ÉCOLE SUPÉRIEURE EN SCIENCES ET TECHNOLOGIES DE L'INFORMATIQUE ET DU NUMÉRIQUE

FUNDAMENTALS OF DATA SCIENCE AND DATA MINING

CHAPTER 3: EXPLORATORY DATA ANALYSIS

Dr. Chemseddine Berbague

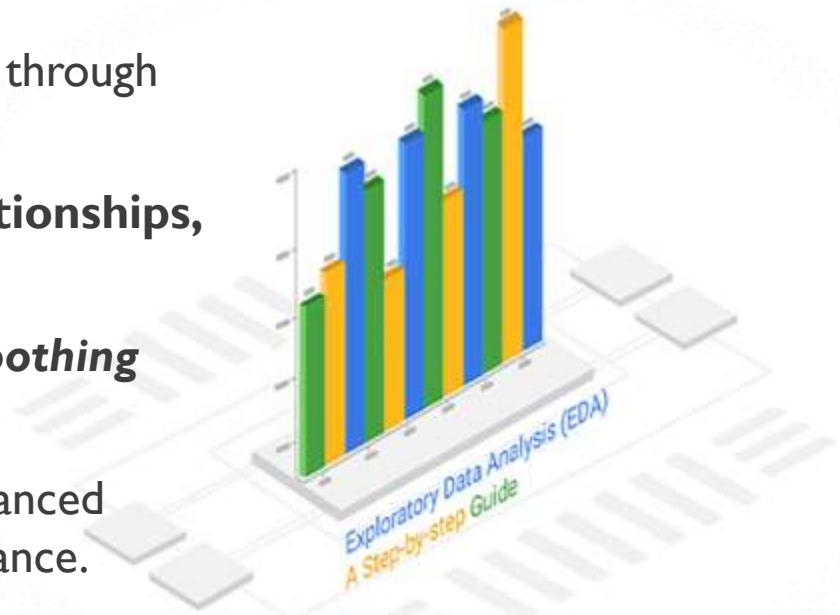
2024-2025

CONTENT

- Introduction to EDA
  - Handling outliers
  - Analyzing pairwise variables
  - Analyzing categorical variables
- Data Transformation
- Data Smoothing
- Basic Data preprocessing
  - Advanced data preprocessing: Feature engineering

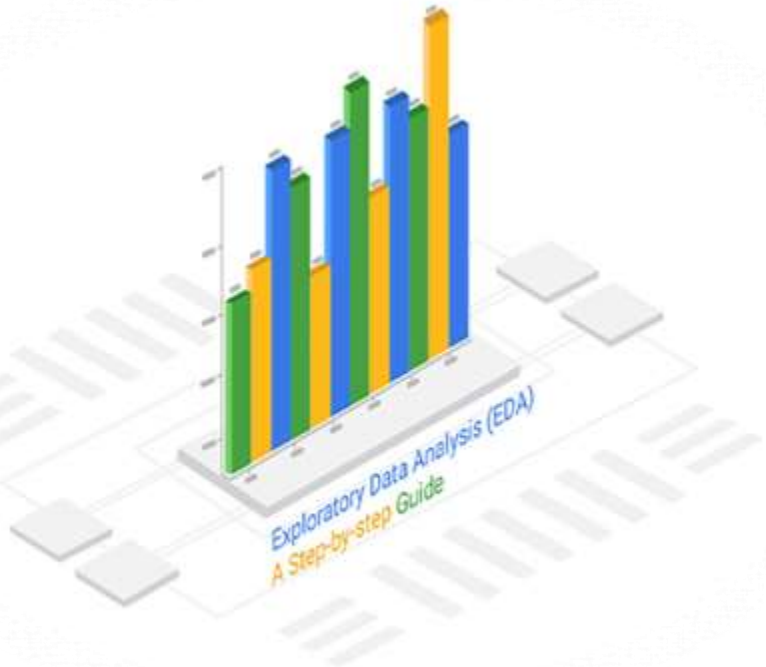
INTRODUCTION

- Exploratory Data Analysis (EDA) focuses on **understanding data** through **visualization** and summary statistics.
- It involves **handling outliers, exploring pairwise variable relationships, and analyzing categorical variables.**
- **Data transformation** ensures models work effectively, while ***smoothing*** reduces noise to highlight trends.
- Basic preprocessing includes **cleaning and scaling** data, while advanced preprocessing like **feature engineering** enhances model performance.

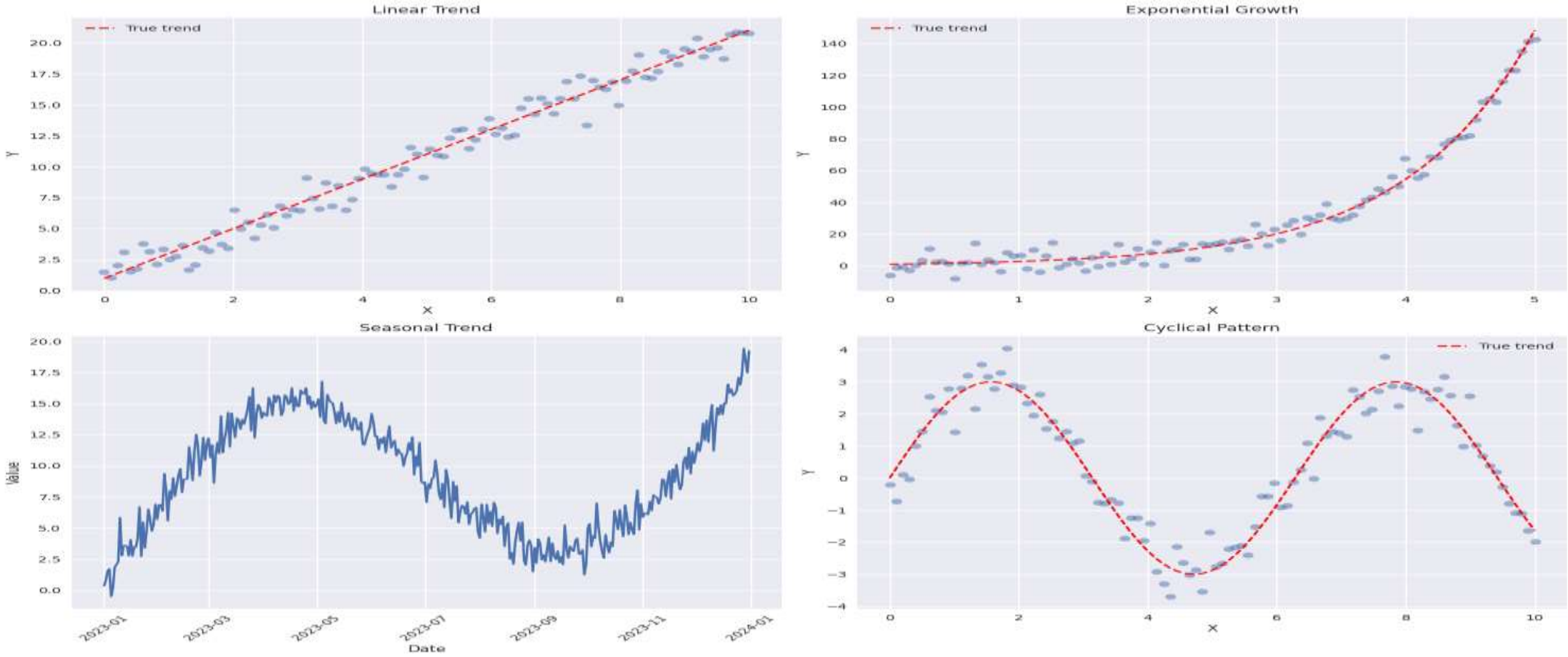


EXPLORATORY DATA ANALYSIS

- Exploratory Data Analysis (**EDA**) is a critical step in the data science process that helps you **understand** the **underlying patterns, trends, and structure** of the data before applying any advanced modeling techniques.
- It involves a variety of techniques to **summarize** and **visualize** the dataset's main characteristics, often using **statistical graphics** and other data **visualization methods**.

