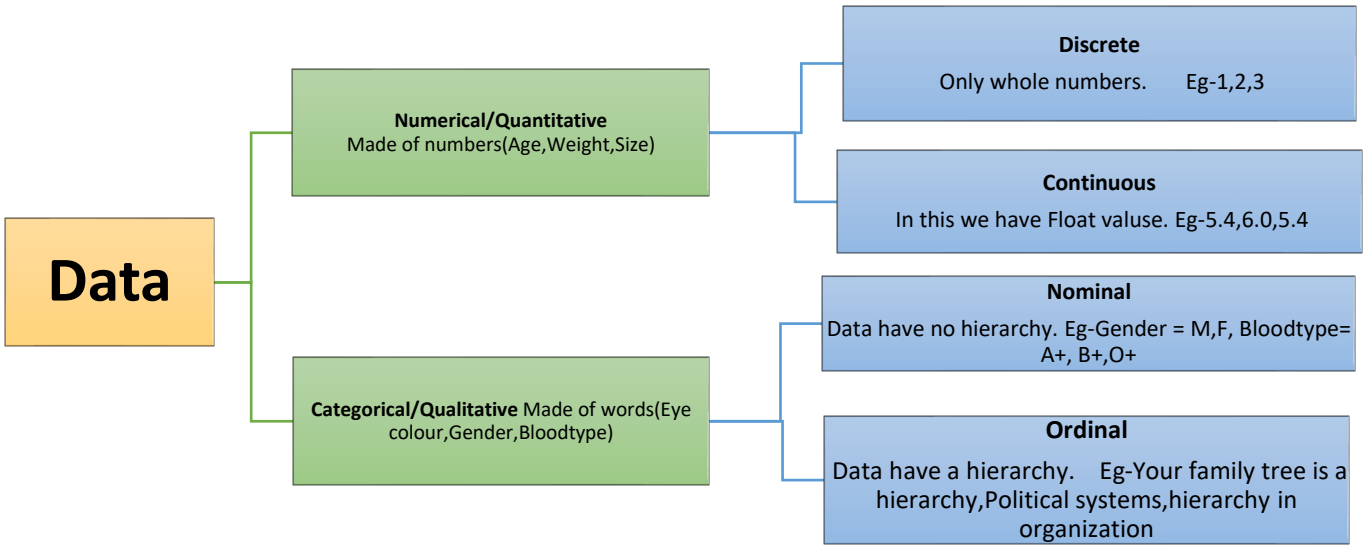
Exploratory Data Analysis(EDA)

Exploratory Data Analysis (EDA) is an approach to analyze the data using visual techniques.It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations.

It's a combination of [Pandas Function + Visualisation]

★Types of Data:-



Relationship between Variables:-

★1-) Ratio & Interval Variable:-

Ratio Variable: - Never fall below Zero. E.g.-Height, Weight

Interval Variable: - Indicates difference between two entities, No zero Value. E.g. -Temperature, Income range.

2-) Dependent & Independent Variable:-

Independent Variable: - The independent variable is the cause. Its value is independent of other variables in your study.

Dependent Variable: - The dependent variable is the effect. Its value depends on changes in the independent variable.

★ Types of Analysis:-

Types of Analysis

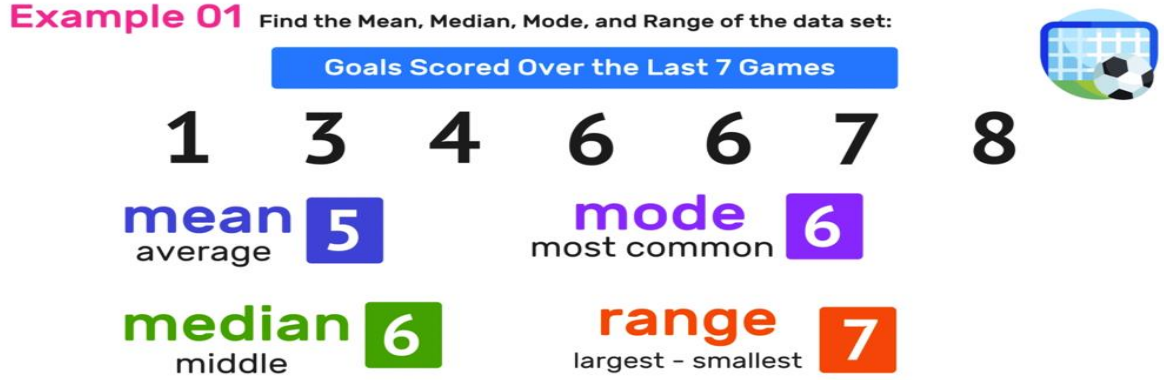
- **Univariate Analysis**:- Focusing & Analyzing one variable at a time.
- **Bivariate Analysis**:- Comparing two variable at a time.(numarical-numarical,categorical-numarical)
- **Multivariate Analysis**:- Comparing more than two variable at a time which may be numarical or catogorical

