***UNIT-2***

***Statistical Learning and Data preprocessing.***

***Syllabus:***

**Statistical Learning**: Introduction to Data representation, types of data-nominal, ordinal, interval and continuous, central tendency-calculating mean mode median, mean vs median, variability, variance, standard deviation, Mean Absolute Deviation using sample dataset, finding the percentile, inter quartile range, Box Plot, Outlier, whisker, calculating correlation, covariance, causation.

**Data Pre-processing-** Exploratory data analysis, Data preparation and preprocessing, visualization, and its tools.

**Statistical Learning:**

***Statistics is a group of methods used to collect, analyze, present, and interpret data and to make decisions.***

**Types of Statistics**

1) Descriptive statistics and inferential statistics.

**Descriptive Statistics:**

**Descriptive statistics is a branch of statistics that deals with summarizing and describing the main features of a dataset. It provides methods for organizing, visualizing, and presenting data meaningfully and informally.**

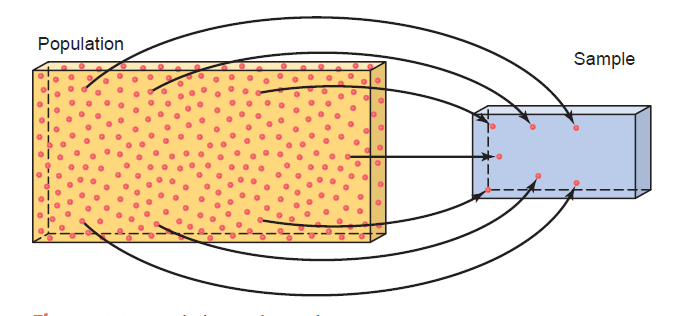
**Common measures and techniques in**[**descriptive statistics**](https://www.simplilearn.com/what-is-descriptive-statistics-article)**include measures of central tendency (such as mean, median, and mode), measures of dispersion (such as range, variance, and standard deviation), frequency distributions (histograms, frequency tables), and graphical representations (box plots, bar charts, pie charts, etc.). These methods help to provide a clear and concise summary of the data, facilitating easier interpretation and understanding.**

**Inferential Statistics**

In statistics, the collection of all elements of interest is called a **population**. The selection of a few elements from this population is called a **sample**.

A major portion of statistics deals with making decisions, inferences, predictions, and forecasts

about populations based on results obtained from samples.



For example, we may make some decisions about the political views of all college and university students based on the political views of 1000 students selected from a few colleges and universities.

The area of statistics that deals with such decision-making procedures is referred to as inferential statistics.

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

**Introduction to Data representation:**

Data refers to raw, unprocessed facts, figures, and statistics collected from various sources, which can be analyzed to gain insights and make decisions.

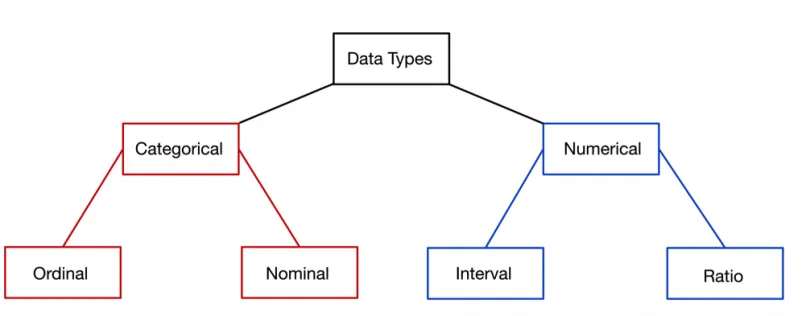
Data Representation in Machine Learning:

Data representation refers to the way in which data is structured and formatted to be used by machine learning algorithms.

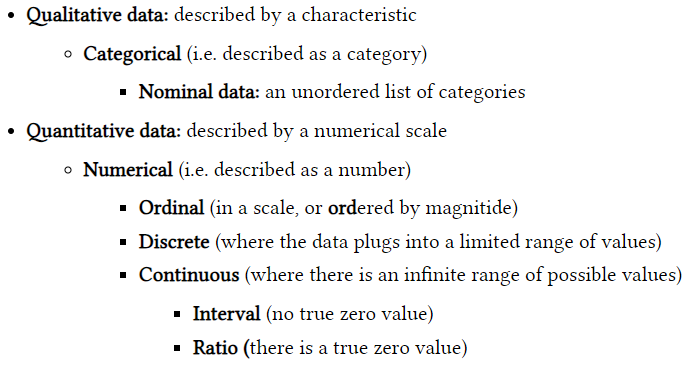
Since machine learning models learn patterns and make decisions based on data, it's essential to represent the data in a way that enables efficient learning and accurate predictions. • Proper representation of data can greatly influence model performance.

Types of Data:

The major categories of data types are as follows.



In statistics, there are four data measurement scales: nominal, ordinal, interval, and ratio. These are simple ways to sub-categorize different types of data.



# Nominal Data (Categorical Data)

* **Definition**: Nominal data represent categories with no intrinsic ordering. They are labels or names used to identify groups or types.
* **Example**: Colors (red, blue, green), gender (male, female), or types of animals (cat, dog, bird).
* **Machine Learning Usage**: Nominal data are often represented as integers or one-hot encoded (a vector where one element is 1 and all others are 0) before being input into a machine learning model. No mathematical operations can be applied to these data.

# Encoding Example:

* + - Gender: Male (0), Female (1)
    - Color: Red (1,0,0), Blue (0,1,0), Green (0,0,1)

# Ordinal Data

* **Definition**: Ordinal data have categories with a clear ordering or ranking, but the distances between these categories are not equal or meaningful.
* **Example**: Rating scales (poor, average, good, excellent), educational levels (high school, undergraduate, graduate).

• Color: Red (1,0,0), Blue (0,1,0), Green (0,0,1)

* **Machine Learning Usage**: Ordinal data can be encoded using integer values that represent the order, but like nominal data, the difference between the values does not carry specific meaning. Some algorithms may take the ordering into account (e.g., tree-based models like decision trees), but most models treat ordinal variables as categorical.

# Encoding Example:

* + - Rating: Poor (1), Average (2), Good (3), Excellent (4)

# Discrete Data

* **Definition**: Discrete data consists of distinct, countable values. Discrete data is often numerical, and it represents quantities that can only take specific values.
* **Example**: Number of students in a class, Number of cars in a parking lot, Number of goals scored in a game.

Example: Representing discrete datanumber\_of\_students = [30, 25, 28, 32, 35]

# Continuous Data

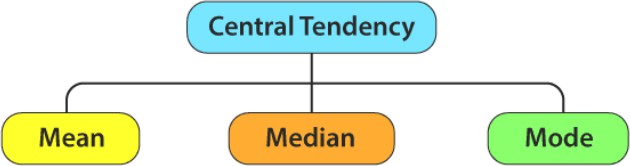
* **Definition**: Continuous data can take any value within a given range. It is not restricted to specific, countable values and can include decimals and fractions.
* Their possible values cannot be counted and can only be described using intervals on the real number line.
* **Example**: Height (e.g., 5.6 feet), Weight (e.g., 65.4 kg), Time (e.g., 2.35 seconds), Temperature (e.g., 37.5°C).

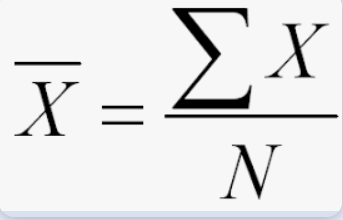
# **Central Tendency**

* In statistics, the central tendency is the descriptive summary of a data set.
* **Definition:**

The central tendency is stated as the statistical measure that represents the single value of the entire distribution or a dataset.

* The three most commonly used measures of central tendency are:

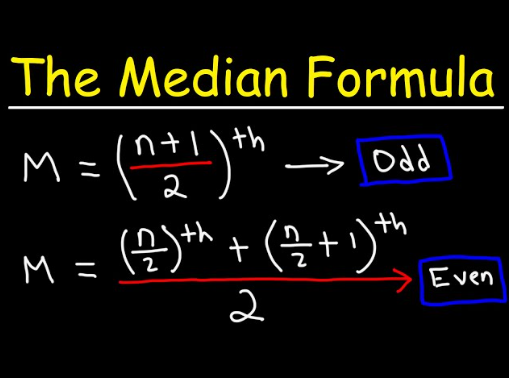


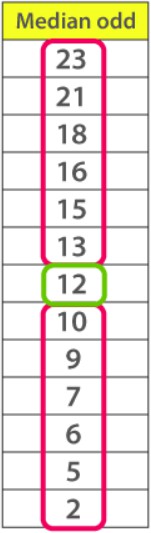
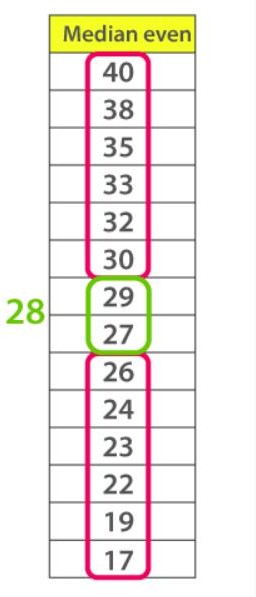
1. **Mean**
   * **Definition**: The mean is the **average** of all the values in a dataset. It is calculated by summing up all the values and then dividing by the total number of values.
   * **Formula**: 
   * Where:
     + X is each value in the dataset
     + N is the total number of values

The mean has one main disadvantage: it is particularly susceptible to the influence of outliers. These are values that are unusual compared to the rest of the data set by being especially small or large in numerical value. For example, consider the wages of staff at a factory below:



1. **Median**
   * **Definition**: The median is the **middle value** of a dataset when the values are arranged in ascending or descending order. If there is an even number of values, the median is the average of the two middle values.