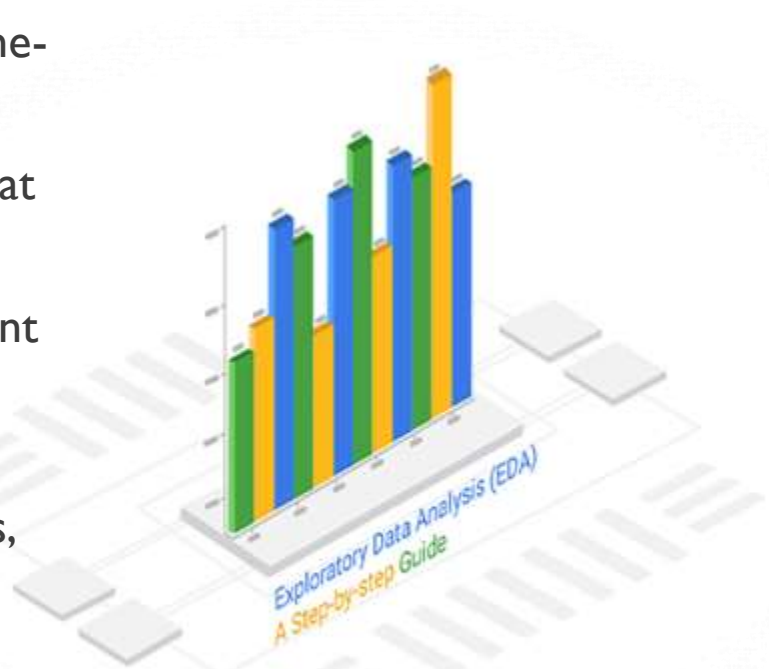


EXPLORATORY DATA ANALYSIS



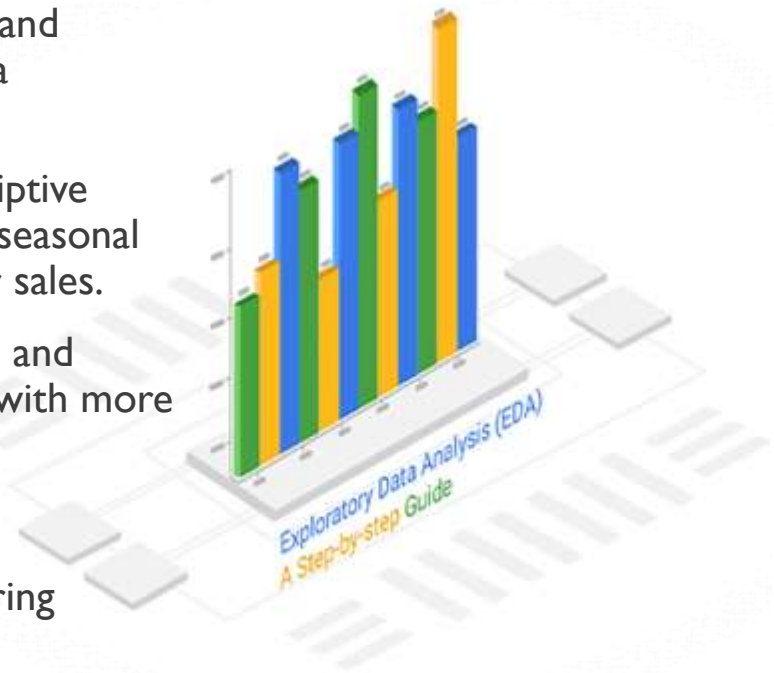
OBJECTIVES OF EDA

- **Data Understanding**: EDA helps you gain an initial understanding of the dataset, such as its size, the types of features (numerical, categorical, time-based), and how the data is distributed.
- **Data Cleaning:** Identify missing values, outliers, and inconsistencies that need to be addressed before moving on to modeling.
- **Feature Selection and Engineering:** EDA can help identify important variables or relationships that can be used to create new features.
- **Detecting Patterns and Relationships**: Visualizations like scatter plots, histograms, and heatmaps can reveal hidden patterns, correlations, or trends.
- **Hypothesis Testing**: EDA can be used to formulate hypotheses based on patterns observed in the data, which can be tested later with statistical models.