★ Statistical Analysis:-

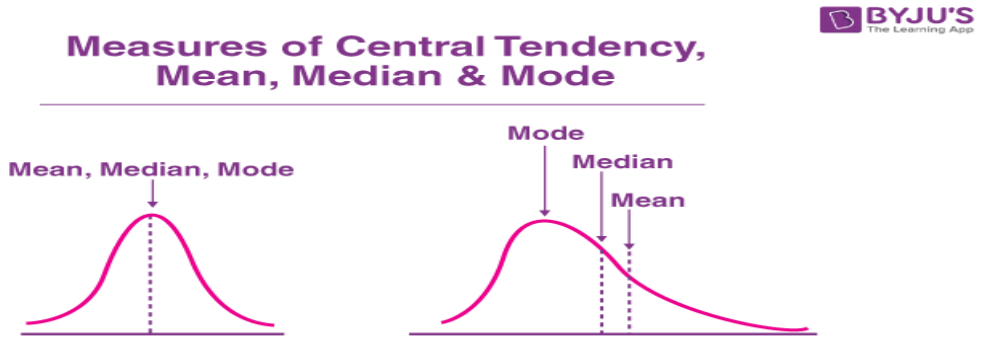
1-) Measures of central Tendency

- **Mean**:- Sum of all values divided by the total number of values.
- **Median**:- A median is the centre value of a given list of observations when arranged in an order.
- **Mode**:- The value that occurs most often.

Example of mean,median mode:-



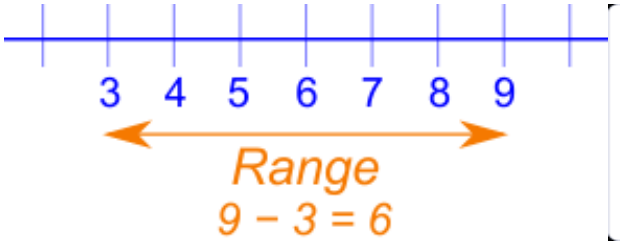
Graphical representation of mean, median mode:-