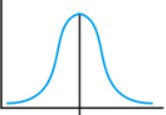
 

1. **Mode**
   * **Definition**: The mode is the **value(s) that appears most frequently** in the dataset. A dataset can have more than one mode if multiple values have the same frequency.



**Symmetrical Distribution**

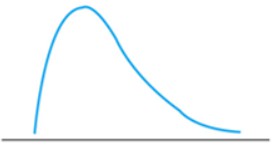
A symmetrical distribution is when the pattern or trend of frequencies on the left and right side of the distribution are the same.



**Skewed Distribution**

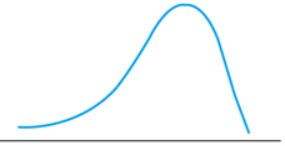
A skewed distribution is when the scores pile or stack up on one side and are spread out on the other (i.e., a distribution that is not symmetrical). There are two types of skewed distributions:

* + Positive Skew: This is when the scores pile up on the lower end of the values with fewer scores at the high end. The side with fewer scores is called the tail and is considered the direction of the skew

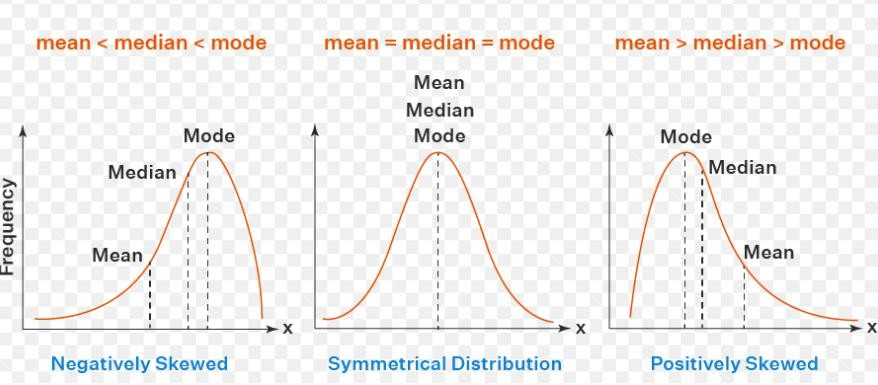


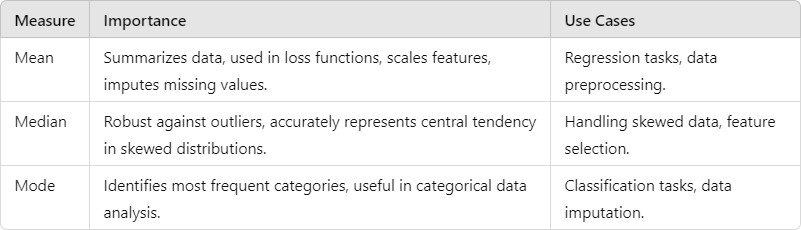
Notice how the side with fewer scores is more spread out and looks like a tail. Since the tail is towards the higher end of the values, it is called a positive skew (i.e., it is skewed right because the tail is pointing to the right).

* + Negative Skew: This is when the scores pile up on the higher end of the values with fewer scores at the low end. See Figure 5 for an example.

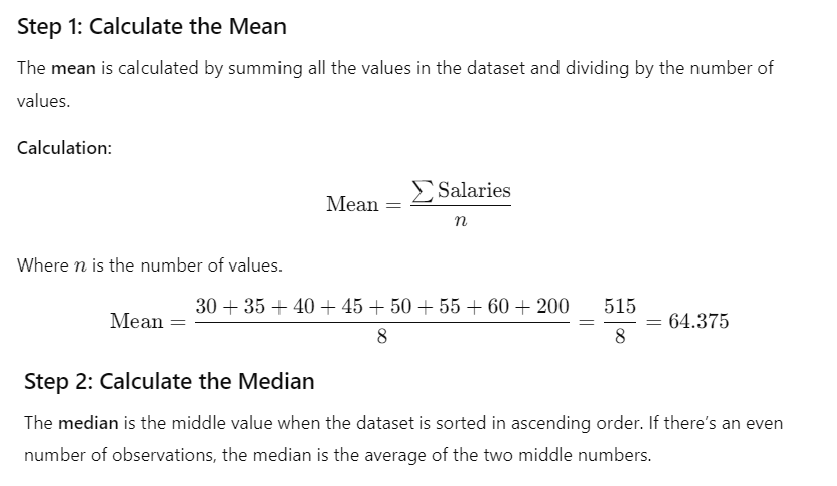


Since the tail is towards the lower end of the values, it is called a negative skew (i.e., it is skewed left because the tail is pointing to the left).

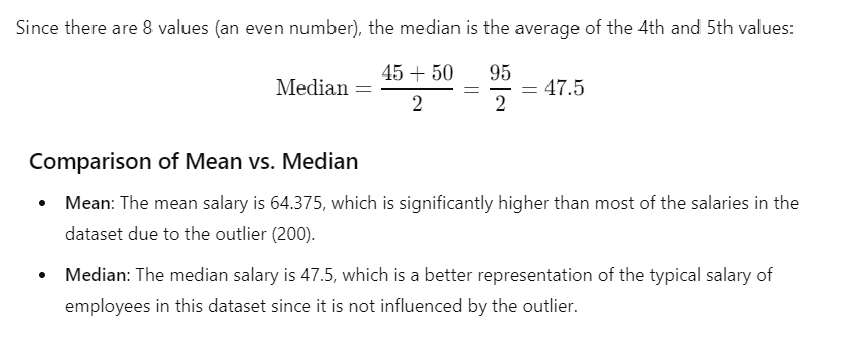




Consider the following dataset representing the salaries (in thousands) of a group of employees: Salaries = [30, 35, 40, 45, 50, 55, 60, 200]



Sorted Salaries = [30, 35, 40, 45, 50, 55, 60, 200]



**Conclusion:**

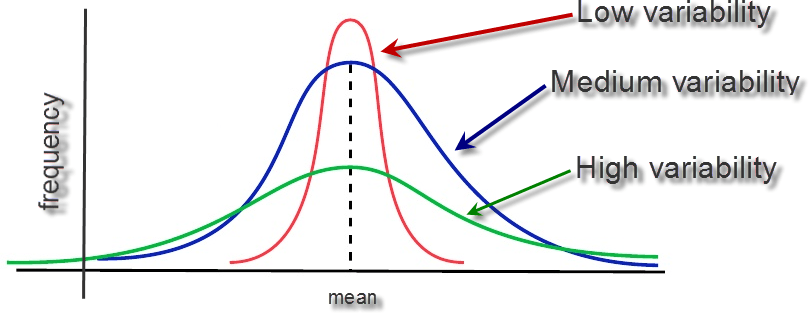
Mean, median, and mode are fundamental tools in machine learning that aid in understanding data distributions, guiding preprocessing decisions, influencing model performance, and ensuring robust analysis. By selecting the appropriate measure based on the data characteristics, practitioners can

improve the accuracy and reliability of their machine learning models.

# Variability, Variance, Standard deviation, Mean Absolute Deviation using sample dataset.

1. **Variability**

**Variability** refers to how spread out or dispersed the values in a dataset are. It describes the extent to which the data points differ from each other and from the central tendency (mean or median). High variability indicates that the data points are spread out over a wider range of values, while low variability suggests that they are clustered closely around the central value.



# Variance

**Variance** is a quantitative measure of the dispersion or spread of a set of values. It calculates the average of the squared differences between each data point and the mean of the dataset. The formula for variance is:



Where:

* + xi = each value in the dataset
  + μ = mean of the dataset
  + n = number of observations in the dataset

# 3. Standard Deviation

* + **Standard Deviation** is the square root of the variance. It provides a measure of variability in the same units as the original data, making it easier to interpret compared to variance. The formula for standard deviation is:



# Mean Absolute Deviation (MAD)

**Mean Absolute Deviation (MAD)** measures the average absolute deviations of each data point from the mean. Unlike variance, which squares the deviations, MAD takes the absolute value, providing a more intuitive sense of variability without emphasizing larger deviations.



Where:

* + ∣xi−μ∣ = absolute deviation of each value from the mean

# Mean Absolute Deviation (MAD) is particularly useful in situations where:

* + You need a robust measure that is not influenced by outliers.
  + The data is skewed or non-normally distributed.
  + You require an intuitive and interpretable measure of variability.
  + You are comparing the variability of multiple datasets.
  + You are working in forecasting or decision-making scenarios where understanding average error is critical.

