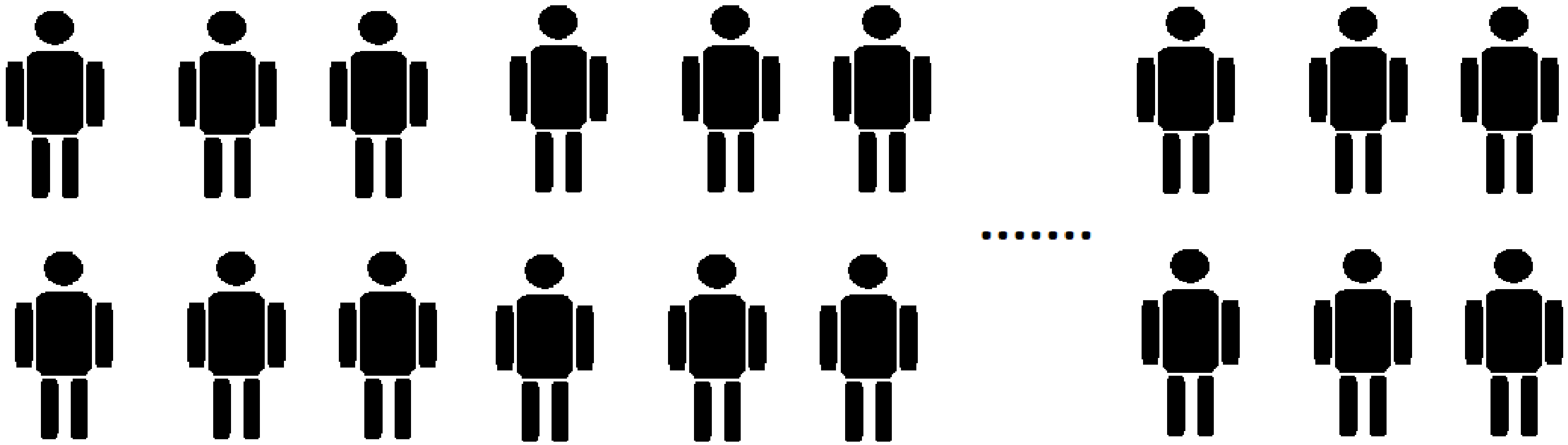
Agenda:

1	Pandas Basics
2	EDA Using Pandas
3	Data Manipulation
4	Indexing & Slicing
5	Inserting/Dropping DataFrame Columns & Rows
6	Null Values

1. Introduction to Statistics

What is Statistics?

- Statistics is the science of summarizing and describing the data.
- For example:
  - Suppose you have a dataset that contains about 100,000,000 observations about Egyptian people height.



What is Statistics?

- If you want to describe how high Egyptian people are, you don't tell the height of each single person of the 100,000,000 people in the Egyptian population! But instead, you simply say "The average height of the Egyptian people is 170cm".

- What you have just done is that you summarized the 100,000,000 observations into one number, 170cm, which we call a statistical measure.

2. Statistical Measures

Statistical Measures:

- A Statistical Measure is a number, that is calculated to summarize many records(rows) of information into one single value.
- Statistical measures can be used to get statistical inference about the population.
- Since statistical measures are related to data, let's first understand types of the data. Data can be:
  - Continuous(Numerical).
  - Or Discrete(Categorical).

Continuous Vs Discrete:

Continuous Data	Discrete Data

➢ Is the data that has infinite number of possible values.
 	
➢ Also known as **Numerical data**.
 	
➢ Continuous data could be:
  ➢ Float dtypes; such as, Salary or Weight.
  ➢ Int dtypes that have large number of possible unique values; such as, number-of-hours-played.
	➢ Is the data that has finite number of possible values
	
	➢ Also known as **Categorical data**.
	
	➢ Continuous data could be:
	  ➢ String dtypes; such as, City-name.
	  ➢ Int dtypes that have small number of possible unique values; such as, number-of-children.

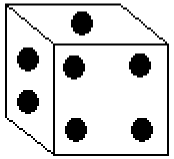
Popular Statistical Measures:

1. **Probability.**
2. **Measures of Central Tendency.**
3. **Measures of dispersion (Deviation).**

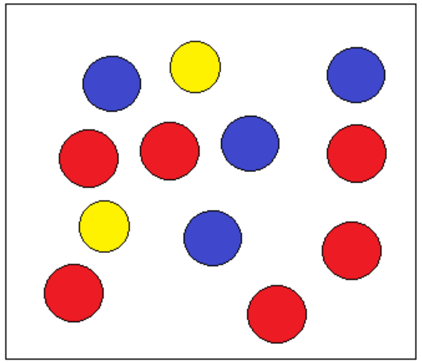
Probability:

- Is the ratio between frequency of the unique-value & total number of samples.

- Example1, suppose you have a dice:
  - The unique possible values are; 1, 2, 3, 4, 5, 6.
  - Probability of 1 = 1 / 6 = .167



- Example1, suppose you have the box of balls on the right:
  - The unique possible values are; blue, red, yellow.
  - Probability of blue = 4 / 12 = .333



Measures of Central Tendency:

- Are the measures used to represent the average values of the data we have.
- There are three main measures of central tendency:
  - Mean.
  - Median.
  - Model.
- Mean & Median are used to summarize Numerical data, while Mode is used to summarize categorical data.

Measures of Central Tendency (Mean):

- Mean is the ratio between the summation of all values and total number of observation in the data.
- For example, suppose you have the following set of observation:
  - [5, 2, 3, 10, 20].
  - Mean = (5+2+3+10+20) / 5 = 8.
- Mean is used with numerical data that doesn’t contain extreme values (outliers), because mean is sensitive to outliers.
- We use symbol μ to represent the mean.

Measures of Central Tendency (Median):

- Median is the middle value in the data after being sorted.
- Steps:
  - First sort the data, then Find the number in the middle, and this is your Median. If there are two number in the middle, then the Median is the average between them.
- Median is used with numerical data that contains outliers.

Example1	Example2
Suppose you have this set of observations: [5, 2, 3, 10, 20] . First sort them → [2, 3, 5, 10, 20]. Median = 5.	Suppose you have this set of observations: [3, 5, 2, 3, 10, 20] . First sort them → [2 , 3, 3, 5, 10, 20]. Median = (3+5) / 2 = 4.

Measures of Central Tendency (Mode):

- Mode is the most frequent value in the data.
- Mode is used with categorical data.

Example1	Example2
➢ Suppose you have this set of observations: [5, 2, 3, 3, 2, 3, 1, 5, 9, 8, 3, 1, 7, 6]. ➢ Mode= 5.	➢ Suppose you have this set of observations: ["Cairo", "Alex", "Aswan", "Alex", "Alex", "Mansoura", "Alex", "Cairo"]. ➢ Mode = "Alex".