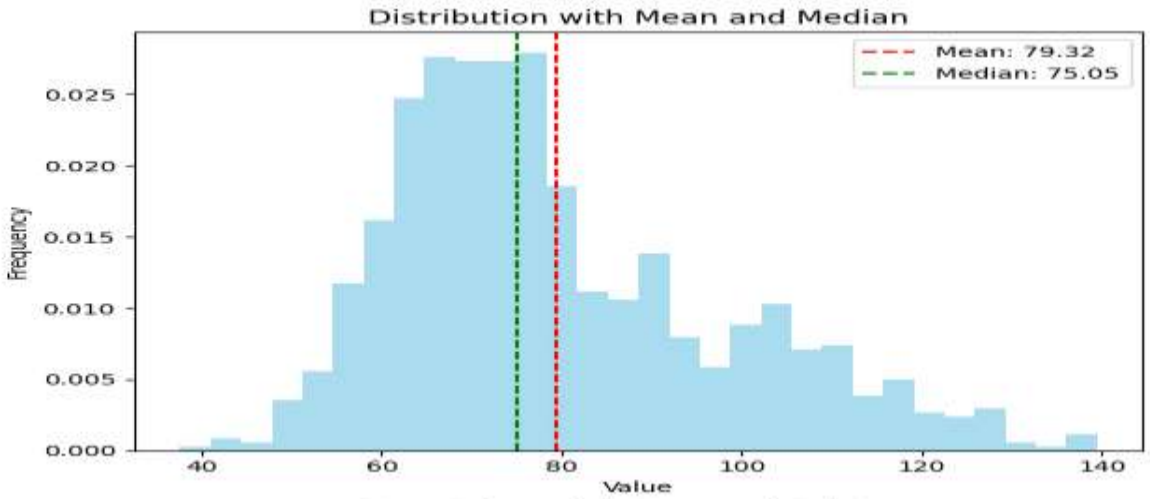


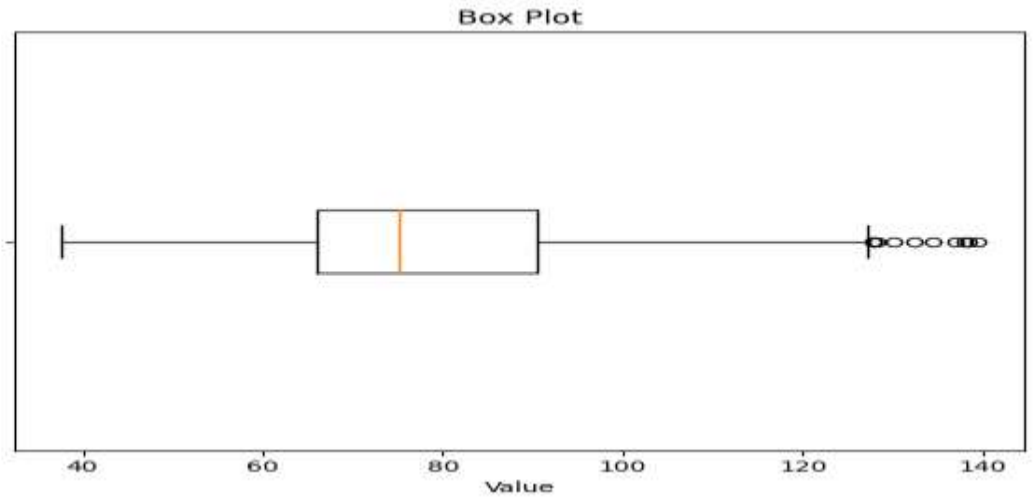
EDA TECHNIQUES

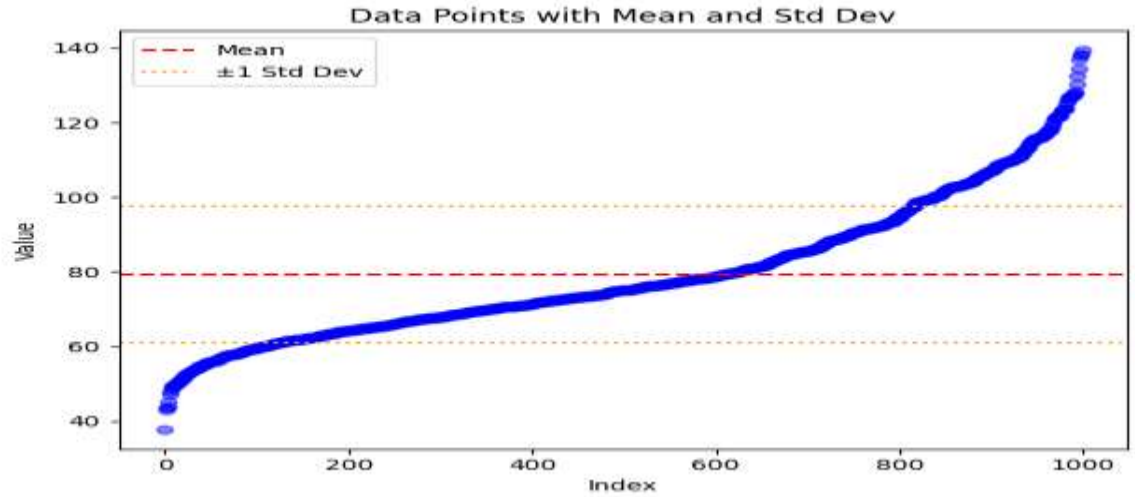
- **Descriptive statistics** play a vital role in data science projects by providing tools to summarize and understand data before formulating hypotheses.
  - **Summarizing Data**: Descriptive statistics like mean, median, mode, variance, and standard deviation help summarize large datasets, making it easier to grasp data distribution and variability.
  - **Formulating Hypotheses**: By understanding trends and patterns from descriptive statistics, we can propose initial hypotheses. For example, if sales data shows a seasonal increase during winter, we might hypothesize that colder weather drives higher sales.
  - **Hypothesis Confirmation**: Once a hypothesis is formed, inferential statistics and modeling are used in later stages to test and confirm these initial assumptions with more robust analysis, such as regression or hypothesis testing.
- Example:
  - **Initial Observation**: Descriptive stats reveal an increase in website traffic during weekends.
  - **Hypothesis**: Weekend promotions drive more traffic.
  - **Testing**: Conduct an A/B test during future weekends to validate this.



BASIC STATISTICAL MEASURES

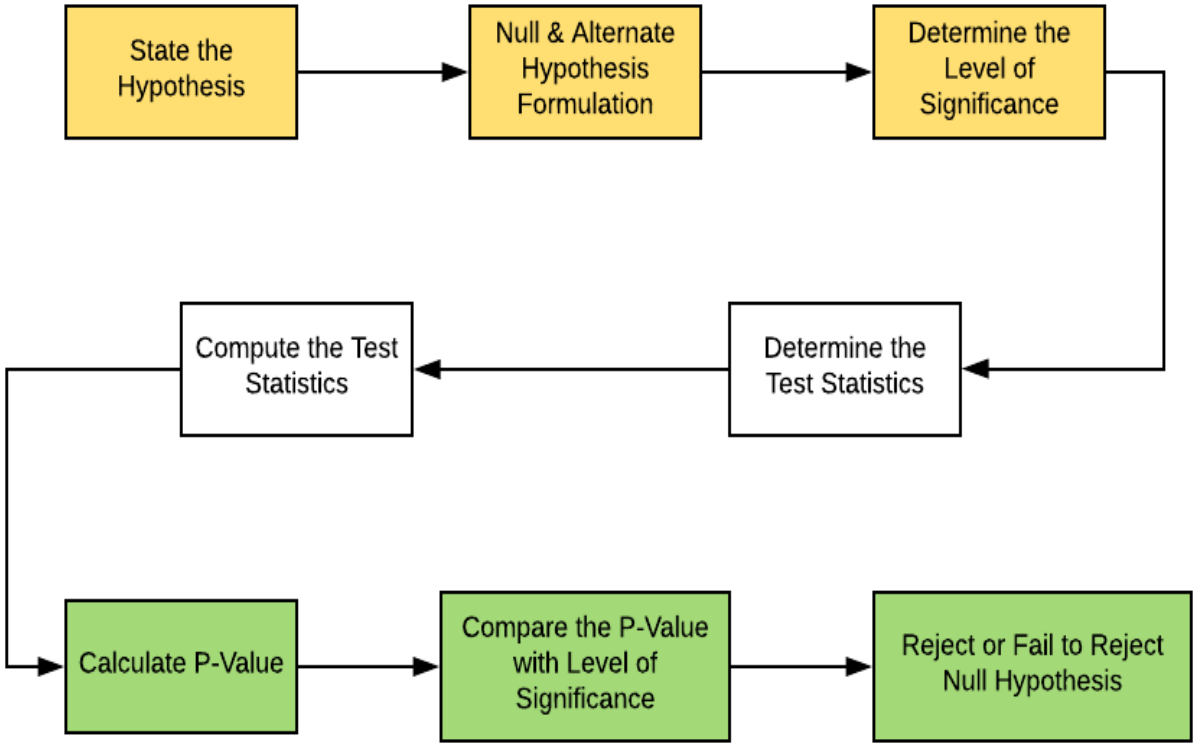






Mean: 79.32
Median: 75.05
Mode: 37.59
Variance: 336.42
Standard Deviation: 18.34

INTERROGATING THE DATA



Hypothesis Testing Workflow

Null hypothesis	Running 5 miles a day **does not** result in the reduction of 10 kg of weight within a month.
Alternate hypothesis	Running 5 miles a day results in the reduction of 10 kg of weight within a month.

DIFFERENT DATA DISTRIBUTIONS

1. Normal Distribution

For $x \in \mathbb{R}$, $\mu \in \mathbb{R}$, $\sigma > 0$:

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

2. Uniform Distribution

For $a < b$, $x \in [a, b]$:

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } x \in [a, b] \\ 0 & \text{otherwise} \end{cases}$$

3. Exponential Distribution

For $x \geq 0$, $\lambda > 0$:

$$f(x) = \lambda e^{-\lambda x}$$

4. Chi-Square Distribution

For $x > 0$, k degrees of freedom:

$$f(x) = \frac{1}{2^{k/2}\Gamma(k/2)} x^{k/2-1} e^{-x/2}$$

where $\Gamma(z)$ is the gamma function.