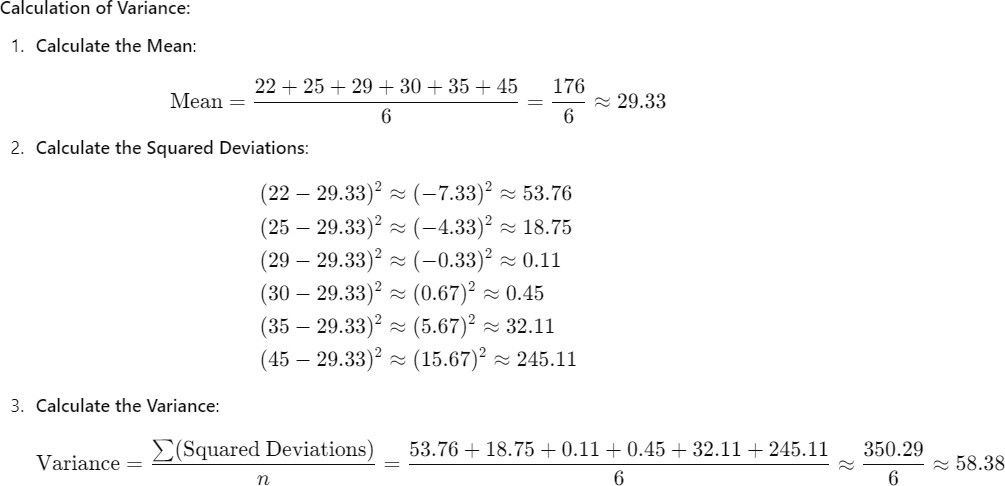
# Sample Dataset

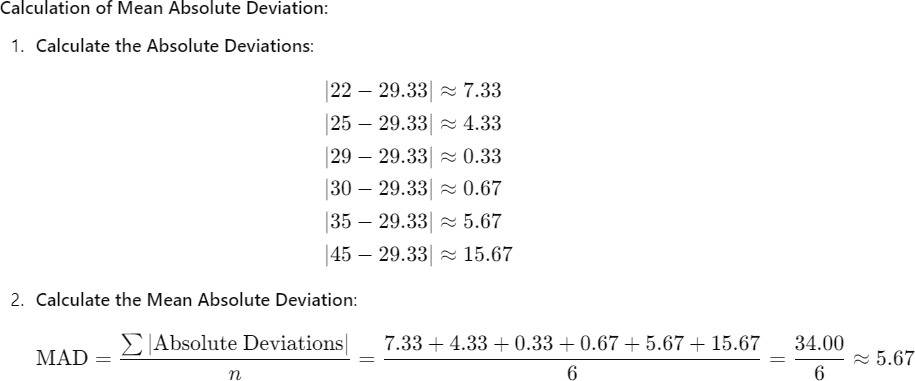
Let's use the following dataset representing the ages of a group of individuals: Ages = [22, 25, 29, 30, 35, 45]

**Variance** measures the average squared deviation of each data point from the mean. It provides a quantitative measure of variability in the dataset.









# Summary of Results

* + **Mean**: ≈29.33\approx 29.33≈29.33
  + **Variance**: ≈58.38\approx 58.38≈58.38
  + **Standard Deviation**: ≈7.63\approx 7.63≈7.63
  + **Mean Absolute Deviation (MAD)**: ≈5.67\approx 5.67≈5.67

# Conclusion

* + **Variance** and **Standard Deviation** give a sense of how much the ages vary from the mean in a squared form, where standard deviation provides the average distance from the mean in the original units.
  + **Mean Absolute Deviation** gives an alternative measure of variability, focusing on absolute distances without squaring, making it more interpretable in terms of the original dataset values.

Understanding these measures helps in analyzing data distributions and assessing how spread out the data points are, which is crucial in many statistical analyses and machine learning applications.

## Variance, Standard Deviation, and Mean Absolute Deviation are essential statistical measures in machine learning that provide valuable insights into data variability, model performance, and prediction accuracy. Why does variability matter?

* While the central tendency tells us where most of your points lie, variability summarizes how far apart they are. This is important because the amount of

variability determines how well you can generalize results from the sample to your population.

* Low variability is ideal because it means that you can better predict information about the population based on sample data. High variability means that the values are less consistent, so it’s harder to make predictions.

Data sets can have the same central tendency but different levels of variability or [vice versa](https://www.scribbr.com/definitions/vice-versa/). Both of them together give you a complete picture of your data.

* **Example:** Variability in normal distributions

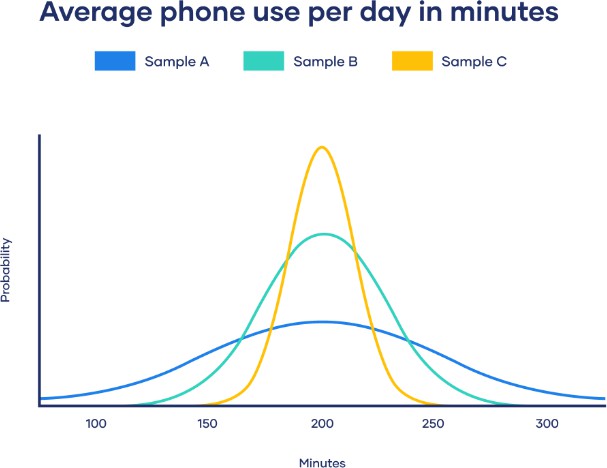
You are investigating the amounts of time spent on phones daily by different groups of people.

Using [simple random samples](https://www.scribbr.com/methodology/simple-random-sampling/), you collect data from 3 groups: Sample A: high school students,

Sample B: college students,

Sample C: adult full-time employees

# Finding the percentile, inter quartile range, Box Plot, Outlier, whisker using sample dataset.



All three of your samples have the same average phone use, at 195 minutes or 3 hours and 15 minutes. This is the x-axis value where the peak of the curves are.

Although the data follows a normal distribution, each sample has different spreads. Sample A has the largest variability while Sample C has the smallest variability.

# Percentiles

* **Definition**: Percentiles divide a dataset into 100 equal parts, indicating the value below which a certain percentage of observations fall. For example, the 25th

percentile (Q1) is the value below which 25% of the data lies.

The **range** is a basic measure of **variability** or **dispersion** in a dataset. It is defined as the difference between the **maximum** and **minimum** values in the dataset. The range gives a sense of how spread out the data is but does not provide insights into the distribution of values between the extremes.

# Interquartile Range (IQR)

The Interquartile Range (IQR) is a measure of statistical dispersion and is calculated as the difference between the third quartile (Q3) and the first quartile (Q1) of a dataset.

* It focuses on the middle 50% of the data and is resistant to outliers, making it a more robust measure of spread than the range.

## IQR=Q3−Q1

Where:

* **Q1** (First Quartile): The 25th percentile, meaning 25% of the data is below this value.
* **Q3** (Third Quartile): The 75th percentile, meaning 75% of the data is below this value.

## Five-number summary

Every distribution can be organized using a **five-number summary**:

* Lowest value
* Q1: 25th percentile
* Q2: the median
* Q3: 75th percentile
* Highest value (Q4)

These five-number summaries can be easily visualized using box and whisker plots.

# Box Plot

**Definition**: A box plot (or whisker plot) is a graphical representation of data that displays the median, quartiles, and potential outliers. The main components of a box plot include:

* **Box**: Represents the IQR (from Q1 to Q3).
* **Whiskers**: Extend to the smallest and largest values within 1.5 times the IQR from the quartiles.
* **Outliers**: Points outside the whiskers.

## Importance in Machine Learning:

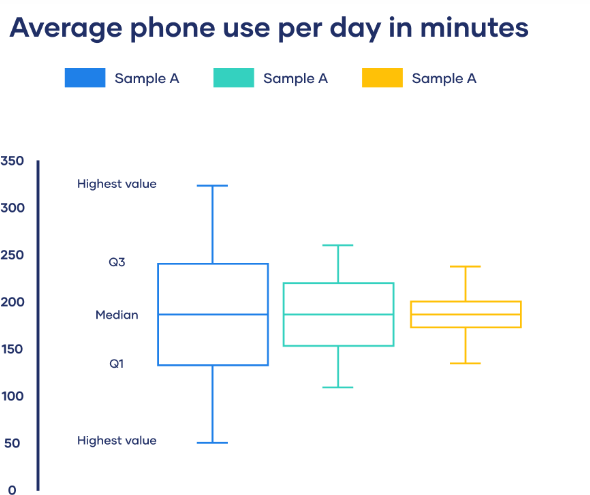
* **Visualizing Data Distribution**: Box plots offer a clear visual representation of data distribution, central tendency, and spread, making it easier to spot patterns or anomalies.
* **Identifying Skewness**: The symmetry of the box and the lengths of the whiskers can indicate whether the data is skewed, informing potential transformations (e.g., log transformation).

# Whiskers

**Definition**: In a box plot, whiskers extend from the box (IQR) to the smallest and largest values within 1.5 times the IQR from the quartiles. Values beyond this range are considered outliers.

## Importance in Machine Learning:

* **Understanding Range**: Whiskers help visualize the spread of the data beyond the quartiles, providing context about the overall range and potential variability.
* Box and whisker plot example For each of our samples, the horizontal lines in a box show Q1, the median and Q3, while the whiskers at the end show the highest and lowest values.



## Outliers

**Definition**: Outliers are data points that deviate significantly from the majority of the data. They can arise from measurement errors, data entry errors, or genuine variability in the data.

## Importance in Machine Learning:

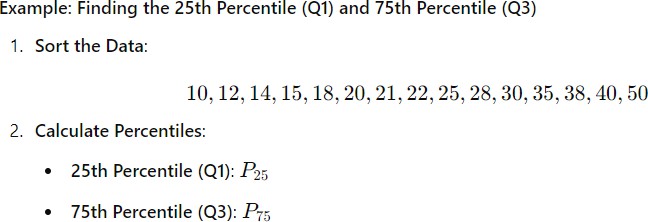
* + **Impact on Model Performance**: Outliers can disproportionately influence statistical measures like mean and variance, leading to skewed model performance.
  + **Preprocessing Step**: Identifying and handling outliers (by removing or transforming them) is a crucial preprocessing step.