Measures of Dispersion:

- Are measures used to measure the spread of the data.
- Also Called Measures of Deviation.
- For example, suppose you have the following two sets of numbers:
  - Set1 = [5, 5, 5, 5, 5] & Set2 = [-5, 0, 5, 10, 15].
  - The two sets contains the same value of mean = 5.
  - But as you can see Set2 has more spread than set1.
  - So, we need a way to measure the amount of spread.

Measures of Dispersion:

- There are two main measures of Dispersion:
  - Variance.
  - Standard deviation.
- Standard Deviation is the most used as a measure of dispersion, that’s why we call it standard, however variance is a popular measure too and has its applications.

Measures of Dispersion (Variance):

- Is the average of all differences between each value in the data & the mean of this data.
- $\sigma 2$ is used to represent the Variance.
- Formula: $\sigma^2 = \frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}$, where X_i represents the i^{th} value in the data, and N represents total number of values.

Measures of Dispersion (Variance):

Example1	Example2
➢ Data = [5, 5, 5, 5, 5] .	➢ Data = [-5, 0, 5, 10, 15].
➢ μ = (5+5+5+5+5) / 5 = 5.	➢ μ = (-5+0+5+10+15) / 5 = 5.
➢ σ2 = $((5-5)^2 + (5-5)^2 + (5-5)^2 + (5-5)^2 + (5-5)^2) / 5 = 0$.	➢ σ2 = $((5--5)^2 + (5-0)^2 + (5-5)^2 + (5-10)^2 + (5-15)^2) / 5 = 50$.
➢ Variance = 0	➢ Variance = 50.

Measures of Dispersion (Standard Deviation):

- Is the square root of the variance.
- σ is used to represent the Standard deviation.
- Formula: $\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}}$, where X_i represents the i^{th} value in the data, and N represents total number of values.
- Standard deviation is always preferred over variance as a measure of dispersion, and the reason is that unlike variance, standard deviation is not sensitive to outliers.

Measures of Dispersion (Standard Deviation):

Example1	Example2
➢ Data = [5, 5, 5, 5, 5] .	➢ Data = [-5, 0, 5, 10, 15].
➢ μ = (5+5+5+5+5) / 5 = 5.	➢ μ = (-5+0+5+10+15) / 5 = 5.
➢ $\sigma 2 = ((5-5)^2 + (5-5)^2 + (5-5)^2 + (5-5)^2 + (5-5)^2) / 5 = 0$.	➢ $\sigma 2 = ((5--5)^2 + (5-0)^2 + (5-5)^2 + (5-10)^2 + (5-15)^2) / 5 = 50$.
➢ $\sigma = \sqrt{\sigma 2} = \sqrt{0} = 0$.	➢ $\sigma = \sqrt{\sigma 2} = \sqrt{50} = 7.07$
➢ Standard deviation = 0.	➢ Standard deviation = 7.07

3. Population Vs Sample

What is Population?

- Population is the whole complete set of observation.
- For example:
  - In Egypt, we have 100,000,000 people if we could collect 100,000,000 observations about their heights, then the population = heights-of-100,000,000-people.
- But could we really collect this huge number of observations? Do we have the resources(money & time) to do this?!
- The answer is No! and here comes the concept of Sample.

What is Sample?

- A sample is a randomly chosen subset from the population, that represents the whole set of observations without having to actually deal with the whole population.
- For example:
  - In Egypt, we could represent the 100,000,000 people with only 1000,000 observations collected randomly.
- The larger the sample is, the more strongly it represents the population, but the harder to collect and work on.

4. Statistics using Pandas

What is Pandas?

- You can apply statistics using Numpy or Pandas.
- Pandas is a library built on Numpy, which is more suitable for dealing with tabular datasets.
- In Pandas tabular data is read as DataFrame which is the main datatype in pandas that represents matrix.
- In pandas, vectors are represented by a datatype called Series.
- Each row or column in the DataFrame is a Series.

Reading Tabular Data:

➢ Tabular datasets come in two main file formats:

CSV files

```python
import pandas as pd
df = pd.read_csv("file.csv")
df
```

	Length	Width	City	Price
0	20	10	Cairo	5000000
1	15	15	Alex	4000000
2	30	20	Aswan	1500000
3	10	50	Alex	8000000
4	5	15	Giza	800000
5	12	10	Alex	1000000
6	5	30	Luxor	500000
7	7	20	Aswan	700000
8	20	40	Alex	9000000
9	8	20	Cairo	900000

XLSX files

```python
import pandas as pd
df = pd.read_excel("file.xlsx")
df
```

	Length	Width	City	Price
0	20	10	Cairo	5000000
1	15	15	Alex	4000000
2	30	20	Aswan	1500000
3	10	50	Alex	8000000
4	5	15	Giza	800000
5	12	10	Alex	1000000
6	5	30	Luxor	500000
7	7	20	Aswan	700000
8	20	40	Alex	9000000
9	8	20	Cairo	900000
10	6	14	Giza	6000000

Pandas for Statistics:

Mean of Length Column	Median of Length Column	Mode of City Column
`df.Length.mean()` 12.545454545454545	`df.Length.median()` 10.0	`df.City.mode()` 0 Alex
Variance of Length Column	**Standard-Deviation of Length column**	
`df.Length.var()` 63.672727272727265	`df.Length.std()` 7.979519238195198	

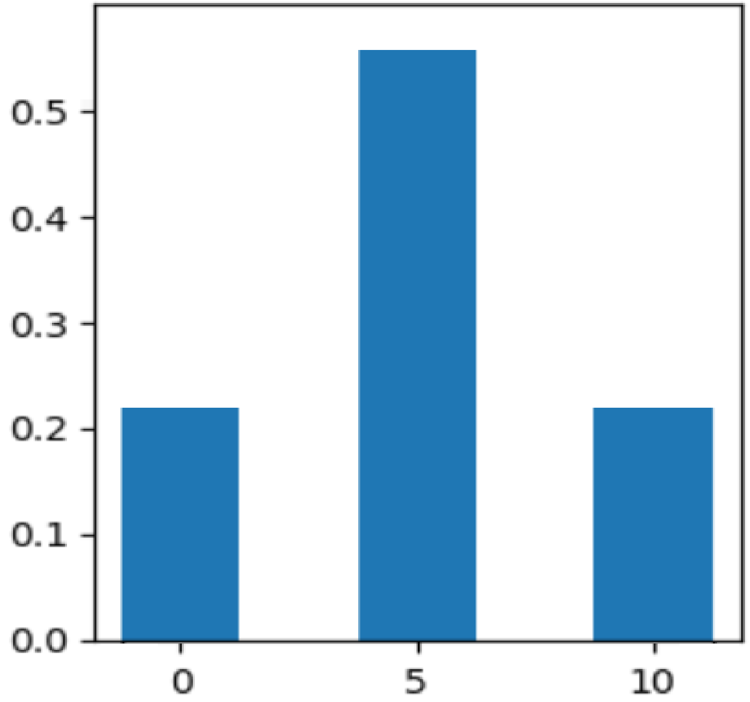
7. Data Distribution

What is Data Distribution?