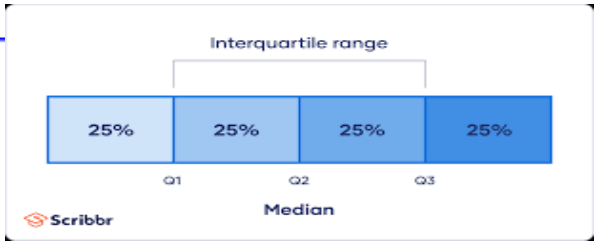


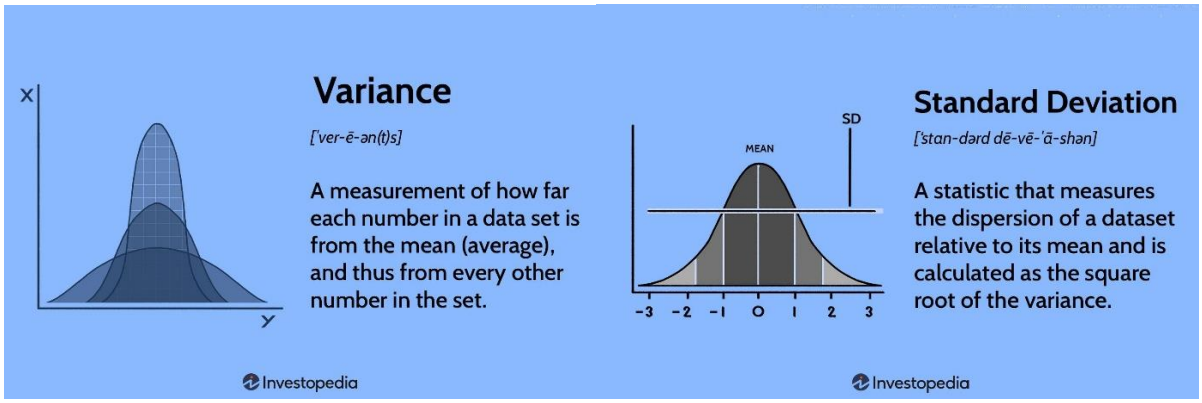
Note:- Mean is effected by outliars. while median & mode is not effected by outliers.

2-) Measures of Dispersion(फैलाव)

- **Range**:- Difference between the highest & lowest value in the data.
  ex:-(if the given data set is {2,5,8,10,3}, then the range will be 10 – 2 = 8)
- **Inter Quertile Range**:- It tells you the spread of the middle half of your distribution. that is lie between(Q1, Q2 ,Q3)
- **Variance**:- Dispersion around mean. we denote variance (σ^2).
- **Standard Deviation**:- Square Root of variance. we denote variance (σ).

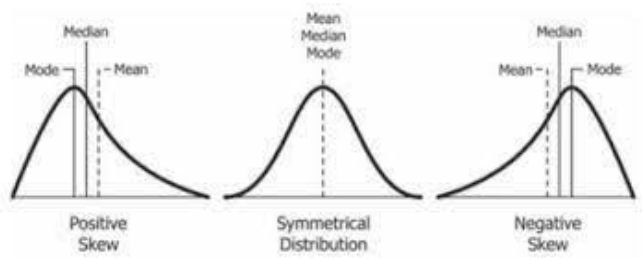


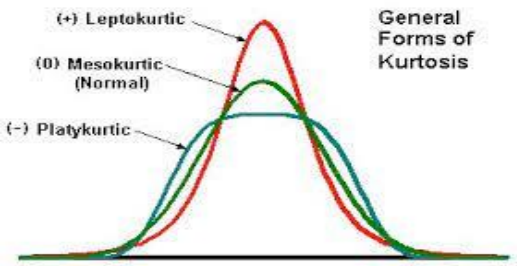




3-) Measures of Shape

- Skewness:- Symmetry in Distribution.
- Kurtosis:- Shape of peaks in Distribution of data.





Some codes of EDA:-

Data = pd.read_csv('Travel.csv')

1) Data.columns # shows the column headings.

2) Data.info () #prints information about the Data Frame.

3) Data.shape # rows and column counts give

4) Data.dtypes # gives each column data type containing

5) Data.describe() # gives the statistical information for the numerical containing data types

6) Data.describe().T # for better fit of the data in screen we can transpose the same

7) Numeric_data.corr() # check the co-relation of the numerical data

8) Numeric_data.cov() # check the covariance of the numerical data

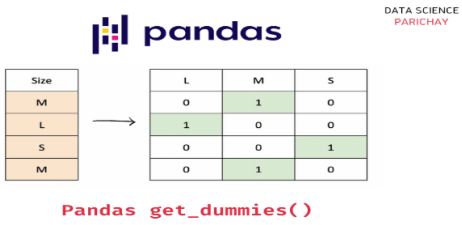
Note: - Difference between Info & Describe

The. Info () method allows us to learn the shape of object types of our data. The. Describe () method gives us statistics summary(Percentile, Mean and Std**) for numerical columns in our Data Frame**.

★Encoding Categorical Variables:-

Encoding categorical data is a process of converting categorical data into integer format so that the data with converted categorical values can be provided to the different models.

There are 3 Methods to convert categorical data into numerical data.