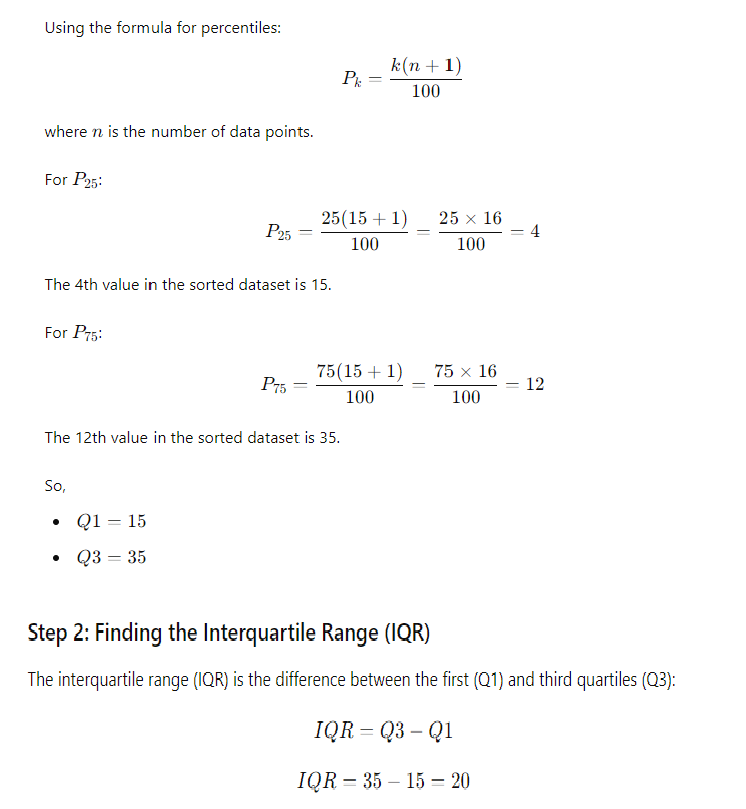
# Sample Dataset

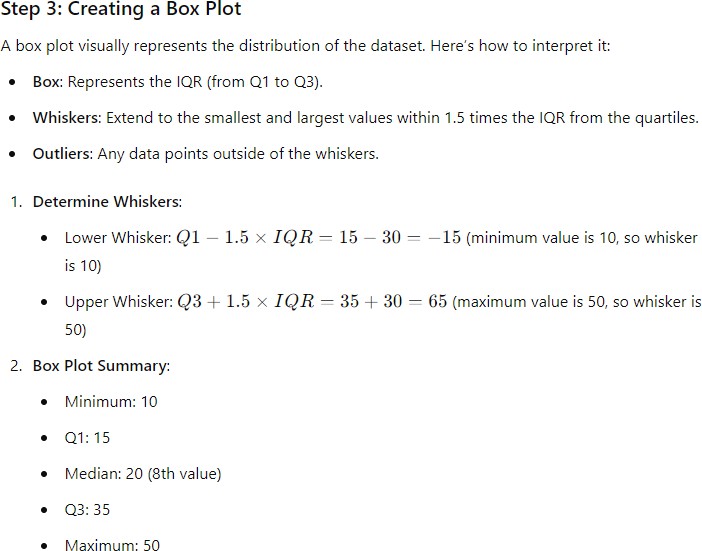
Let's use the following sample dataset of 15 numerical values: 12,15,14,10,18,20,22,21,30,25,28,35,40,38,50

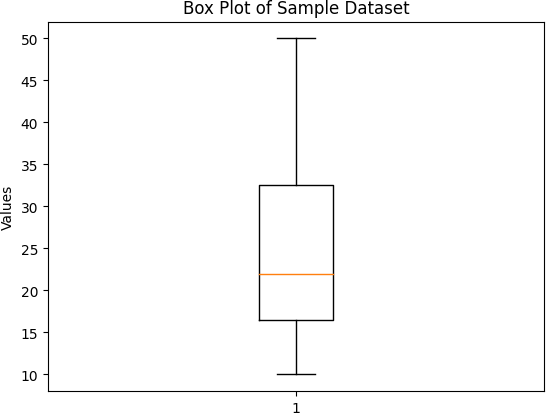
## Step 1: Finding Percentiles

Percentiles divide a dataset into 100 equal parts. The 𝑘kth percentile is the value below which 𝑘%of the data falls.







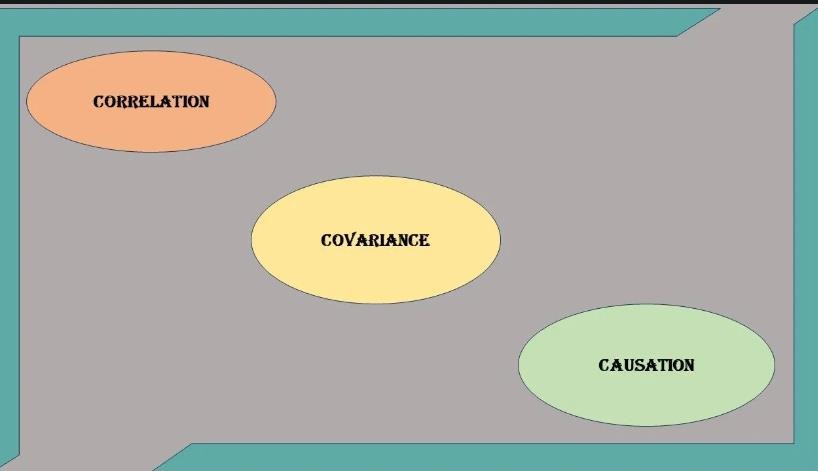


## Conclusion

Understanding these concepts is crucial for effective data preprocessing and analysis in machine learning. They are used to:

* Analyze the distribution and spread of data,
* Detect and handle outliers,
* Visualize data effectively.

**Calculating correlation, covariance, causation.**

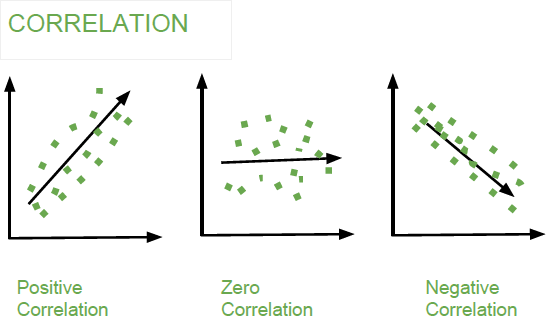
****

# Correlation:

**Correlation** measures the strength and direction of the linear relationship between two variables. It is normalized between **-1** and **+1**:

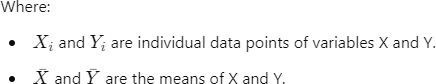
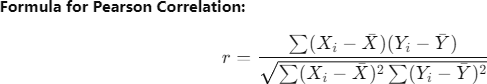
• **+1**: Perfect positive correlation.

* + **0**: No correlation.
  + **-1**: Perfect negative correlation.



## Types of Correlation:

* + **Pearson Correlation**: Measures linear relationships.
  + **Spearman Rank Correlation**: Measures the strength of the relationship between two ranked variables (handles non-linear data).



## Correlation in Machine Learning

**Use Case**: Correlation is used to measure the linear relationship between features (variables) in a dataset. This is important during **feature selection** and **exploratory data analysis**.

## Example:

* + In a dataset with features like **height** and **weight**, there is likely to be a high correlation between the two. Including both height and weight in a model may lead to redundancy. A high correlation value will inform you to remove one of the features to avoid multicollinearity.

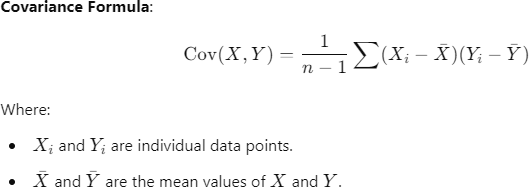
## When to Use Correlation in ML:

* + **Feature Selection**: Before training a model, it's important to eliminate redundant features. If two features have a high correlation, they provide similar information to the model, and keeping both can cause overfitting or inefficiencies.
  + **Multicollinearity**: In models like **linear regression**, highly correlated features can lead to **multicollinearity**, which reduces the interpretability of the model and can make the coefficients unstable.

# Covariance:

**Covariance** measures how much two variables vary together.

* + A positive value indicates that as one variable increases, the other tends to increase as well.
  + A negative value indicates an inverse relationship. Covariance doesn't give the strength of the relationship and is not standardized like correlation.



## Covariance in Machine Learning

**Use Case**: Covariance is used in **dimensionality reduction** techniques like **Principal Component Analysis (PCA)** to understand how features vary together. This helps in reducing the number of features while retaining as much information as possible.

## Example:

* + In a dataset with features like **height** and **weight**, covariance helps PCA determine how much of the variation in the dataset is captured by these two variables. PCA will project the data into new dimensions, minimizing redundancy while keeping important information.

## When to Use Covariance in ML:

* + **Principal Component Analysis (PCA)**: PCA uses the covariance matrix to find the direction (principal components) along which the variance in the data is maximized. It reduces the dimensionality of the dataset by transforming it into a set of uncorrelated variables.

# Causation:

**Causation** refers to a cause-and-effect relationship where one variable directly affects another. While correlation and covariance measure relationships, they do not establish causality.

## Correlation does not imply causation.

**Example:**

* + If an increase in advertising leads to increased sales, you may infer a causal relationship, but you would need further analysis, experiments, or domain knowledge to confirm it.

## Causation in Machine Learning

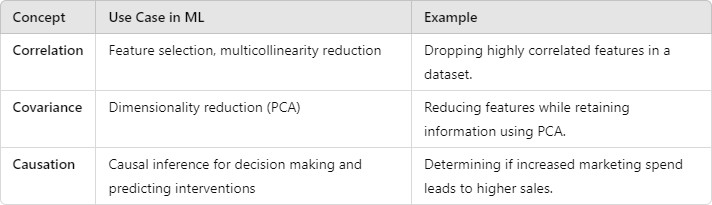
**Use Case**: While correlation and covariance show relationships between variables, **causation** is crucial in real-world decision-making, especially in fields like healthcare, economics, or marketing.

## Example:

* + In healthcare, you may want to know if a particular drug **causes** a reduction in blood pressure. A model may show a correlation between taking the drug and lower blood pressure, but that doesn’t confirm causality. You would need to conduct randomized controlled trials (RCTs) or apply **causal inference techniques** to establish a cause-and- effect relationship.