- Data Distribution is a way to describes how the observations are distributed or spread across the unique values of the data.
- In other words, Data Distribution represents how much each unique value occurs in the data or how frequent each unique value is.

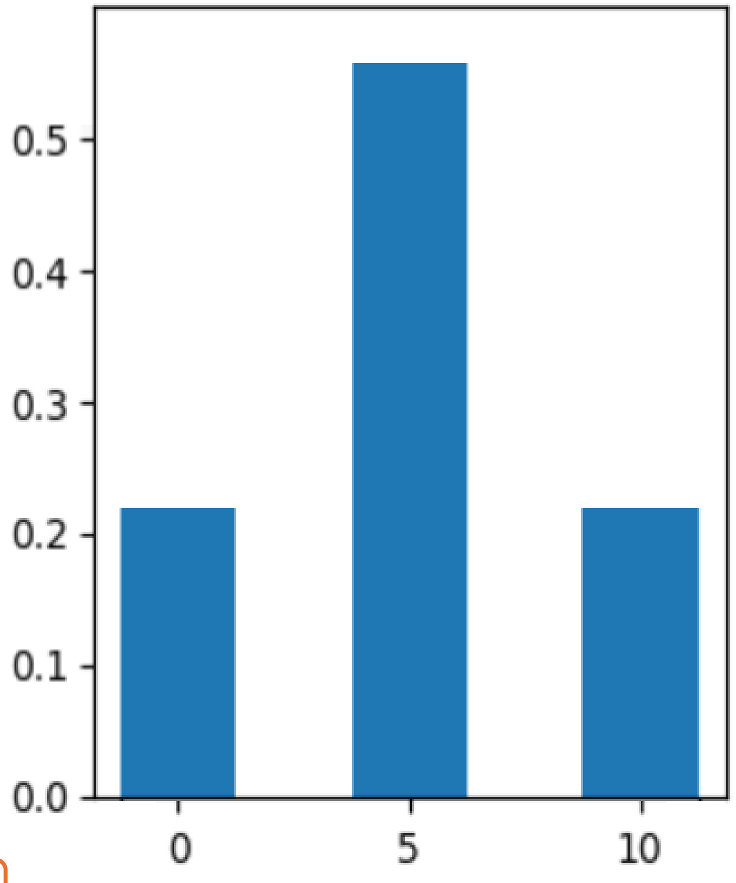
Data Distribution Example:

- If you have Random Variable X = [0, 5, 5, 5, 10, 0, 5, 10, 5].
- Then the data distribution of this random variable is distributed as following:
  - 22.2% of the data belong to (X=0).
  - 55.6% of the data belong to (X=5).
  - 22.2% of the data belong to (X=0).



Data Distribution Histogram:

- It's common to represent the data distribution as a graph called Histogram.
- A histogram is a 2-dimensional graph, where:
  - X-axis represents the unique values in the Random Variable.
  - Y-axis represents the probability of each unique value.
  - Each unique value has a bar (rectangle) whose height is equal to the probability.

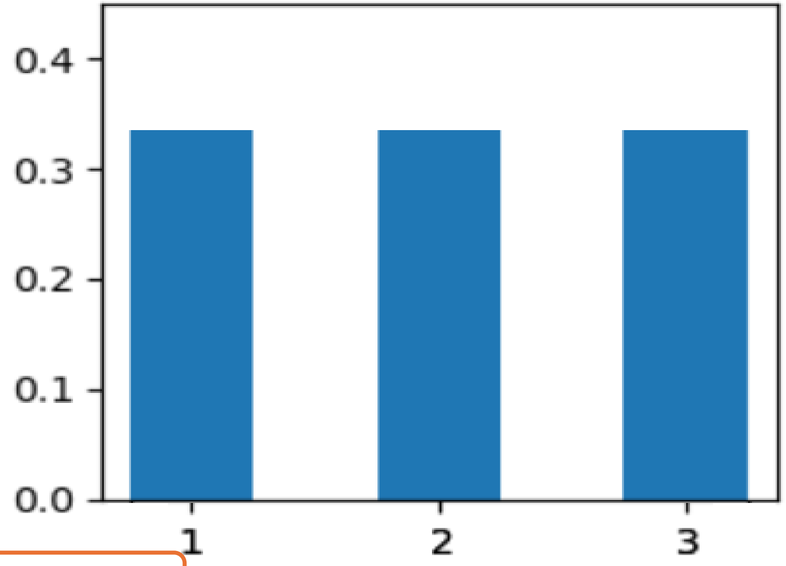


Data Distribution Types:

- There are so many types of data distribution, however we will cover the most important & most popular ones:
  - Uniform Distribution.
  - Normal Distribution.
  - Right-Skewed Distribution.
  - Left-Skewed Distribution.

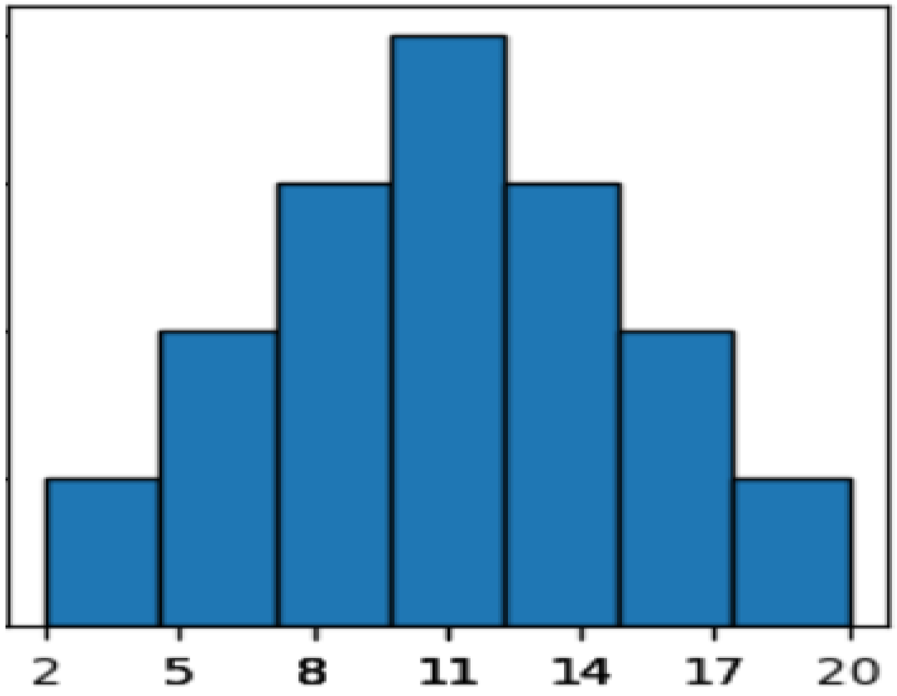
Uniform Distribution:

- Is Data Distribution where observations are equally distributed among the unique values. In other words, all the unique values occur equally with the same frequency.
- For example, Suppose you have X = [1, 2, 2, 3, 1, 3], then the distribution is:
  - 33.3% of the data belong to (X=1).
  - 33.3% of the data belong to (X=2).
  - 33.3% of the data belong to (X=3).