5. Beta Distribution

For $x \in [0, 1]$, $\alpha > 0$, $\beta > 0$:

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}$$

where $B(\alpha, \beta)$ is the beta function:

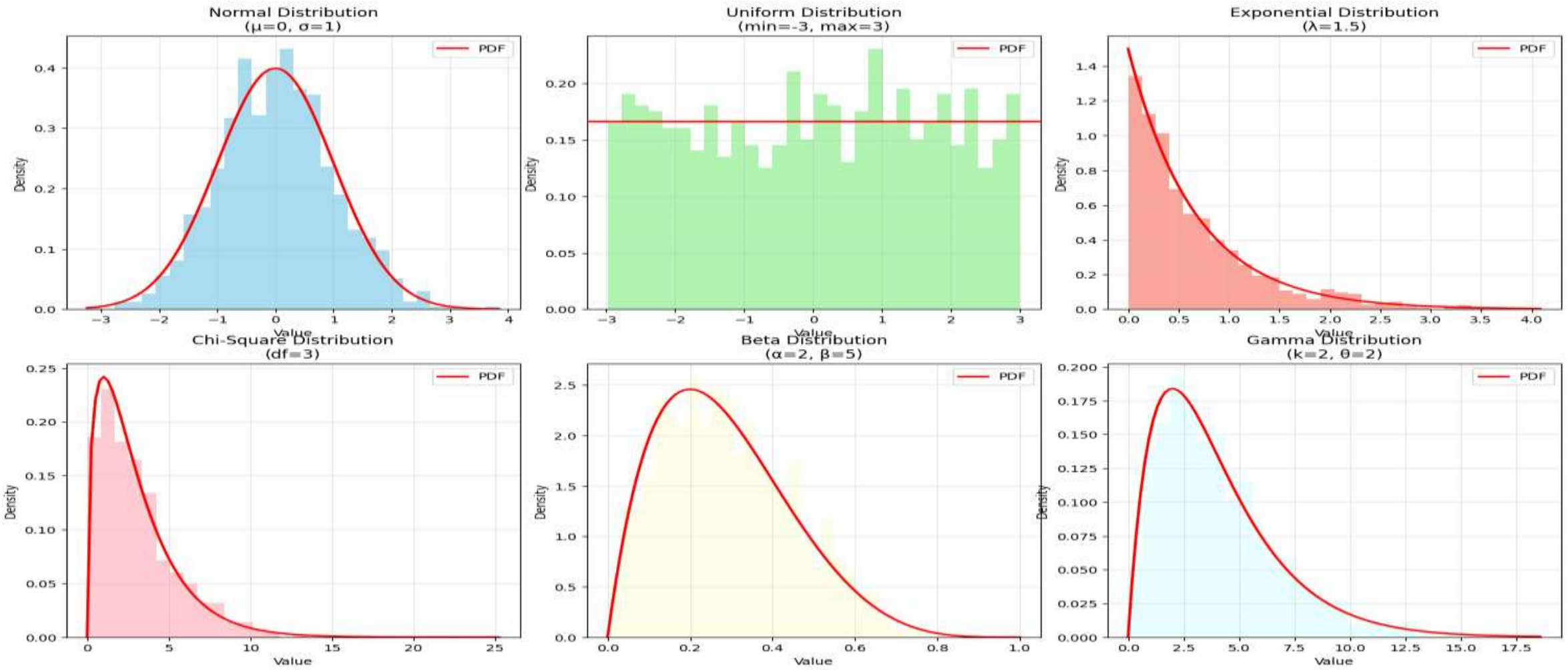
$$B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

6. Gamma Distribution

For $x > 0$, shape parameter $k > 0$, scale parameter $\theta > 0$:

$$f(x) = \frac{x^{k-1} e^{-x/\theta}}{\theta^k \Gamma(k)}$$

DIFFERENT DATA DISTRIBUTIONS



EDA TECHNIQUES: OUTLIERS

- Outliers are data points that differ significantly from other observations in a dataset. They can skew statistical analyses and lead to misleading results if not addressed properly.
- **Importance of Mitigating Outliers**
  - **Impact on Models**: Outliers can disproportionately affect the performance of predictive models, leading to reduced accuracy and reliability.
  - **Distortion of Results**: They can skew summary statistics (mean, standard deviation) and lead to erroneous interpretations.
  - **Insights and Anomalies**: In some cases, outliers may represent critical information or rare events that are valuable for decision-making.

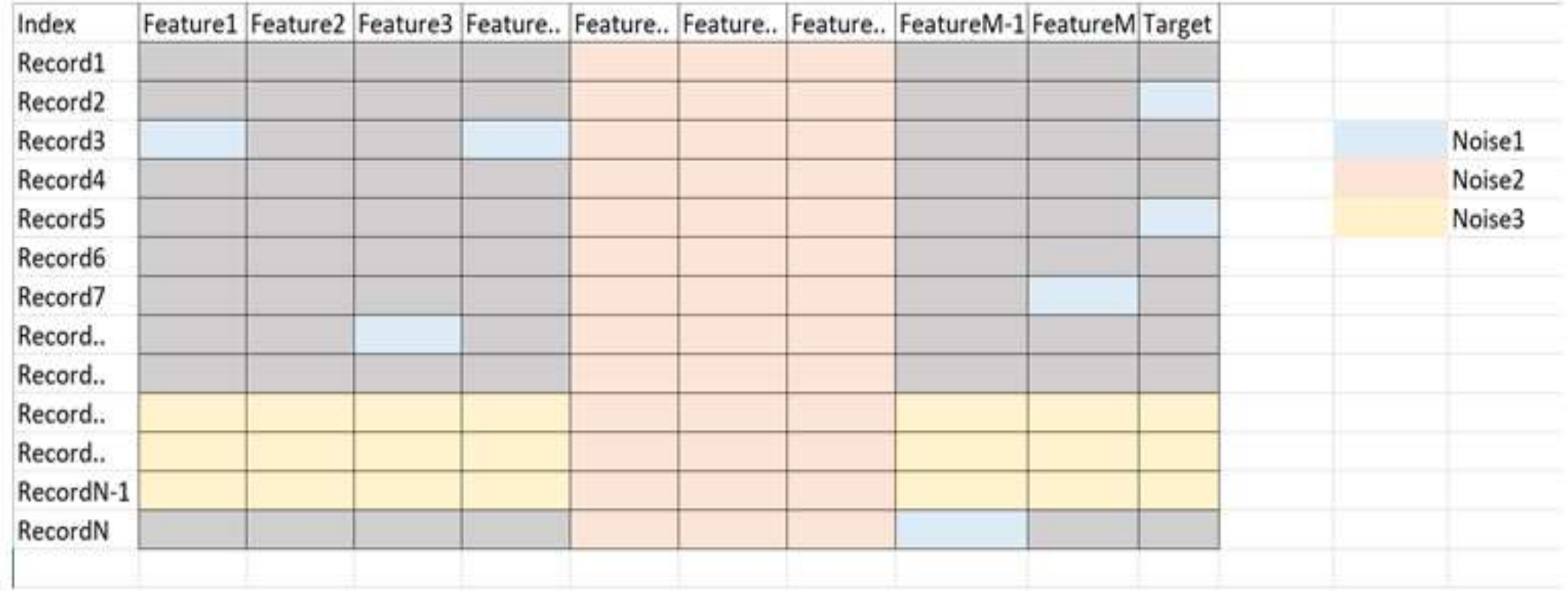
EDA TECHNIQUES: OUTLIERS

- **Types of Outliers**
  - **Outlier as an Instance**:
    - A specific observation that deviates from the expected pattern. For example, a transaction amount that is significantly higher than typical values in financial data.
  - **Outlier as a Feature**:
    - Certain features (variables) may exhibit outlier behavior, impacting the entire dataset. For instance, a feature indicating customer age that has extremely high values compared to the average age.
  - **General Outliers**:
    - Outliers can be defined as points beyond a certain threshold, often determined using statistical methods (e.g., Z-scores, IQR).