## When to Use Causation in ML:

* + **Causal Inference**: Understanding causation is important when making decisions or predictions. In models where you predict the outcome of an intervention (e.g., increasing marketing spend to increase sales), you must determine if one variable is causing changes in another.
  + **Time Series Analysis**: In time series data, identifying causality between variables can improve forecasts and interventions. For example, you might analyze the causal relationship between a stock price and market indicators to make informed trading decisions.



## Conclusion:

* + **Correlation** helps in understanding linear relationships and assists in feature selection.
  + **Covariance** helps in dimensionality reduction techniques, ensuring the most relevant information is retained.
  + **Causation** is critical in making informed decisions, especially when predicting real- world outcomes.

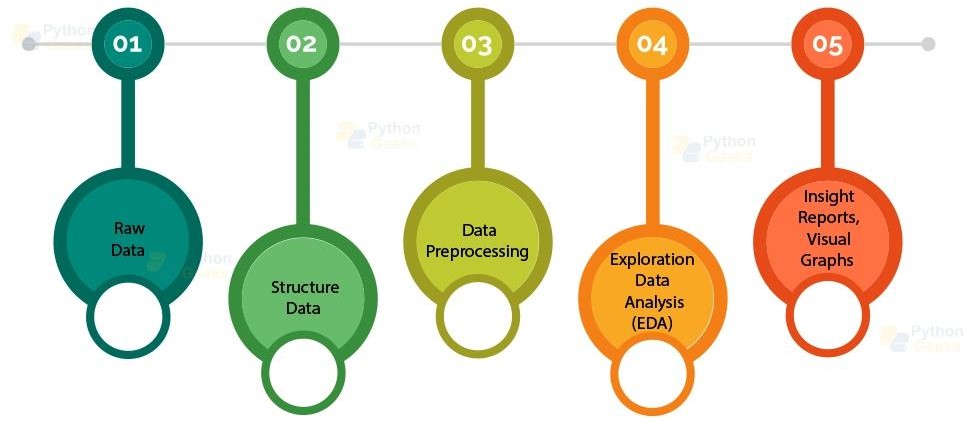
Data Preprocessing

# Data Preprocessing in Machine Learning

* Data preprocessing involves preparing raw data for machine learning models. It is a key step to ensure that the model receives clean, structured, and normalized data.

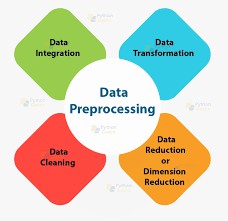
# Why Do We Need Data Preprocessing?

Real-time data contains lots of missing values and distortions. For machine learning to give correct outputs, it must go through a series of steps so the data is organised and is in a standard format. Cleaned data increases the accuracy and efficiency of the learning model.



# 4 Steps in Data Preprocessing

* Data Cleaning
* Data Integration
* Data Transformation
* Data Reduction



Data Cleaning:

* + Ensures the data is accurate, complete, and consistent. Without cleaning, the model could produce biased or incorrect results.
  + Example: Handling missing values ensures the model doesn’t have gaps or make

wrong assumptions about certain patterns.

* + Data Cleaning is particularly done as part of data preprocessing to clean the data by:

Missing values Noisy Data Removing outliers

Data Integration:

* + Merges data from multiple sources, enriching the dataset and providing more context for the model to learn from.
  + Example: Integrating transactional data from different departments within a company could lead to a more comprehensive fraud detection model.
  + Data Integration is one of the data preprocessing steps that are used to merge the data present in multiple sources into a single larger data store like a data warehouse.

Data Transformation:

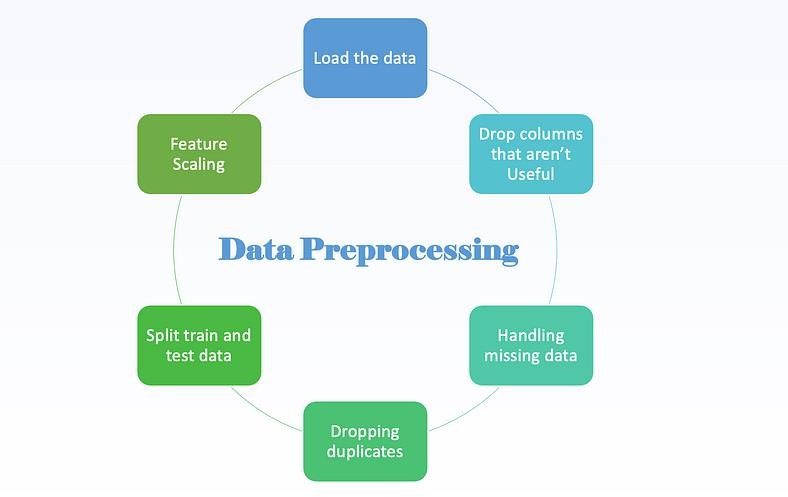
* + Prepares data for machine learning algorithms. Without transformation, some models may struggle to learn from raw data due to scale differences or categorical variables.
  + Example: Standardizing numerical data is critical for algorithms like SVM, as it ensures all features are treated equally.
  + Once data cleaning has been done, we need to consolidate the quality data into alternate forms by changing the value, structure, or format of data using the Data Transformation strategies.

Data Reduction:

Reduces the complexity and size of the dataset, which helps prevent overfitting and speeds up computation time.

* + Example: PCA is often used in image processing tasks to reduce the number of features while retaining the most important information.
  + The size of the dataset in a data warehouse can be too large to be handled by data analysis and data mining algorithms.

One possible solution is to obtain a reduced representation of the dataset that is much smaller in volume but produces the same quality of analytical results.



# Drop columns that aren’t useful

Often, a DataFrame will contain columns that are not useful to your analysis. Such columns should be dropped from the DataFrame to make it easier for you to focus on the remaining columns. It can be done as follow:

# Handling Missing Values:

Techniques include removing instances with missing values, imputing missing values with the mean, median, or mode.

# Removing Duplicates

Identifying and removing duplicate instances to ensure the dataset is clean.

# Splitting Data

Dividing the dataset into training and testing sets to evaluate the model's performance.

# Feature Scaling

**Normalization:** Rescaling the features to a range of [0, 1].

**Standardization:** Rescaling the features to have a mean of 0 and a standard deviation of

# Encoding Categorical Data:

**One-Hot Encoding:** Converting categorical variables into binary vectors.

**Label Encoding:** Converting categorical variables into integer values.

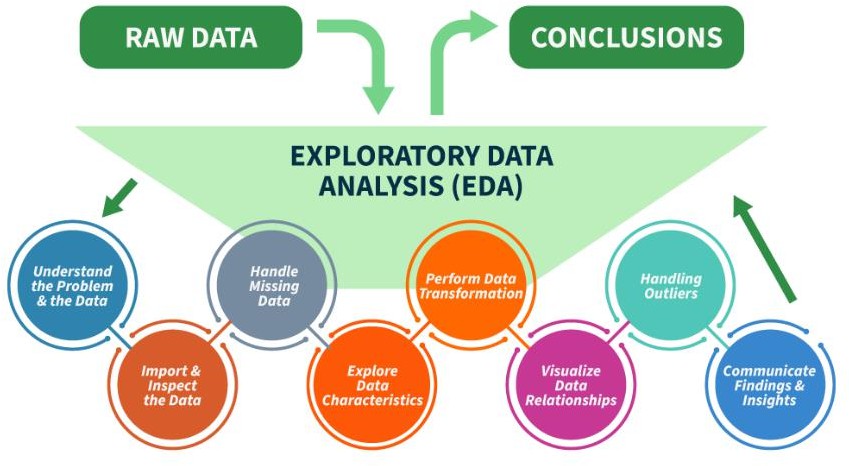
# Conclusion

Preprocessing ensures the data fed into machine learning models is clean, consistent, and properly structured. Without preprocessing, the model could underperform or produce inaccurate results. For example, missing values, unscaled data, and categorical data that are not encoded could cause models to behave poorly.

**Impact on Model Performance**: Good preprocessing can reduce the complexity of the model, improve generalization, and speed up convergence in training.

# Exploratory Data Analysis (EDA)

EDA is the process of analyzing and summarizing the dataset before applying machine learning models. It provides insights into the data’s structure, patterns, relationships between variables, and potential issues like outliers and missing values.



# Key aspects of EDA include

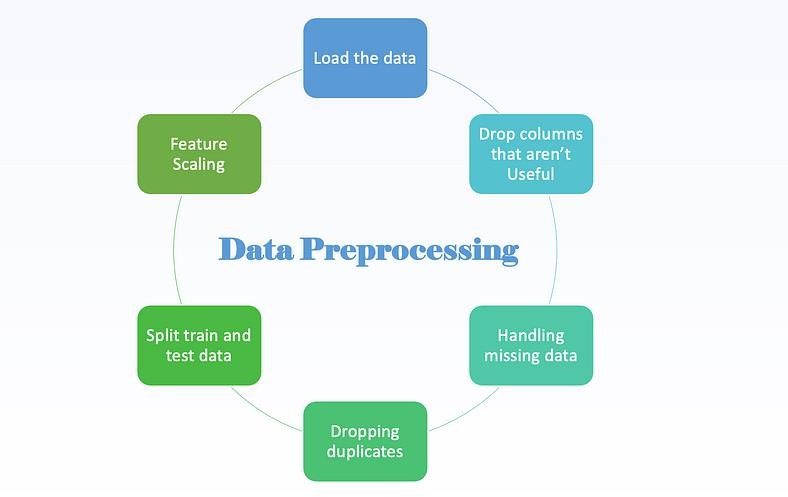
* **Distribution of Data**: Examining the distribution of data points to understand their range, central tendencies (mean, median), and dispersion (variance, standard deviation).
* **Graphical Representations**: Utilizing charts such as histograms, box plots, scatter plots, and bar charts to visualize relationships within the data and distributions of variables.
* **Outlier Detection**: Identifying unusual values that deviate from other data points. Outliers can influence statistical analyses and might indicate data entry errors or unique cases.
* **Correlation Analysis**: Checking the relationships between variables to understand how they might affect each other. This includes computing correlation coefficients and creating correlation matrices.
* **Handling Missing Values**: Detecting and deciding how to address missing data points, whether by imputation or removal, depending on their impact and the amount of missing data.
* **Summary Statistics:** Calculating key statistics that provide insight into data trends and nuances.