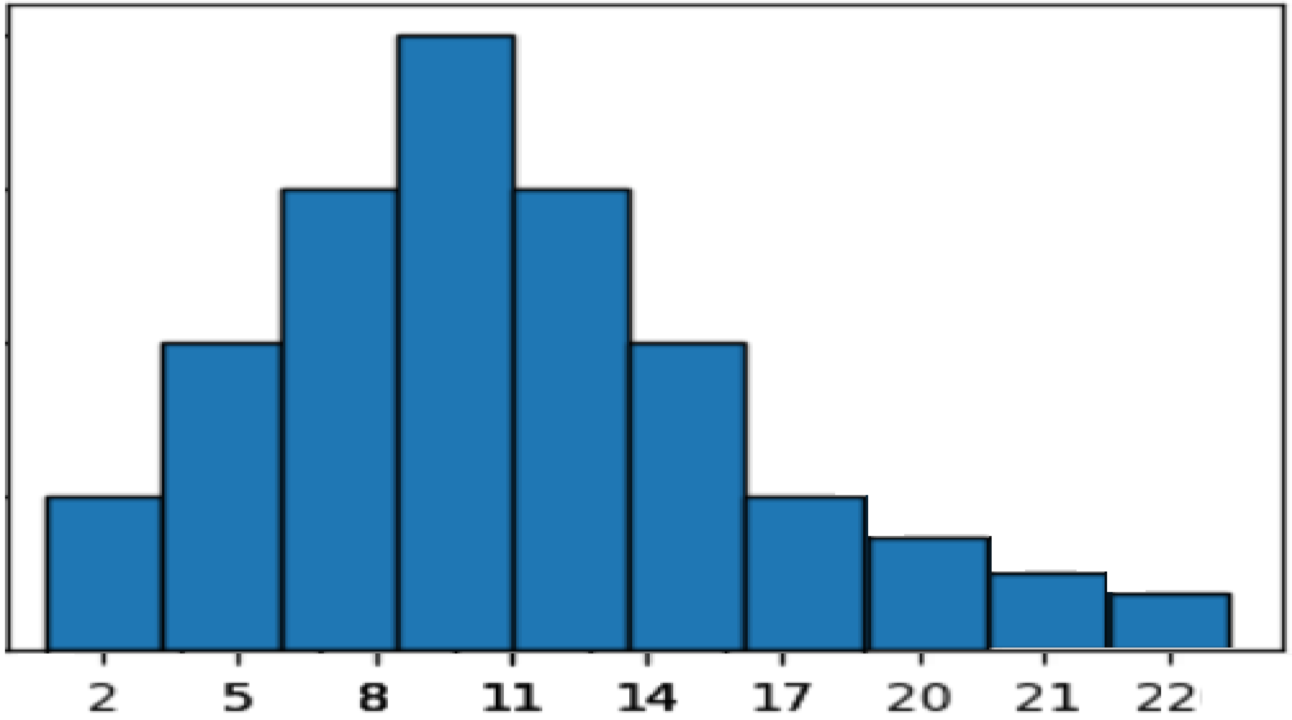
Normal Distribution:

- Is Data Distribution where observations are distributed around the mean the most, with fewer values occurring farther away from the mean in both directions.
- The distribution histogram takes a shape of symmetric bell.



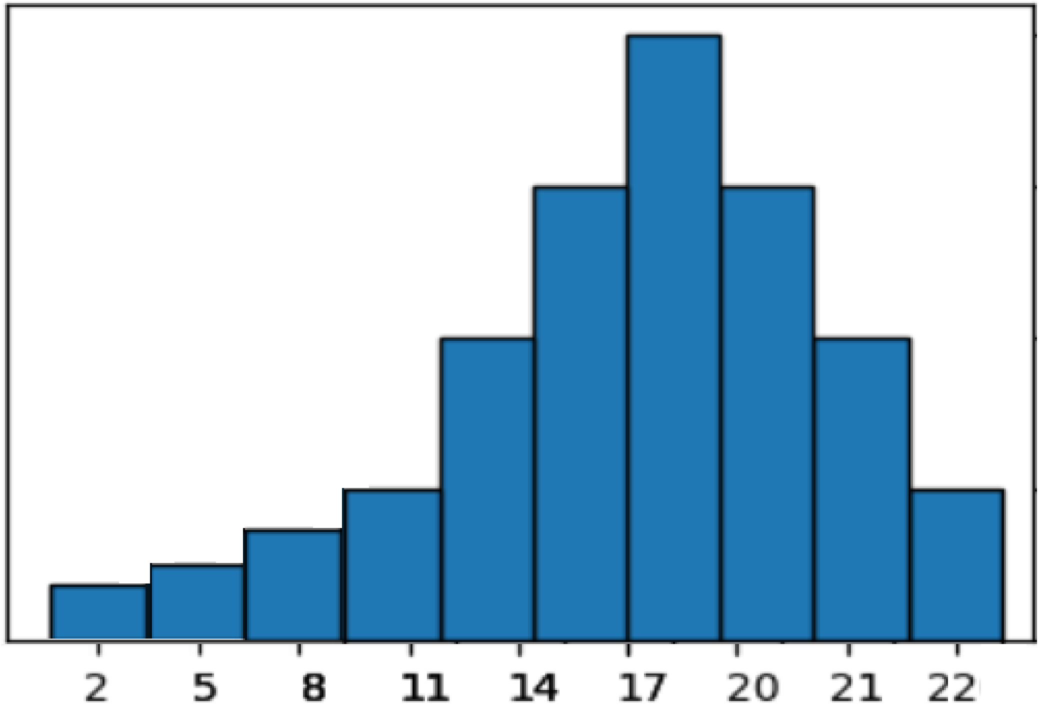
Right-Skewed Distribution:

- Is Data Distribution where observation are mostly distributed around mean and left side to the mean, with few observations at the extreme right to the mean.



Left-Skewed Distribution:

- Is Data Distribution where observation are mostly distributed around mean and right side to the mean, with few observations at the extreme left to the mean.