EDA TECHNIQUES: OUTLIERS

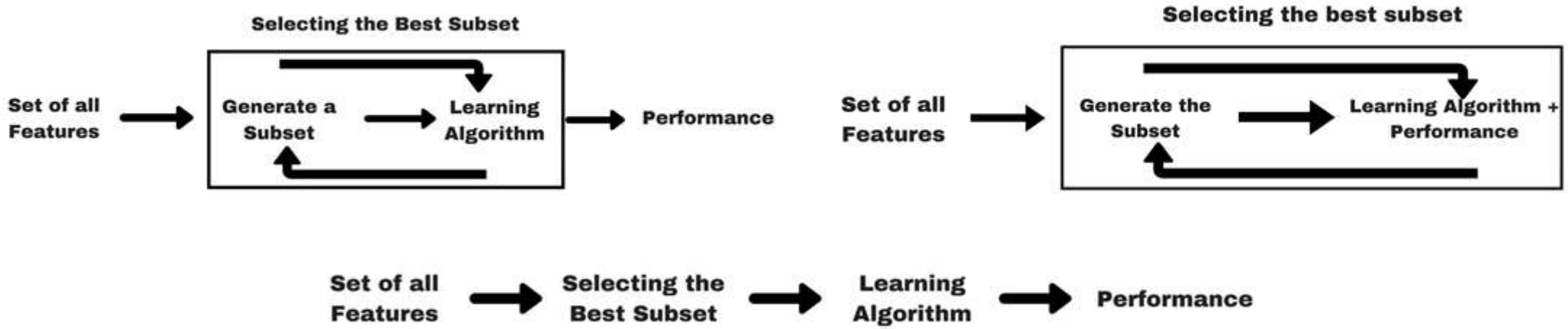
- **Detection Techniques:**
  - Statistical Methods: Z-score, Modified Z-score, Interquartile Range (IQR).
  - Visualization: Box plots, scatter plots, and histograms can help identify outliers visually.
- **Mitigation Strategies:**
  - Trimming: Removing outliers from the dataset.
  - Transformation: Applying mathematical transformations (e.g., log transformation) to reduce the impact of outliers.
  - Imputation: Replacing outliers with statistical estimates (mean, median) to reduce their influence.
  - Robust Models: Using algorithms less sensitive to outliers, such as tree-based models or robust regression techniques.

EDA TECHNIQUES: TYPES OF OUTLIERS



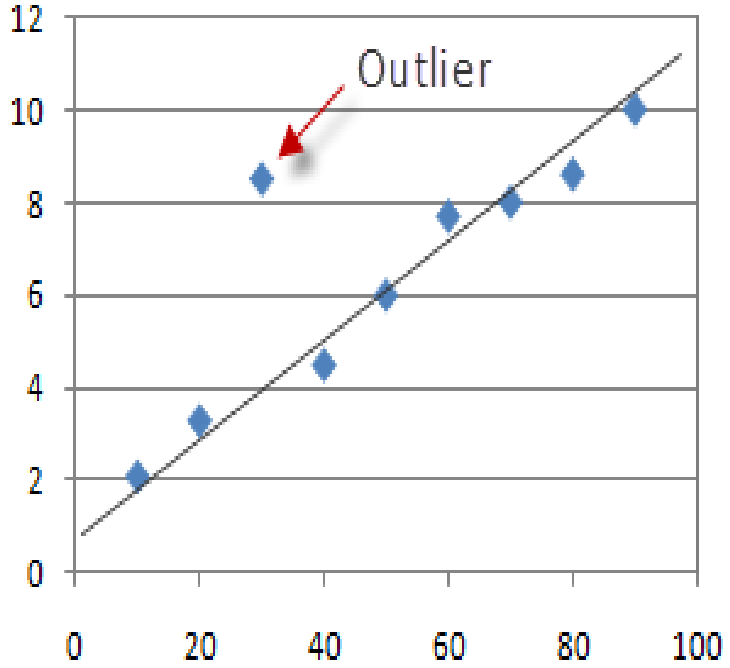
Types of data niose

EDA TECHNIQUES: TYPES OF OUTLIERS

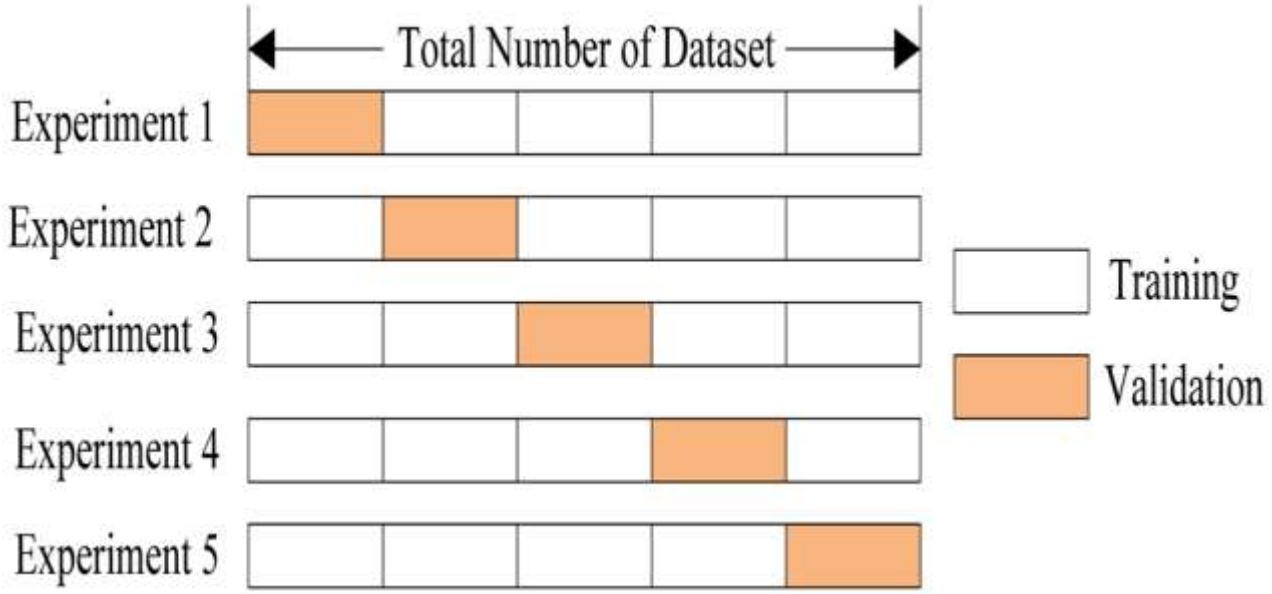


- Using greedy algorithms, heuristics, bio-inspired algorithms …etc.

EDA TECHNIQUES: TYPES OF OUTLIERS



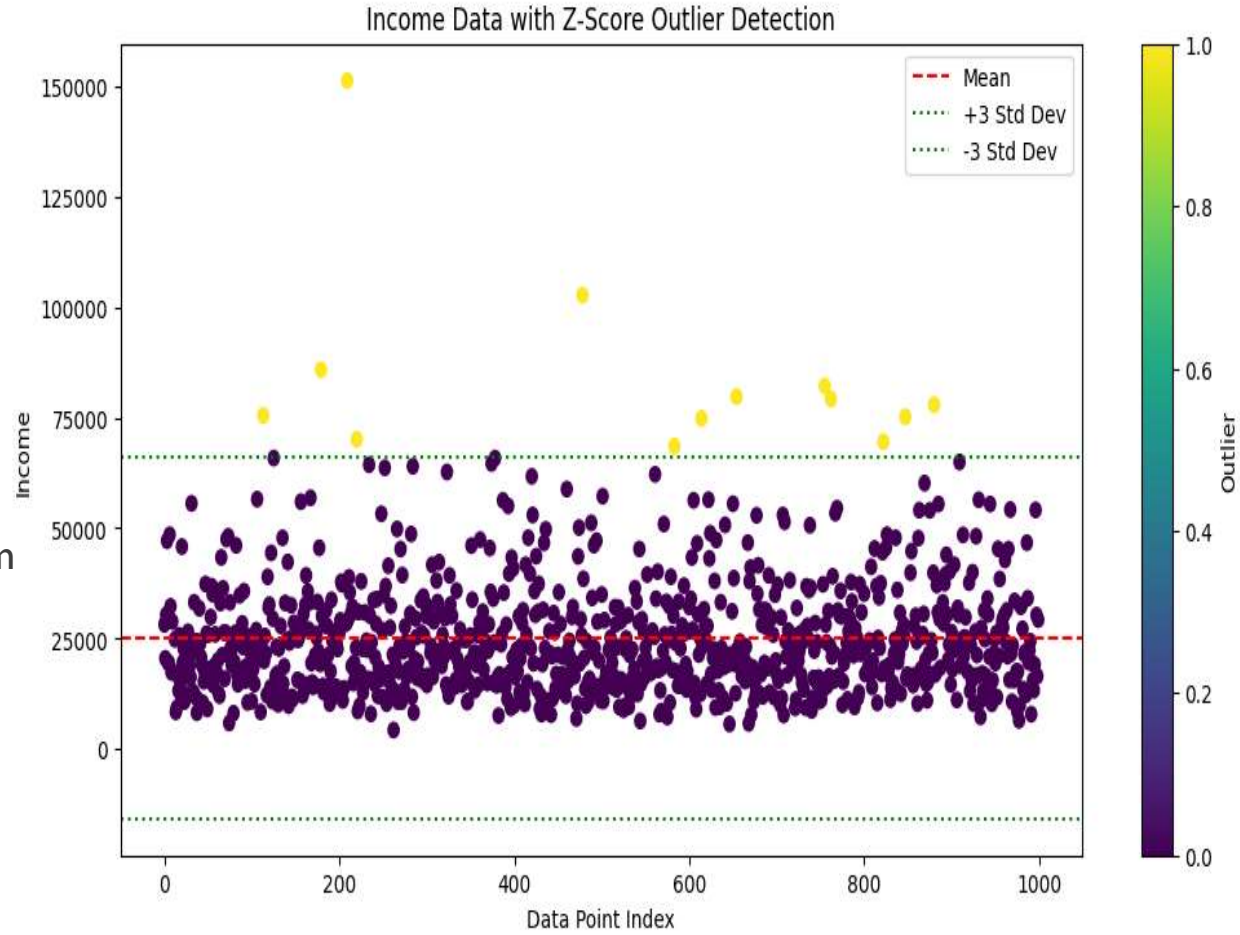
Noise as a record (one item)