Data preparation and preprocessing steps using sample dataset

Data preparation and preprocessing are crucial steps in building a machine learning model. The main goal is to clean, format, and structure the dataset to make it suitable for model training and

prediction.

Lets walk through the main steps of data preparation and preprocessing using a sample dataset, including handling missing values, outlier detection, data scaling, encoding categorical variables, and splitting the data into training and testing sets.



**Steps in Data Preparation and Preprocessing**

1. **Loading the Dataset**
2. **Handling Missing Values**
3. **Handling Categorical Variables (Encoding)**
4. **Outlier Detection and Handling**
5. **Feature Scaling**
6. **Data Splitting (Training and Testing Sets)**
7. **Loading the Dataset**

This is the initial step where the data is imported into the working environment. Depending on the data source, this can involve:

* + **Reading from CSV/Excel files**: Using libraries like pandas in Python to load data from files.
  + **Dataframe Creation**: Once loaded, the data can be represented in a structured format (e.g., a DataFrame) for easier manipulation.

1. **Handling Missing Values**

Missing values can introduce bias and inaccuracies in models, so they must be addressed:

* + **Identifying Missing Values**: Use functions to detect missing values (e.g., isnull() in pandas).
  + **Imputation**: Filling in missing values using various strategies:
    - **Mean/Median/Mode Imputation**: Replacing missing values with the mean, median, or mode of the column.
  + **Dropping**: If a significant amount of data is missing from a specific row or column, it may be appropriate to remove it entirely.

1. **Handling Categorical Variables (Encoding)**

Categorical variables need to be transformed into a numerical format that machine learning algorithms can process:

* + **Label Encoding**: Assigning a unique integer to each category (suitable for ordinal data).

Education levels are a classic example of ordinal data because they have a meaningful order, but the difference between them is not necessarily equal.

### **Example Dataset:**

| **Education Level** | **Ordinal Value** |
| --- | --- |
| No Schooling | 0 |
| Primary Education | 1 |
| Secondary Education | 2 |
| Bachelor's Degree | 3 |
| Master's Degree | 4 |
| PhD | 5 |

**One-Hot Encoding**: Creating binary columns for each category (suitable for nominal data) to avoid the algorithm assuming a hierarchy in categories.

One-hot encoding is a method of converting categorical data into a binary matrix, where each category is represented as a separate column with values of **0** or **1**.

**Example:**

Suppose we have a categorical feature **"Color"** with three unique values:

| **Color** |
| --- |
| Red |
| Green |
| Blue |
| Green |
| Red |
| Blue |

**One-Hot Encoding Transformation:**

| **Color** | **Red** | **Green** | **Blue** |
| --- | --- | --- | --- |
| Red | 1 | 0 | 0 |
| Green | 0 | 1 | 0 |
| Blue | 0 | 0 | 1 |
| Green | 0 | 1 | 0 |
| Red | 1 | 0 | 0 |
| Blue | 0 | 0 | 1 |

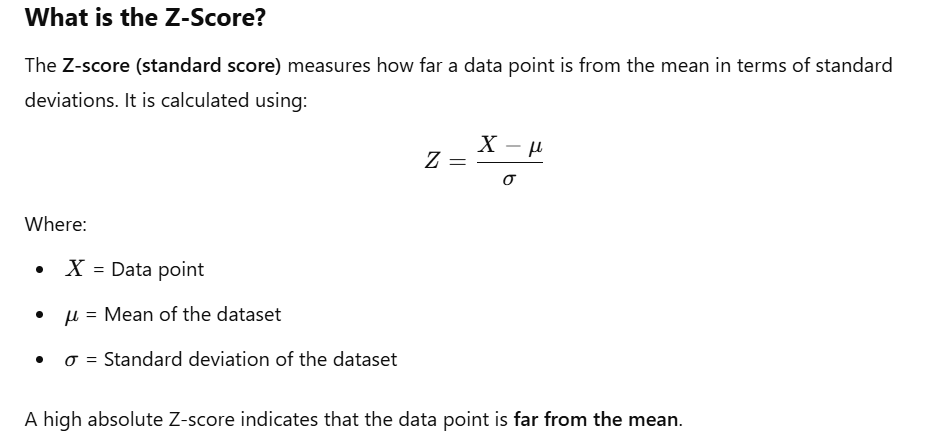
Each unique category becomes its own column, and we use **1** to indicate presence and **0** for absence.

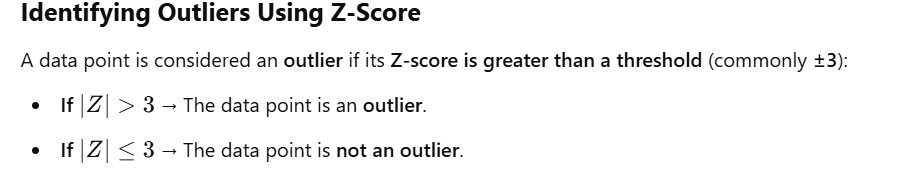
* + **Binary Encoding**: Combines label encoding and one-hot encoding, useful for high cardinality features.

1. **Outlier Detection and Handling**

Outliers can skew results and lead to misleading insights. Identifying and handling them is essential:

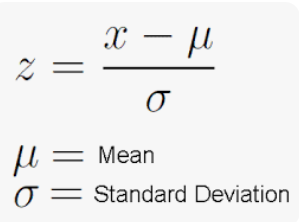
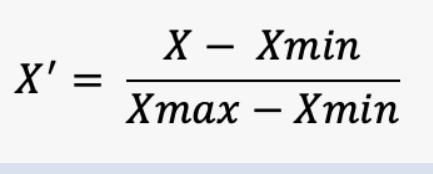
* + **Detection Methods**:
    - **Statistical Tests**: Z-score or IQR (Interquartile Range) method to identify outliers.



* + - 
    - **Visualization**: Using box plots or scatter plots to visually assess outliers.
  + **Handling Methods**:
    - **Removal**: If outliers are errors or irrelevant, they can be excluded from the dataset.
    - **Transformation**: Applying transformations (e.g., logarithmic) to reduce the impact of outliers.

1. **Feature Scaling**

Feature scaling ensures that numerical features are on a similar scale, which is vital for algorithms sensitive to the scale of data (e.g., gradient descent-based algorithms):

* + **Standardization (Z-score Normalization)**: Transforming features to have a mean of 0 and a standard deviation of 1.
  + 
  + **Min-Max Scaling**: Scaling features to a range between 0 and 1.
  + 

1. **Data Splitting (Training and Testing Sets)**

Splitting the data into training and testing sets is essential for evaluating the performance of a model:

* + **Training Set**: Used to train the model (typically 70-80% of the data).
  + **Testing Set**: Used to evaluate the model's performance on unseen data (typically 20-30% of the data).
  + **Cross-Validation**: Further validation techniques (like k-fold cross-validation) can be employed to ensure the model's robustness by training and testing it on different subsets of the data multiple times.
  + **Step 1: Loading the Dataset**

We first load a sample dataset for preprocessing. Let’s use a sample dataset with numerical and categorical data.

import pandas as pd # Sample dataset data = {

'Age': [25, 30, 45, 50, None, 22],

'Income': [50000, 60000, None, 80000, 70000, 30000],

'Gender': ['Male', 'Female', 'Male', 'Female', 'Female', None],

'Purchased': ['Yes', 'No', 'Yes', 'No', 'Yes', 'No']

}

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

* + **Step 2: Handling Missing Values**

Missing data is common in real-world datasets. You can either remove rows/columns with missing values or impute them (replace them with meaningful values like mean, median, or mode).

# Handling missing values by imputation

df['Age'].fillna(df['Age'].mean(), inplace=True) # Filling missing 'Age' with mean

df['Income'].fillna(df['Income'].median(), inplace=True) # Filling missing 'Income' with median df['Gender'].fillna(df['Gender'].mode()[0], inplace=True) # Filling missing 'Gender' with mode print(df)

# Step 3: Handling Categorical Variables (Encoding)

Machine learning algorithms need numerical data to work. Hence, categorical data must be converted into numbers using techniques like One-Hot Encoding or Label Encoding.

One-Hot Encoding for nominal categorical variables like Gender in this example.

from sklearn.preprocessing import OneHotEncoder # Applying One-Hot Encoding on 'Gender'

df = pd.get\_dummies(df, columns=['Gender'], drop\_first=True) # 'drop\_first=True' avoids dummy variable trap

print(df)

Label Encoding for ordinal categorical variables:

For the Purchased column (which is binary), we can use Label Encoding. from sklearn.preprocessing import LabelEncoder

# Applying Label Encoding on 'Purchased'

le = LabelEncoder()

df['Purchased'] = le.fit\_transform(df['Purchased']) print(df)

# Step 4: Outlier Detection and Handling

Outliers can distort the model's performance. To handle outliers, you can either remove them or cap them.