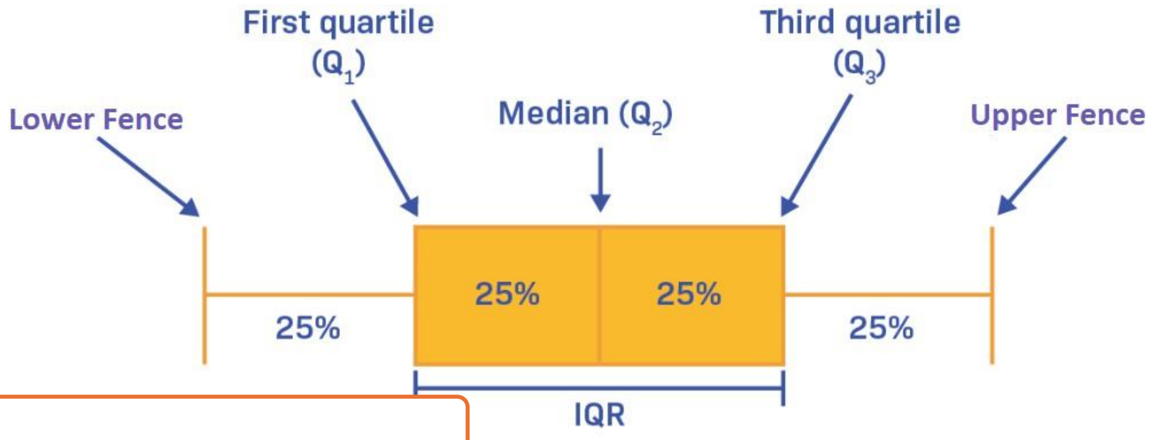
8. Quartiles

What are Quartiles?

- Is a technique used to identify outliers, which are extreme values that occur in the data.
- For example:
  - Suppose you have a random variable X=[20, 30, 10, 50, 180] where X represents people ages.
  - The value 180 is an outlier because it's a strange or extreme value, since it's no common to see a 180 years-old person.

What are Quartiles?

- Quartiles are numbers used to detect fences or thresholds, where if a number exceeds these fences, then this number is considered to be an outlier.
- There are three types of quartiles to calculate to be able to calculate the fences. These three quartiles are:
    - First Quartile (Q1).
    - Second Quartile (Q2).
    - Third Quartile (Q3).



How to Calculate Quartiles?

Steps:

1. Sort the Random Variable data.
2. Calculate the median of the Random Variable, and this is your Q2.
3. Calculate the median of the subset right to Q2, and this is your Q1.
4. Calculate the median of the subset left to Q2, and this is your Q3.

Calculate Quartiles Example:

90 33 47 -50 10 19 11 13 16 28 15 19 23 21 44 30 34 36 10 45

1- Sort: -50 10 10 11 13 15 16 19 19 21 23 28 30 33 34 36 44 45 47 90

2- Find Q2: -50 10 10 11 13 15 16 19 19 $\underbrace{21 \quad 23}_{Q2 = 22}$ 28 30 33 34 36 44 45 47 90

Q2 = 22

3- Find Q1 & Q3: -50 10 10 11 $\underbrace{13 \quad 15}_{Q1 = 14}$ 16 19 19 21 | 23 28 30 33 $\underbrace{34 \quad 36}_{Q3 = 35}$ 44 45 47 90

Q1 = 14 Q2 = 22 Q3 = 35

-50 10 10 11 13 | 15 16 19 19 21 | 23 28 30 33 34 | 36 44 45 47 90

Outlier fences:

- There are two fences we need to calculate so that if a number exceed these fences, then it is considered an outlier.
- These two fences are:
  - Upper Fence:
    - If a number is larger than the upper fence, then it is considered an outlier.
  - Lower Fence:
    - If a number is smaller than the lower fence, then it is considered an outlier.

How to Calculate Outlier fences?

- Steps:
  1. Calculate IQR, where IQR = Q3 – Q1.
  2. Calculate Lower-Fence where, Lower-Fence = Q1 – 1.5*IQR.
  3. Calculate Upper-Fence where, Upper-Fence = Q3 + 1.5*IQR.
- Example:

90 33 47 -50 10 19 11 13 16 28 15 19 23 21 44 30 34 36 10 45

Q1 = 14 Q2 = 22 Q3 = 35

-50 10 10 11 13 | 15 16 19 19 21 | 23 28 30 33 34 | 36 44 45 47 90

IQR = Q3 - Q1 = 35 - 14 = 21
Lower-Fence = Q1 - 1.5 * IQR = 14 - 1.5 * 21 = -17.5
Upper-Fence = Q3 + 1.5 * IQR = 35 + 1.5 * 21 = 66.5

-50 is an outlier, because it is < Lower-Fence ==> (-50 < -17.5)
90 is an outlier, because it is > Upper-Fence ==> (90 > 66.5)

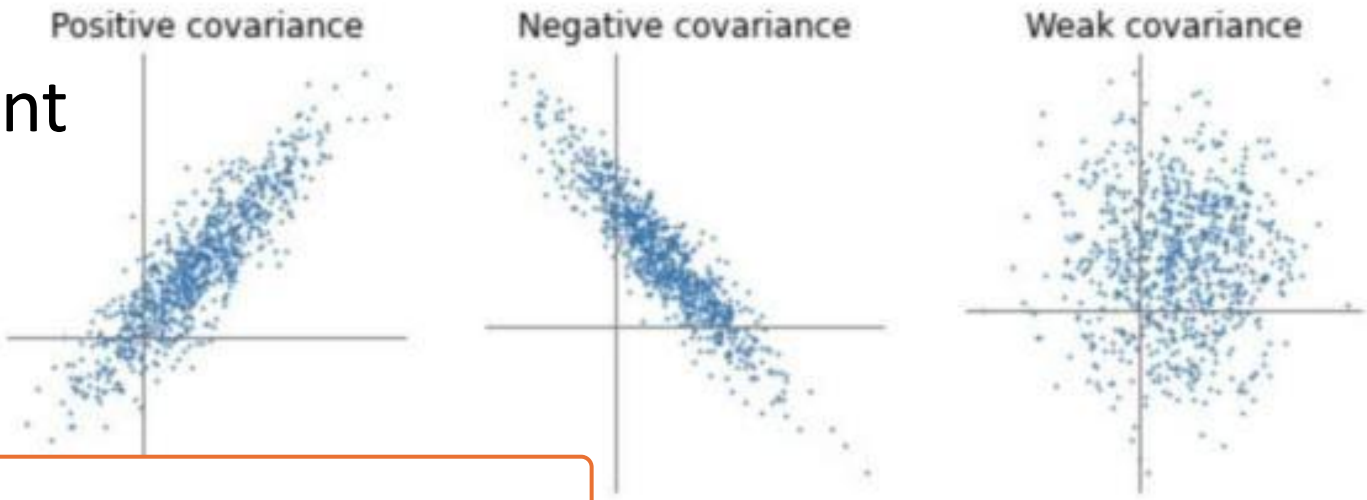
9. Covariance & Correlation

What is Covariance?