Using resampling to mitigate data noise

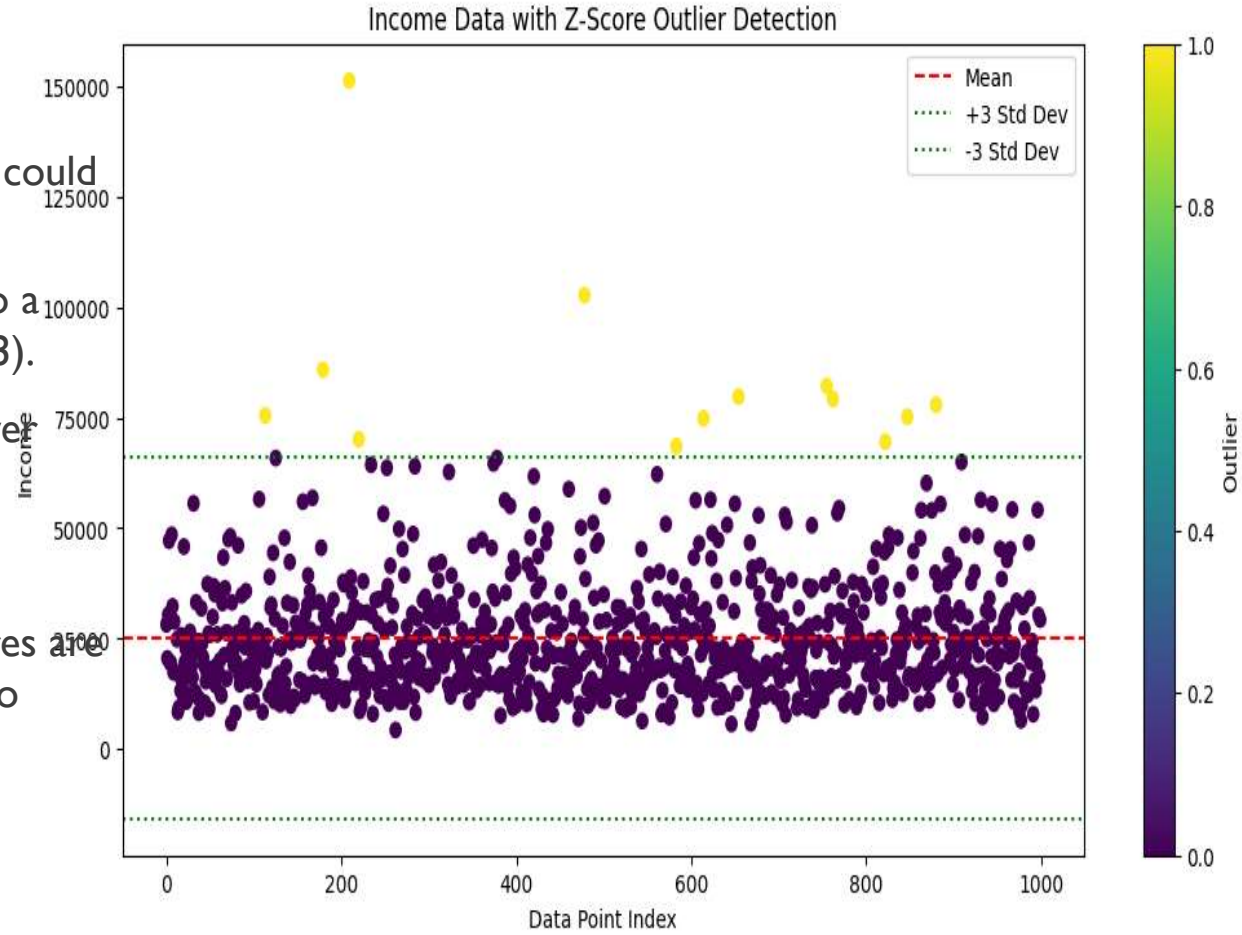
EDA TECHNIQUES: HANDLING OUTLIERS USING STATS

- **Z-Score (Standard Score) Method**
  - The **Z-score** method measures how far a data point is from the mean in terms of standard deviations.
- **How Z-Score Detects Outliers:**
  - A data point with a Z-score higher than a certain threshold (e.g., 3 or -3) is often considered an outlier.
  - This threshold means the data point lies 3 standard deviations away from the mean, indicating that it is significantly different from the rest of the data.
- **Steps:**
  - **Calculate** the Z-score for each data point.
  - **Flag** data points that have Z-scores beyond a chosen threshold (commonly |Z| > 3).



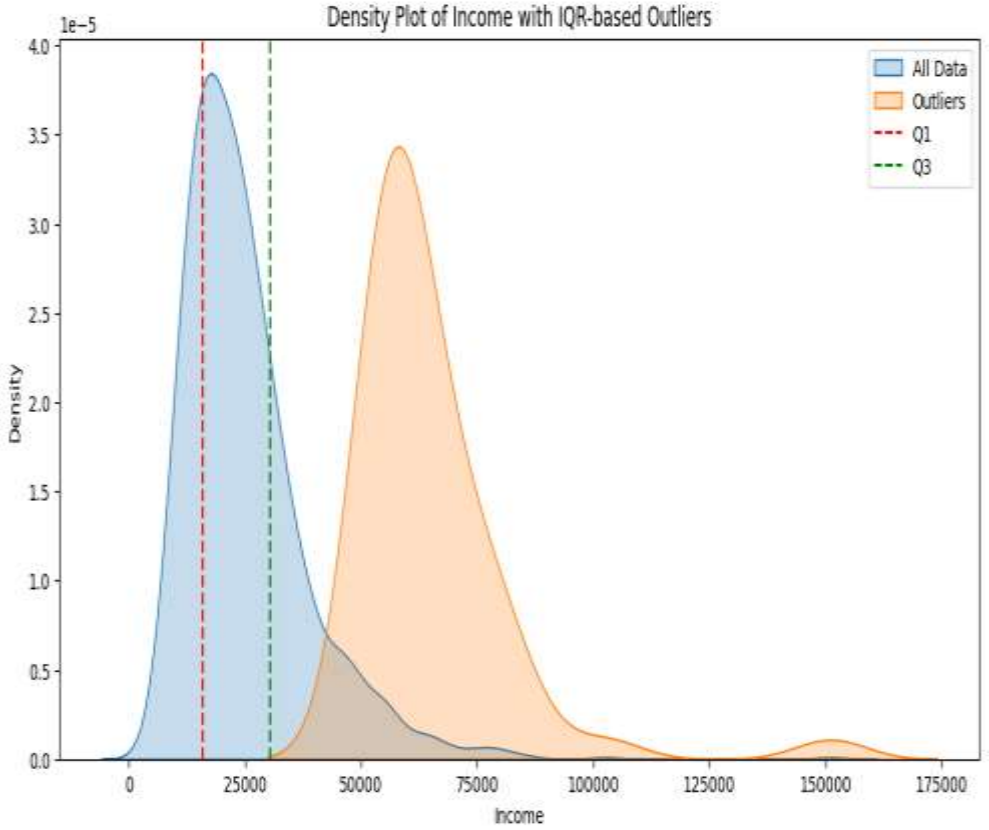
EDA TECHNIQUES: HANDLING OUTLIERS USING STATS