# Detecting outliers using IQR

Q1 = df['Income'].quantile(0.25) Q3 = df['Income'].quantile(0.75)

IQR = Q3 - Q1

# Filter out outliers based on IQR (Income column)

outliers = df[(df['Income'] < (Q1 - 1.5 \* IQR)) | (df['Income'] > (Q3 + 1.5 \* IQR))]

print(outliers)

# Alternatively, remove or cap the outliers

df['Income'] = df['Income'].clip(Q1 - 1.5 \* IQR, Q3 + 1.5 \* IQR) print(df)

# Step 5: Feature Scaling

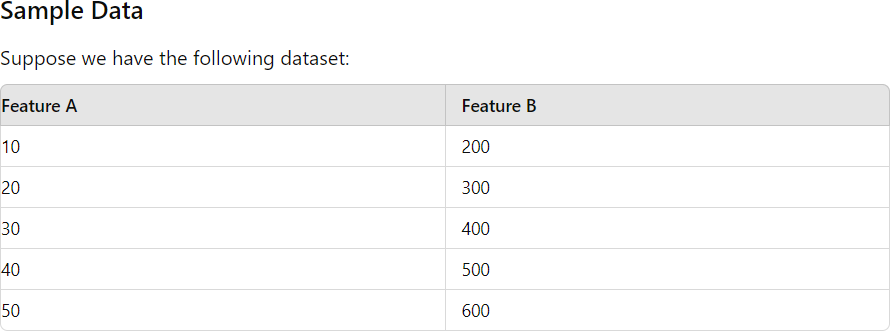
Feature scaling ensures that all numerical features are on the same scale,

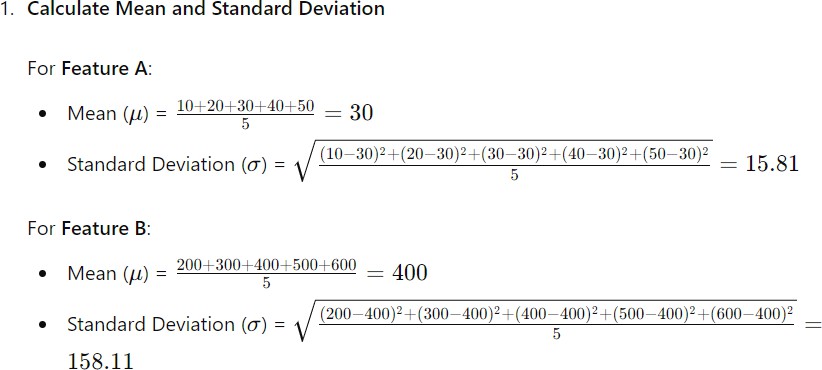
which is important for algorithms like KNN, SVM, and Gradient Descent-based models.

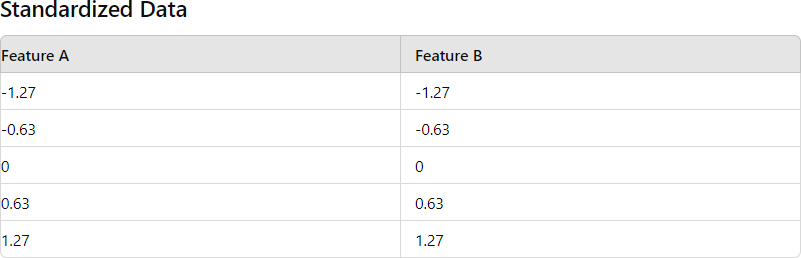
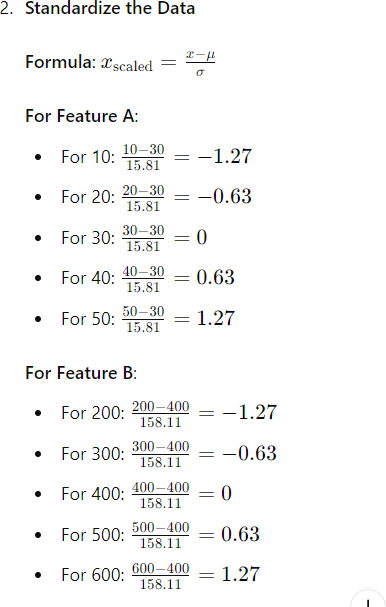
from sklearn.preprocessing import StandardScaler # Standardize numerical features 'Age' and 'Income' scaler = StandardScaler()

df[['Age', 'Income']] = scaler.fit\_transform(df[['Age', 'Income']]) print(df)

let’s go through an example of using StandardScaler with sample data.



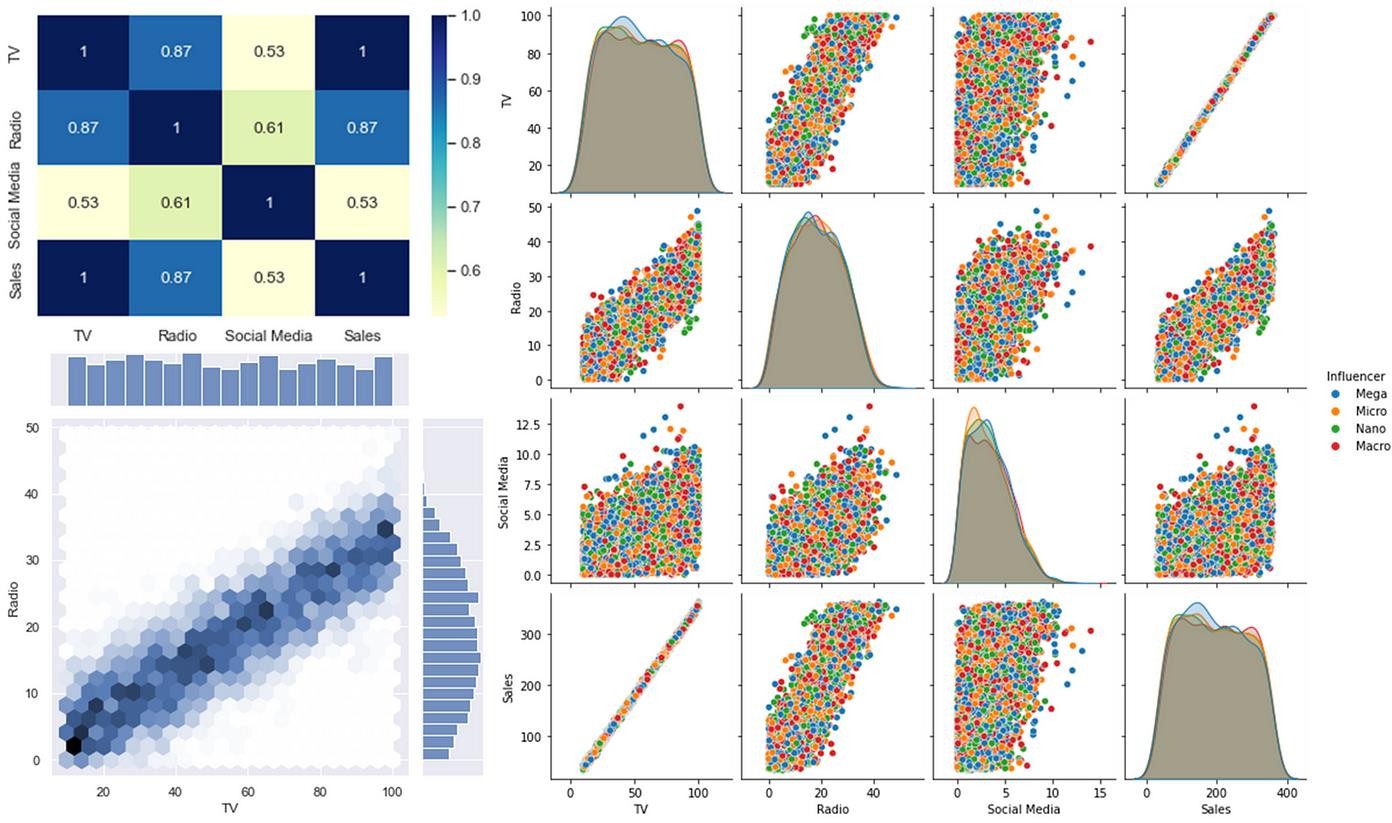




# Visualization and its tools

Visualization is the process of representing data, information, or concepts visually using graphs, charts, maps, and other visual formats to make it easier to understand and interpret.

It is widely used across industries such as business, education, healthcare, and technology to uncover insights, patterns, and trends in data.

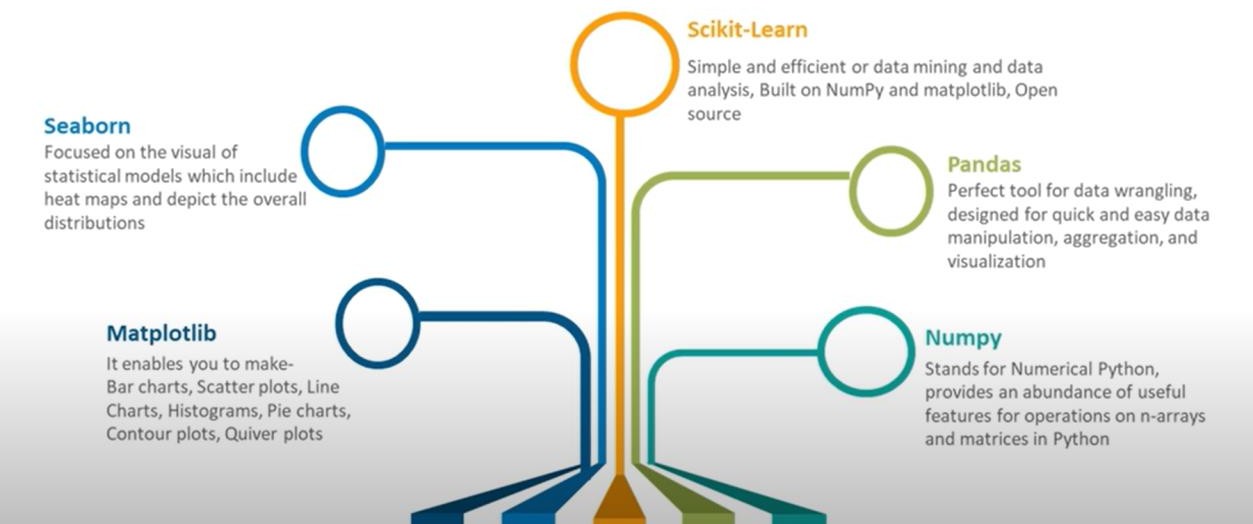


* In **Machine Learning (ML)**, visualization plays a critical role in understanding the data, model performance, and decision-making processes. It helps in data exploration, feature selection, model evaluation, and interpreting complex models. By visualizing data and results, we can gain insights into patterns, correlations, and potential issues such as overfitting or bias in the model.