- Is a Statistical measure used to describe how much two variables change together.
- For example, suppose you have two random variables X & Y:
  - If Covariance is highly positive, then the relation between them is Positive, which means if X increases, then Y increases also.
  - If Covariance is highly negative, then the relation between them is Negative, which means if X increases, then Y decreases.
  - If Covariance is near to zero, then the relation is weak or there is no relation.

What is Covariance?

- Covariance can also be defined as "How much the deviation of one variable(X) from its mean is (related/or similar) to the deviation of another variable(Y) from its mean".
- The deviation of a random variable from its mean represent the amount of change and the direction of this change also.
- Cov(X, Y) is used to represent covariance between X & Y.



How to Calculate Covariance?

- Formula:
  - $Cov(X, Y) = \sum_{i=1}^{n}((X_i - \mu_x) * (Y_i - \mu_y)) \,/\, n$.
  - n is the number of samples.
  - μ_x is the mean of Random Variable X.
  - μ_y is the mean of Random Variable Y.
- Example:

X = [1, 2, 3, 4, 5, 6, 7, 8, 9] Y = [9, 8, 7, 6, 5, 4, 3, 2, 1]

$\mu_x = 5$ $\mu_y = 5$ n = 9

Cov(X, Y) = ((1-5)*(9-5) + (2-5)*(8-5) + (3-5)*(7-5) + (4-5)*(6-5) + (5-5)*(5-5)
+ (6-5)*(4-5) + (7-5)*(3-5) + (8-5)*(2-5) + (9-5)*(1-5) +)/n
= -6.667

Result: Cov(X, Y) = -6.667 < 0.

Conclusion: The relation between X & Y is Negative.

What is Correlation?

- Is a Statistical measure that is the same as Covariance, except that Correlation is normalized, which give us sense about the relation strength.
- Normalized means that Correlation has values in range = [-1:1].
- For example, suppose you have two random variables X & Y:
  - If Correlation is near to 1, then the relation between them is Strong Positive. While If Correlation is near to 2, then the relation between them is Strong Negative.
  - If Correlation is near to 0, then the relation is weak.

Correlation Vs Covariance:

- Correlation has values in range [-1 : 1]. While Covariance had values between [∞, -∞].
- Having a range between -1 & 1 is very useful since this helps us know how much strong is the relation between the two variables.
- This is useful if I want to compare two relations. While in covariance this is not possible.
- Example:

Correlation	Covariance
➢ Relation1 = .5 ➢ Relation2 = .25 ➢ Relation1 is twice strong as Relation2.	➢ Relation1 = 5 ➢ Relation2 = 2.5 ➢ You can't tell how much Relation1 is stronger than Relation2.

How to Calculate Correlation?

- Formula:
  - $Corr(X, Y) = Cov(X, Y)/(\sigma_x * \sigma_y)$.
  - σ_x is the Standard-deviation of Random Variable X.
  - σ_y is the Standard-deviation of Random Variable Y.
- Example:

X = [1, 2, 3, 4, 5, 6, 7, 8, 9]   Y = [9, 8, 7, 6, 5, 4, 3, 2, 1]

$\sigma_x = 2.582$ $\sigma_y = 2.582$

Cov(X, Y) = -6.667

$Corr(X, Y) = Cov(X, Y) / (\sigma_x * \sigma_y) = -6.667 / (2.582 * 2.582) = -1$

Result: Corr(X, Y) = -1.

Conclusion: The relation between X & Y is Negative.

Covariance & Correlation using Pandas:

Covariance Matrix

- Get the covariance between all the pairs of columns in the DataFrame.

```
random_variable1 = df.Length
random_variable2 = df.Width
df.cov()
```

	Length	Width	Price
Length	6.367273e+01	-7.090909e-01	6.240000e+06
Width	-7.090909e-01	1.633636e+02	2.234000e+07
Price	6.240000e+06	2.234000e+07	1.002800e+13

Correlation Matrix

- Get the correlation between all the pairs of columns in the DataFrame.

```
random_variable1 = df.Length
random_variable2 = df.Width
df.corr()
```

	Length	Width	Price
Length	1.000000	-0.006953	0.246945
Width	-0.006953	1.000000	0.551948
Price	0.246945	0.551948	1.000000

1. Pandas Basics

Import:

```
import pandas as pd
```

Create Series:

With default index	Specify the index

`s = pd.Series([1, 2, 3, 4])`
`s`
 	
`0    1`
`1    2`
`2    3`
`3    4`
`dtype: int64`
	`# Specify the indeces`
	`s = pd.Series([1, 2, 3, 4], index = ["A", "B", "C", "D"])`
	`s`
	
	`A    1`
	`B    2`
	`C    3`
	`D    4`
	`dtype: int64`

Create DataFrame:

```
data = [[1, 444, 'abc'],
        [2, 555, 'def'],
        [3, 666, 'ghi'],
        [4, 444, 'xyz']]
df = pd.DataFrame(data, columns=["col1", "col2", "col3"])
df
```

	col1	col2	col3
0	1	444	abc
1	2	555	def
2	3	666	ghi
3	4	444	xyz

```
# another way
data = {'col1':[1,2,3,4],
        'col2':[444,555,666,444],
        'col3':['abc','def','ghi','xyz']}

df = pd.DataFrame(data)
df
```

	col1	col2	col3
0	1	444	abc
1	2	555	def
2	3	666	ghi
3	4	444	xyz

Rename DataFrame Columns & index:

Rename DataFrame Columns

```
df = pd.DataFrame([[1, 444, 'abc'],
                   [2, 555, 'def'],
                   [3, 666, 'ghi'],
                   [4, 444, 'xyz']])
display(df)
columns=["col1", "col2", "col3"]
df.columns = columns
display(df)
```

	0	1	2
0	1	444	abc
1	2	555	def
2	3	666	ghi
3	4	444	xyz

	col1	col2	col3
0	1	444	abc
1	2	555	def
2	3	666	ghi
3	4	444	xyz

Rename DataFrame index

```
df = pd.DataFrame([[1, 444, 'abc'],
                   [2, 555, 'def'],
                   [3, 666, 'ghi'],
                   [4, 444, 'xyz']])
display(df)
index = ["row1", "row2", "row3", "row4"]
df.index = index
display(df)
```

	0	1	2
0	1	444	abc
1	2	555	def
2	3	666	ghi
3	4	444	xyz

	0	1	2
row1	1	444	abc
row2	2	555	def
row3	3	666	ghi
row4	4	444	xyz

Pandas Dtypes:

Bool ➢ Represents Numerical datatypes with True & False values.	**Int** ➢ Represents Numerical datatypes with integer values.	**Float** ➢ Represents Numerical datatypes with continuous values.
Category ➢ Represents Categorical datatypes.	**Object** ➢ Is a mix of categorical datatypes & Numerical datatypes. ➢ Can carry any python object; such as, lists, tuples, strings, etc.	