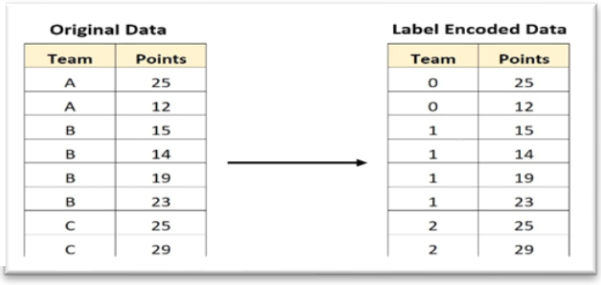
Method 1: One Hot Encoding /Get-Dummies method :- The get dummies function is used to convert categorical variables into dummy or indicator variables. A dummy or indicator variable can have a value of 0 or 1.

Dummies= pd.get_dummies(x,drop_first=True)

x= Dataframe

drop first = True implies that the dropping the first variable, Because it is useless.

Method 2: Label Encoding :- we replace the categorical value with a numeric value between 0 and the number of classes minus 1.



Original Data

Team	Points
A	25
A	12
B	15
B	14
B	19
B	23
C	25
C	29

Label Encoded Data

Team	Points
0	25
0	12
1	15
1	14
1	19
1	23
2	25
2	29

	id	Gender	Age	Department	Rating
0	101	M	21	QA	A
1	102	M	25	QA	B
2	103	M	24	Dev	B
3	104	F	28	Dev	C
4	105	F	25	UI	B

```
# Converting Ordinal Variable Rating to numeric
EmployeeData['Rating'].replace({'A':3, 'B':2, 'C':1}, inplace=True)
EmployeeData
```

	id	Gender	Age	Department	Rating
0	101	M	21	QA	3
1	102	M	25	QA	2
2	103	M	24	Dev	2
3	104	F	28	Dev	1
4	105	F	25	UI	2

Method 3: replace () method:- it's a string method where we replace the categorical value with a numeric value.

Missing value treatment:-

Q. What Is a Missing Value? How Is a Missing Value Represented in a Dataset?

Ans-The values or data that is not stored (or not present) for some variables in the given dataset.

In the dataset, the blank shows the missing values.

In Pandas, usually, missing values are represented by **NaN**. It stands for **Not a Number.**

Q. Why Do We Need to Care About Handling Missing Data?

Ans. It is important to handle the missing values appropriately.

- Many machine learning algorithms fail if the dataset contains missing values. However, algorithms like K-nearest and Naïve Bayes support data with missing values.
- You may end up building a biased machine learning model, leading to incorrect results if the missing values are not handled properly.
- Missing data can lead to a lack of precision in the statistical analysis.

How to find total number of missing values from the entire dataset

- train_df.isnull().sum()

#There are 2 primary ways of handling missing values:

1.Deleting the Missing value

Generally, this approach is not recommended. It is one of the quick and dirty techniques one can use to deal with missing values.

There are 2 ways one can delete the missing data values:

- Deleting the entire row (listwise deletion)

If a row has many missing values, you can drop the entire row. If every row has some (column) value missing, you might end up deleting the whole data. The code to drop the entire row is as follows:

- df = train_df.dropna(axis=0)
- Note: here we give axis=0 for rows

- Deleting the entire column

If a certain column has many missing values, then you can choose to drop the entire column. The code to drop the entire column is as follows:

- df = train_df.drop(['Dependents'],axis=1)
- Note: Here Dependents is a column name, axis=1 id for column

2.Imputing the Missing Value

There are many imputation methods for replacing the missing values. You can use different python libraries such as Pandas, and Sci-kit Learn to do this. Let's go through some of the ways of replacing the missing values.

Replace the missing value using 'fillna' method :-
- train_df['Dependents'] = train_df['Dependents'].fillna(0)
  Note: "fillna"- defaulty fill with 0

Replacing with the mean :-
- train_df['LoanAmount'] = train_df['LoanAmount'].fillna(train_df['LoanAmount'].mean())

Replacing with the mode :-
- train_df['Married'] = train_df['Married'].fillna(train_df['Married'].mode()[0])

Replacing with the median :-
- train_df['Loan_Amount_Term'].fillna(train_df['Loan_Amount_Term'].median())