**Importance of Visualization in Machine Learning:**

* **Data Exploration**: Understand distributions, correlations, outliers, and missing data.
* **Feature Engineering**: Identify which features contribute the most to the model’s predictions.
* **Model Diagnostics**: Visualize the performance of models, compare different models, and tune hyperparameters.
* **Interpretability**: Make complex machine learning models more understandable for human users (e.g., decision trees, feature importance plots).

**What libraries do we use for Machine learning?**

****

**Popular Tools for Visualization in Machine Learning:**

1. **Matplotlib**:
   1. A widely used library in Python for creating static, animated, and interactive plots.
   2. Useful for plotting simple line charts, bar charts, scatter plots, histograms, etc.
2. **Seaborn**:
   1. Built on top of Matplotlib, Seaborn provides beautiful, high-level interface visuals like heatmaps, violin plots, and pair plots.

# Matplotlib

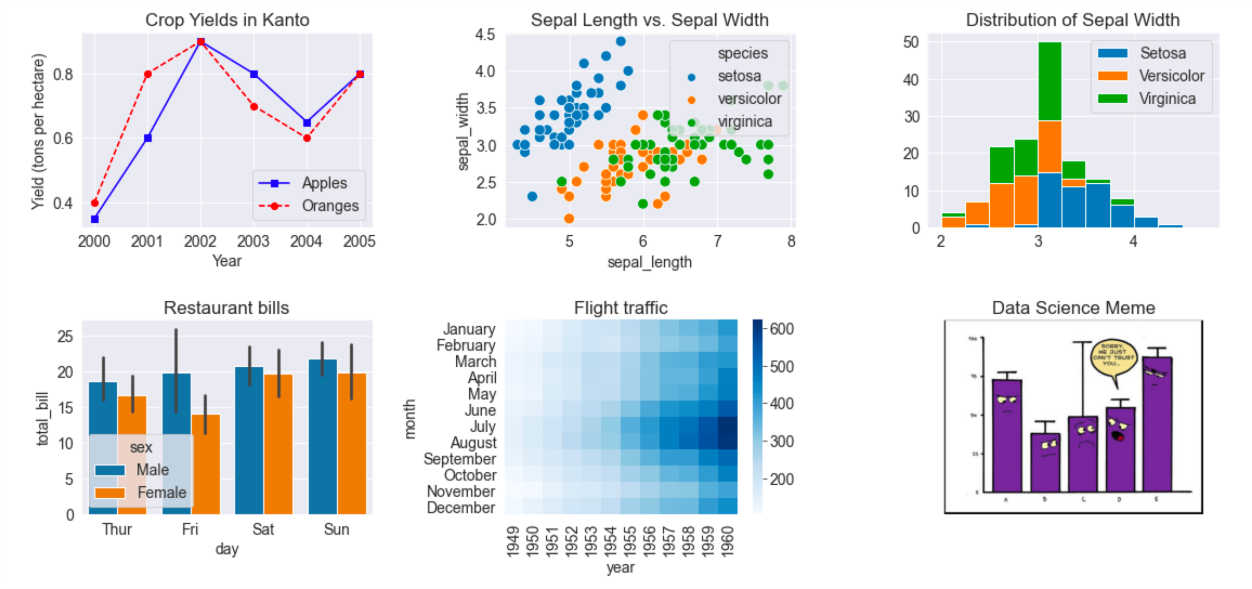
Matplotlib is a Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It is needed when a programmer wants to visualize the patterns in the data.

A module named pyplot makes it easy for programmers for plotting .

# Seaborn

**Seaborn** is a Python visualization library based on Matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics.

It is particularly useful for making complex statistical plots with minimal code and for visualizing data distributions and relationships.



# Importing necessary libraries import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

# Creating a simple dataset data = {

'Age': [22, 25, 47, 52, 46, 56, 29, 21, 28, 32],

'Salary': [50000, 60000, 120000, 110000, 105000, 130000, 48000, 40000, 52000, 58000],

'Purchased': [0, 0, 1, 1, 1, 1, 0, 0, 0, 1]

}

# Convert to DataFrame

df = pd.DataFrame(data) # Display the dataset print(df)

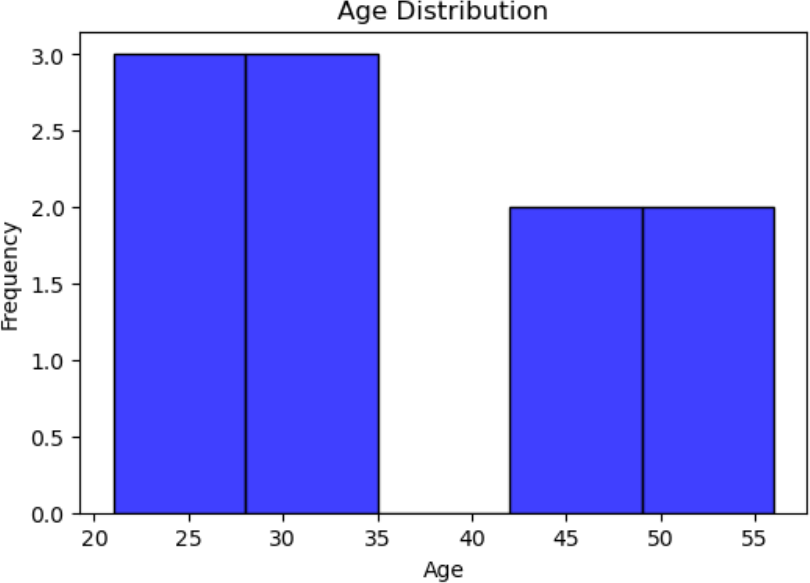
**#Histogram for Age Distribution**

**# Plotting a histogram for Age distribution**

plt.figure(figsize=(6,4)) sns.histplot(df['Age'], color='blue') plt.title('Age Distribution')

plt.xlabel('Age')

plt.ylabel('Frequency') plt.show()



**#Scatter Plot for Age vs. Salary # Scatter plot for Age vs Salary plt.figure(figsize=(6,4))**

**sns.scatterplot(x='Age', y='Salary', data=df, hue='Purchased', palette='coolwarm', s=100)**

**plt.title('Age vs Salary (Colored by Purchase Status)')**

**plt.xlabel('Age') plt.ylabel('Salary') plt.show()**

****

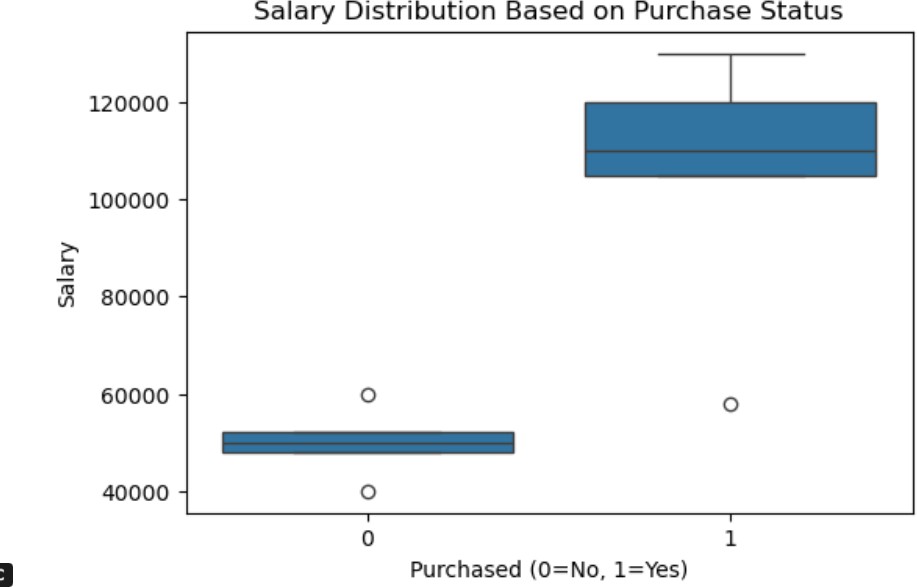
**#Boxplot for Salary Based on Purchase Status**

**# Boxplot to compare Salary for Purchased and Not Purchased plt.figure(figsize=(6,4))**

**sns.boxplot(x='Purchased', y='Salary', data=df)**

**plt.title('Salary Distribution Based on Purchase Status') plt.xlabel('Purchased (0=No, 1=Yes)')**

**plt.ylabel('Salary') plt.show()**

****

# Pie Chart

sales = [5000, 7000, 4000, 8000, 6000]

categories = ['Electronics', 'Clothing', 'Groceries', 'Furniture', 'Books']

colors = ['skyblue', 'salmon', 'lightgreen', 'gold', 'plum']

# Create Pie Chart

plt.figure(figsize=(6, 6))

plt.pie(sales, labels=categories, autopct='%1.1f%%', colors=colors, startangle=140)

plt.title("Sales Distribution Across Different Categories")

# Show Plot

plt.show()