Pandas Dtypes:

Get Datatypes of all columns	Get Datatype of one column

`# Datatype of all columns`
`df.dtypes`
 	
`col1      int64`
`col2      int64`
`col3     object`
`dtype: object`
	`# Datatype of one column`
	`df['col1'].dtype`
	
	`dtype('int64')`

Change Datatype:

Change Datatype of one column	Change Datatypes of group of columns

Change Datatype of one column

```
df["col1"] = df["col1"].astype("category")
df.dtypes
```

```
col1    category
col2       int64
col3      object
dtype: object
```

Change Datatypes of group of columns

```
cols = ["col1", "col3"]
df[cols] = df[cols].astype("category")
df.dtypes
```

```
col1    category
col2       int64
col3    category
dtype: object
```

2. EDA using Pandas

EDA using Pandas:

- EDA is about exploring and understanding the data and getting insights about it.
- EDA is short for Exploratory Data Analysis.
- Pandas provides built-in methods and features that helps us to answer different questions about the data.

EDA using Pandas:

Get the first n rows of DataFrame

```
1 df.head(2)
```

	col1	col2	col3
0	1	444	abc
1	2	555	def

Get the last n rows of DataFrame

```
1 df.tail(2)
```

	col1	col2	col3
2	3	666	ghi
3	4	444	xyz

Get Statistical Measures about Numerical Columns

```
1 df.describe()
```

	count	mean	std	min	25%	50%	75%	max
col2	4.0	527.25	106.274409	444.0	444.0	499.5	582.75	666.0

Get Statistical Measures about Categorical Columns

```
1 df.describe(include="category")
```

	col1
count	4
unique	4
top	1
freq	1

EDA using Pandas:

Get DataFrame Rows Names

```
df.index
```

```
RangeIndex(start=0, stop=4, step=1)
```

Get DataFrame Columns Names

```
df.columns
```

```
Index(['col1', 'col2', 'col3'], dtype='object')
```

Get Unique Values

```
# get the unique values
df['col2'].unique()
```

```
array([444, 555, 666], dtype=int64)
```

Get Unique Values Number

```
# get number of unique values
df['col2'].nunique()
```

```
3
```

EDA using Pandas:

Get Max Value	Get Element whose value is Max
`df["col2"].max()` 666	`df.col2.idxmax()` 2
Get Min Value	**Get Element whose value is Min**
`df["col2"].min()` 444	`df.col2.idxmin()` 0

EDA using Pandas:

DataFrame Basic Information

```
1  df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   col1    4 non-null      category
 1   col2    4 non-null      int64
 2   col3    4 non-null      object
dtypes: category(1), int64(1), object(1)
memory usage: 400.0+ bytes
```

Unique Values Frequency

```
1  d = df['col2'].value_counts()
2  d
```

```
444    2
555    1
666    1
Name: col2, dtype: int64
```

Get Max Value of All Columns

```
1  df.max()
```

```
col2      666
col3      xyz
dtype: object
```

Summation

```
1  df['col2'].sum()
```

```
2109
```

3. Data Manipulation

Data Manipulation:

- Pandas provides built-in methods and attributes that allows you to apply different operations over Pandas DataFrames or Series.
- Below are the most popular & Important attributes that allows you to apply data manipulation.

Data Manipulation (Pandas to Numpy):

Convert Pandas Series into Numpy 1D-Array	Convert Pandas DataFrame into Numpy 2D-Array
`df['col1'].values`	`df.values`

`[1, 2, 3, 4]`
`Categories (4, int64): [1, 2, 3, 4]`
	`array([[1, 444, 'abc'],`
	`       [2, 555, 'def'],`
	`       [3, 666, 'ghi'],`
	`       [4, 444, 'xyz']], dtype=object)`

Data Manipulation (Replace Values):

Replace a Single Value

```
df.replace(555, "ali")
```

	col1	col2	col3
0	1	444	abc
1	2	ali	def
2	3	666	ghi
3	4	444	xyz

Replace Multiple Values using Dictionary

```
df.replace({444: "omar", "abc": 444})
```

	col1	col2	col3
0	1	omar	444
1	2	555	def
2	3	666	ghi
3	4	omar	xyz

Replace Multiple Values by One Value

```
df.replace([1, 666, "abc"], "aaa")
```

	col1	col2	col3
0	aaa	444	aaa
1	2	555	def
2	3	aaa	ghi
3	4	444	xyz

Data Manipulation (Mapping):

Before Mapping

```
df
```

	col1	col2	col3
0	1	444	abc
3	4	444	xyz
1	2	555	def
2	3	666	ghi

After Mapping

```
df.col2 = df.col2.map({444: "Fours", 555: "Fives", 666: "Sixs"})
df
```

	col1	col2	col3
0	1	Fours	abc
3	4	Fours	xyz
1	2	Fives	def
2	3	Sixs	ghi

Data Manipulation (Sorting):

Ascending sorting

```
sorted_df = df.sort_values(by='col1')
sorted_df
```

	col1	col2	col3
0	1	Fours	abc
1	2	Fives	def
2	3	Sixs	ghi
3	4	Fours	xyz

Descending sorting

```
sorted_df = df.sort_values(by='col1', ascending=False)
sorted_df
```

	col1	col2	col3
3	4	Fours	xyz
2	3	Sixs	ghi
1	2	Fives	def
0	1	Fours	abc

Data Manipulation (Apply method):

- Is a methods used to apply a certain function to each sample in the DataFrame.

Example1

```
def duplicate(x):
    return x*2

df['col1'].apply(duplicate)
```

```
0     11
1     22
2     33
3     44
Name: col1, dtype: object
```

Example2

```
# apply built in function
df['col1'].apply(len)
```

```
0     1
1     1
2     1
3     1
Name: col1, dtype: int64
```

Example3

```
# or use lambda function
df['col1'].apply(lambda x: x*2)
```

```
0     11
1     22
2     33
3     44
Name: col1, dtype: object
```

Example4

```
df2 = pd.DataFrame([[1, 'ALI'   , 3],
                    [5, 'OMAR'  , 8],
                    [4, 'AHMED', 9]])
df2.apply(lambda x: x*2)
```

	0	1	2
0	2	ALIALI	6
1	10	OMAROMAR	16
2	8	AHMEDAHMED	18

4. Indexing & Slicing

Indexing:

- Means accessing one element in Series or DataFrame, using its index or name.

- There are two ways to apply indexing:
  - Either using the element’s Index.
  - Or using the element’s Name.

Slicing:

- Means accessing many elements in Series or DataFrame, by specifying a range of Indices or name.

- There are two ways to apply Slicing:
  - Either, using a range of elements’ Indices.
  - Or using a range of elements’ Names.