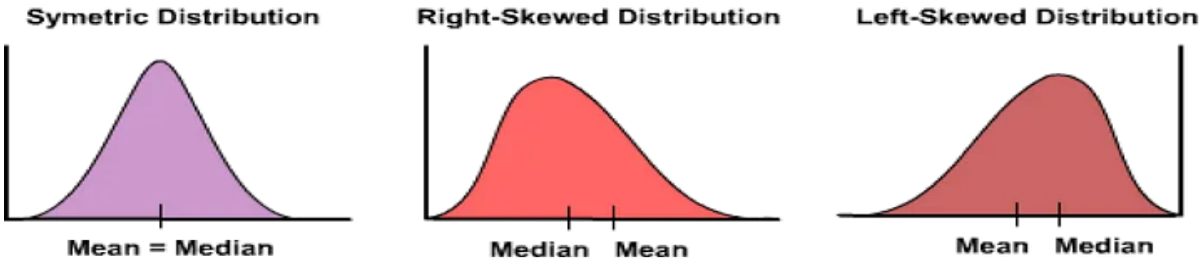
Replacing with the previous value – forward fill :-
- test.fillna(method='ffill')

Replacing with the next value – backward fill :-
- test.fillna(method='bfill')

★ Handling Skewed Data :-

Data is skewed when its distribution curve is asymmetrical (as compared to a normal distribution curve that is perfectly symmetrical) and skewness is the measure of the asymmetry. The skewness for a normal distribution is 0.

There are 2 different types of skews in data, left(negative) or right(positive) skew**.**



Effects of skewed data: Degrades the model's ability (especially regression based models) to describe typical cases as it has to deal with rare cases on extreme values. Skewed data also does not work well with many statistical methods. However, **tree based models are not affected.**

To ensure that the machine learning model capabilities is not affected, skewed data has to be transformed to approximate to a normal distribution. The method used to transform the skewed data depends on the characteristics of the data.

To check for skew in data:

- df.skew().sort_values(ascending=False)

Dealing with skew data:

1.log transformation: transform skewed distribution to a **normal distribution**

Not able to log 0 or negative values (add a constant to all value to ensure values > 1)

Log transform a single column

- df['col1'] = np.log(df['col1']

Log transform multiple columns in dataframe

- df = df[['col1', 'col2']].apply(lambda x: np.log(x))

2. Remove outliers

3. Normalize (min-max)

4. Cube root: when values are too large. Can be applied on negative values

5. Square root: applied only to positive values

6. Reciprocal

7. Square: apply on left skew

8. Box Cox transformation: transform non-normal to approximate a normal distribution using eq 1 below, with λ between [-5,5]. The optimum λ is selected which gives the best approximation to a normal distribution.

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

Eq1

Eq 1 Only works for positive values of y. Use eq 2 when there are negative values of y.

$$y(\lambda) = \begin{cases} \frac{(y + \lambda_2)^{\lambda_1} - 1}{\lambda_1}, if\ \lambda_1 \neq 0 \\ \log(y + \lambda_2), if\ \lambda = 0 \end{cases}$$

Eq2

#Difference between Box Cox transformation and log transformation :-

Mathematical form: The Box Cox transformation involves applying a mathematical function that varies depending on a parameter called lambda (λ), while the log transformation involves taking the natural logarithm of the data.

Parameter estimation: The Box Cox transformation requires estimating the optimal value of λ, which can be done using maximum likelihood estimation or a similar method. In contrast, the log transformation does not require any parameter estimation.

Handling of zero values: The Box Cox transformation requires the data to be positive, so it cannot handle zero or negative values. The log transformation can handle zero values by adding a small constant to the data before taking the logarithm.

Bias: The Box Cox transformation can introduce bias, especially if the optimal value of λ is estimated from a small sample size. The log transformation is generally less biased.

Impact on outliers: The Box Cox transformation can be more robust to outliers than the log transformation because it allows for a wider range of transformations. The log transformation can be sensitive to extreme values and may amplify the impact of outliers on the transformed data.

Overall, the choice between the Box Cox transformation and log transformation depends on the characteristics of the data and the goals of the analysis. Both techniques can be effective in transforming non-normal data, but the Box Cox transformation may be more flexible and robust, while the log transformation may be simpler and more straightforward to apply.