- **Handling Outliers using Z-Score:**
  - **Remove** the outliers if they are errors or if their presence could distort the analysis.
  - **Cap (Winsorize)** the outliers by setting extreme values to a predefined threshold (e.g., replacing Z-scores > 3 with Z = 3).
  - **Transform** the data using methods like logarithmic or power transformations to reduce the impact of outliers.
- **When to Use:**
  - Best suited for **normally distributed** data because Z-scores are based on mean and standard deviation, which are sensitive to skewed data and outliers themselves.



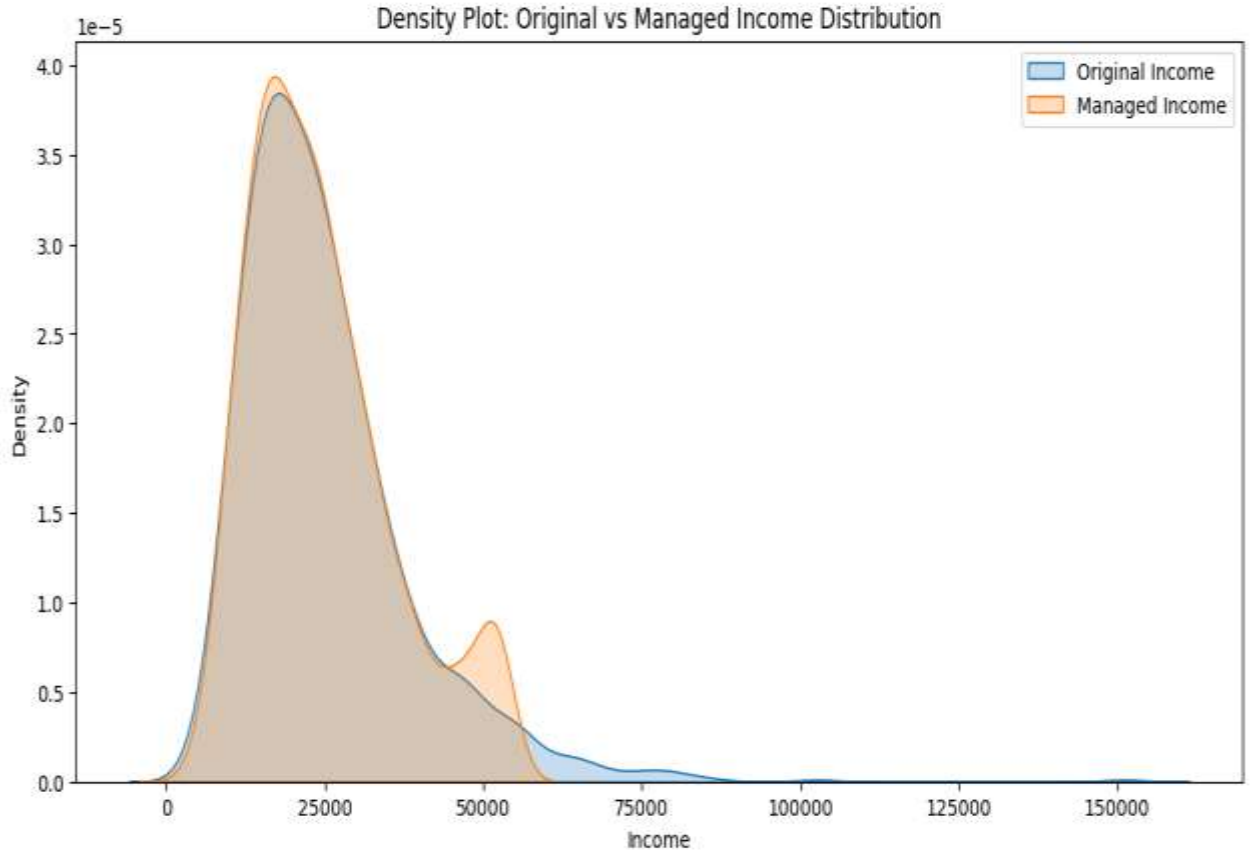
EDA TECHNIQUES: HANDLING OUTLIERS USING STATS

- **Interquartile Range (IQR) Method**
  - The **Interquartile Range (IQR)** method identifies outliers by looking at the spread of the middle 50% of the data. The IQR is the difference between the third quartile (Q3) and the first quartile (Q1):
    - IQR=Q3−Q1
- **How IQR Detects Outliers:**
  - Outliers are typically defined as data points that lie below Q1−1.5×IQR
  - or above Q3+1.5×IQR
  - These thresholds capture data points that fall far from the bulk of the data.
- **Steps:**
  - **Calculate** the first (Q1) and third quartiles (Q3).
  - **Compute** the IQR as IQR=Q3−Q1.
  - **Determine** the lower and upper bounds for potential outliers:
    - Lower Bound = Q1−1.5×IQRQ1
    - Upper Bound = Q3+1.5×IQR
    - **Flag** data points that fall outside these bounds as outliers.