Indexing & Slicing using Index:

- **Series Indexing**

```
df.col1.iloc[2]
```

```
3
```

- **Matrix Indexing**

```
df.iloc[2, 1]
```

```
666
```

- **Series Slicing**

```
df.col1.iloc[0:2]
```

```
0    1
1    2
Name: col1, dtype: category
Categories (4, int64): [1, 2, 3, 4]
```

- **Matrix Slicing**

```
df.iloc[:, 0:2]
```

	col1	col2
0	1	444
1	2	555
2	3	666
3	4	444

Indexing & Slicing using Names:

- **Series Indexing**

```
df.col1.loc[3]
```

```
4
```

- **Matrix Indexing**

```
df.loc[2, "col1"]
```

```
3
```

- **Series Slicing**

```
df.col1.loc[2:3]
```

```
2    3
3    4
Name: col1, dtype: category
Categories (4, int64): [1, 2, 3, 4]
```

- **Matrix Slicing**

```
df.loc[:, "col1":"col2"]
```

	col1	col2
0	1	444
1	2	555
2	3	666
3	4	444

5. Inserting/Dropping DataFrame Columns & Rows

Insert New Columns:

The first way

```
new_col = df.col1 + df.col2
df.insert(3,"new" , new_col)
df
```

	col1	col2	col3	new
0	1	444	abc	445
1	2	555	def	557
2	3	666	ghi	669
3	4	444	xyz	448

Another way

```
df['new'] = df.col1 + df.col2
df
```

	col1	col2	col3	new
0	1	444	abc	445
1	2	555	def	557
2	3	666	ghi	669
3	4	444	xyz	448

Drop Columns:

Drop one columns

```
1  df.drop('new',axis=1)
```

	col1	col2	col3
0	1	444	abc
1	2	555	def
2	3	666	ghi
3	4	444	xyz

Drop many columns

```
1  df.drop(['col1', 'new'],axis=1)
```

	col2	col3
0	444	abc
1	555	def
2	666	ghi
3	444	xyz

Insert New Rows:

```
new_row = {"col1": -1, "col2": 222, "col3": "OuO"}
df.append(new_row, ignore_index=True)
```

	col1	col2	col3
0	1	444	abc
1	2	555	def
2	3	666	ghi
3	4	444	xyz
4	-1	222	OuO

Drop Rows:

Drop one row

```
df.drop(2, axis=0)
```

	col1	col2	col3
0	1	444	abc
1	2	555	def
3	4	444	xyz

Drop many rows

```
df.drop([1, 3], axis=0)
```

	col1	col2	col3
0	1	444	abc
2	3	666	ghi

6. Null Values

What are Null Values?

- Null Values means missing values, which means that an element doesn’t have a value, or have a value of None or Nan.
- Null Values occur due to problems during gathering data, for example a client forgot or to enter his age.

Check for Null Values:

```
null = df.isnull()
pd.DataFrame(null.sum()).T
```

	col1	col2	col3
0	0	1	2

Handle Null Values:

- There are three options to do to handle missing values:

1. Drop rows that contain Null Values.
2. Drop columns that contain Null Values.
3. Replace Null Values with Mean, Median, or Mode.

Handle Null Values (Drop Rows):

Drop all rows

```
df.dropna()
```

	col1	col2	col3
2	1	2.0	3.0

Drop rows in specific columns

```
df.dropna(subset=["col2"])
```

	col1	col2	col3
0	1	2.0	3.0
2	1	2.0	3.0

Handle Null Values (Drop Columns):

Drop all columns

```
df.dropna(axis=1)
```

	col1	col3
0	1	3.0
1	5	3.0
2	1	3.0

Drop Specific Columns

```
df.drop(["col3"], axis=1)
```

	col1	col2
0	1	2.0
1	5	NaN
2	1	2.0

Handle Null Values (Replace Rows Null Values):

```
mean = df.col3.mean()
df.col3 = df.col3.fillna(value=mean)
df
```

	col1	col2	col3
0	1	2.0	3.0
1	5	NaN	3.0
2	1	2.0	3.0

Assignment 1: The automated_stat_analyzer Function

A retail company needs a utility to quickly summarize sales data. Students must create a function that identifies the "Central Tendency" and "Dispersion" of any numerical column.

Requirements:

•Accept a Pandas DataFrame and a column name.
•Calculate the **Mean** ,**Median** dna ,**Standard Deviation**.
•Identify if the data is "Skewed" by comparing the Mean and Median.
•**Bonus** eht nruter ,lacirogetac si nmuloc eht fI :**Mode** instead.

```python
import pandas as pd
import numpy as np

data = {
    'Transaction_ID': range(1, 11),
    'Product_Category': ['Electronics', 'Home', 'Electronics', 'Sports', 'Home',
                         'Electronics', 'Home', 'Sports', 'Electronics', 'Electronics'],
    'Sales_Amount': [150, 200, 155, 300, 210, 180, 205, 1000, 190, 160], # 1000 is an Outlier
    'Customer_Age': [25, 34, np.nan, 45, 23, 31, 29, np.nan, 38, 40],    # Contains Nulls (NaN)
    'Rating': [5, 4, 3, 5, 2, 4, 5, 2, 4, 3]
}
```

```python
import pandas as pd

def automated_stat_analyzer(df, column_name):
    """
    Company Task: Provide a summary report of a specific data variable.

    Instructions:
    1. Check if the column is numerical or categorical.
    2. For numerical: Calculate Mean, Median, and Standard Deviation.
    3. For categorical: Calculate the Mode.
    4. Return a dictionary with these statistical measures.
    """
    # TODO: Implement using df[column_name].mean(), .median(), .std(), or .mode()
    pass
```

Assignment 1 The null_handling_strategy Function

Incoming user data often has missing values. Students must implement a flexible strategy to handle these "Null Values" to prepare data for Machine Learning

Requirements:

- Check for null values in the DataFrame.
- Apply a strategy based on parameters: "naidem_llif" ro ,"naem_llif" ,"swor_pord"
- Ensure the function only fills numerical columns when using mean or median.

```python
import pandas as pd
import numpy as np

data = {
    'Transaction_ID': range(1, 11),
    'Product_Category': ['Electronics', 'Home', 'Electronics', 'Sports', 'Home',
                         'Electronics', 'Home', 'Sports', 'Electronics', 'Electronics'],
    'Sales_Amount': [150, 200, 155, 300, 210, 180, 205, 1000, 190, 160], # 1000 is an Outlier
    'Customer_Age': [25, 34, np.nan, 45, 23, 31, 29, np.nan, 38, 40],    # Contains Nulls (NaN)
    'Rating': [5, 4, 3, 5, 2, 4, 5, 2, 4, 3]
}
def null_handling_strategy(df, strategy="fill_mean"):
    """
    Company Task: Clean a dataset by resolving missing (NaN) values.
    """
    # TODO: Implement using .isnull(), .dropna(), or .fillna()
    pass
```

Thank You