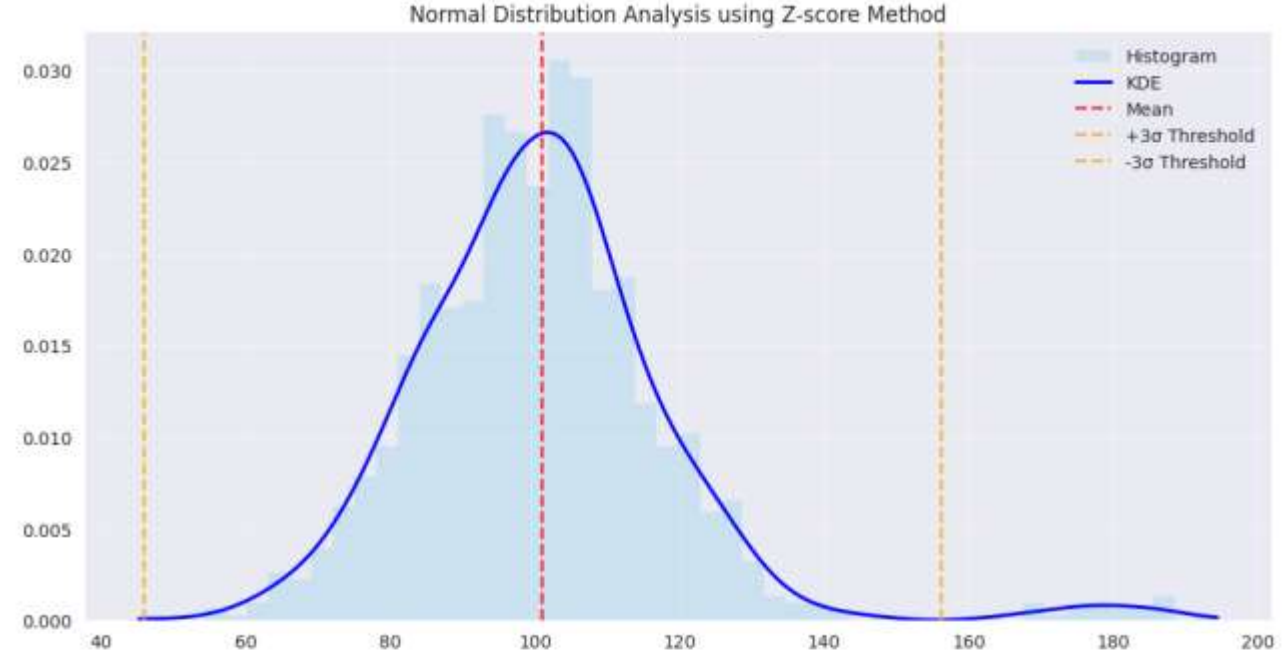


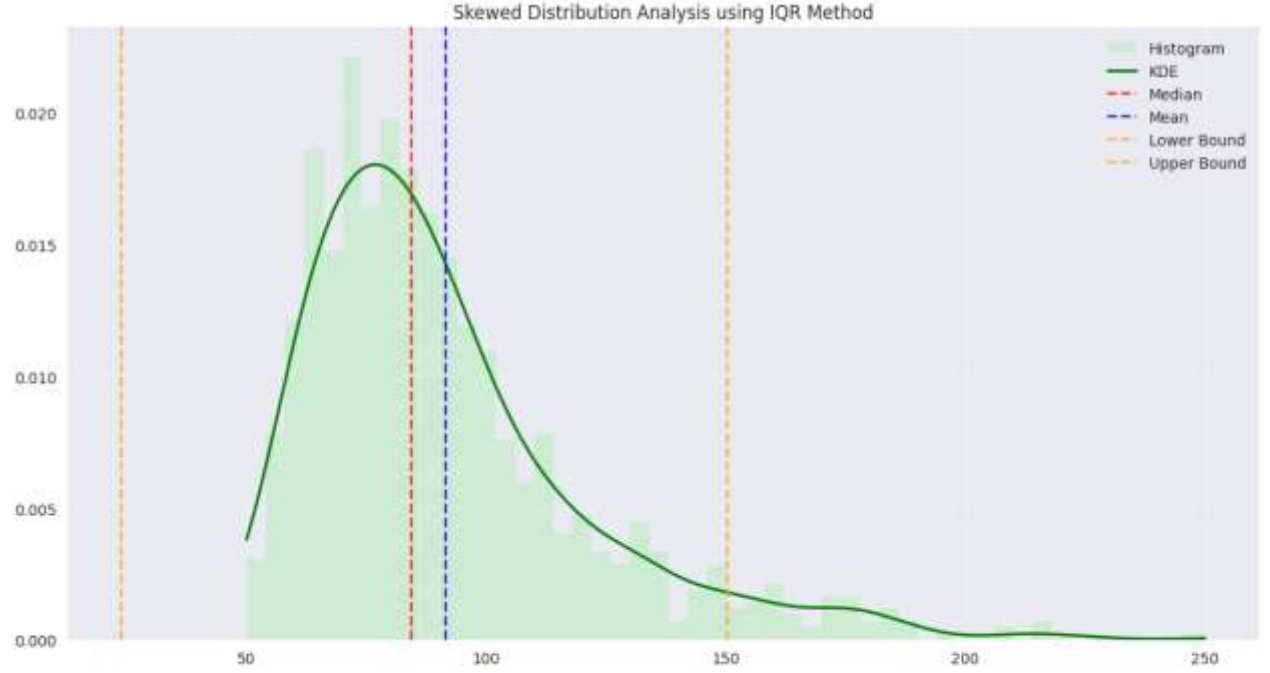
EDA TECHNIQUES: HANDLING OUTLIERS USING STATS

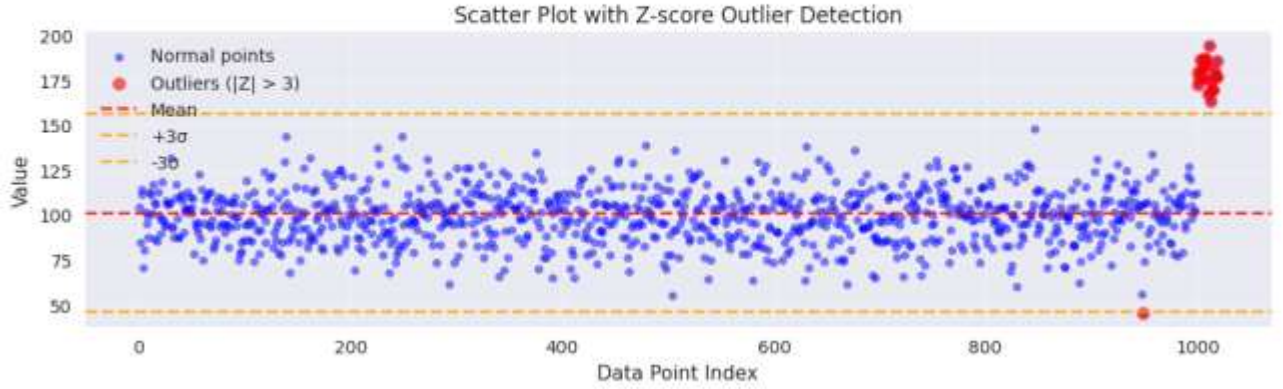
- **Handling Outliers using IQR:**
  - **Remove** the outliers, especially if they are data entry errors or unrepresentative of the population.
  - **Impute** the outliers by replacing them with the median or mean of the remaining data.
  - **Cap** extreme values at the upper and lower bounds to limit their impact.
- **When to Use:**
  - Suitable for **skewed** data and **non-normal distributions** because it is based on percentiles, which are less affected by extreme values compared to the mean and standard deviation.

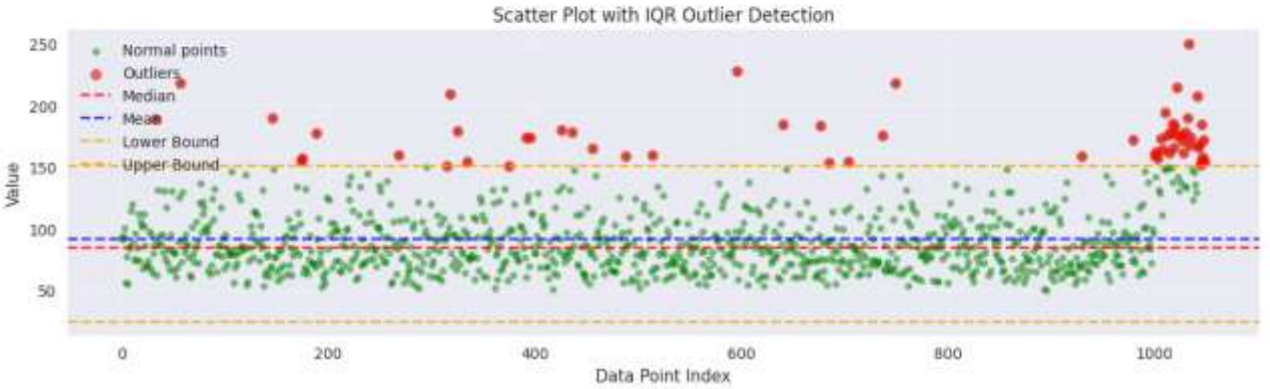


EDA TECHNIQUES: HANDLING OUTLIERS USING STATS









EDA TECHNIQUES: OUTLIERS

- **Logarithmic Transformation**
  - A **logarithmic transformation** is often used to reduce the effect of outliers, especially when the data is heavily skewed. This technique works by compressing large values more than small values, making the data more symmetric.
- **Example:**
  - Transforming a skewed distribution:
  - Original data: [10, 100, 1000, 10000]
  - Log-transformed data: [1, 2, 3, 4]
- **Handling Outliers:**
  - Log transformation can **reduce the impact** of extreme values by shrinking the range of the data.

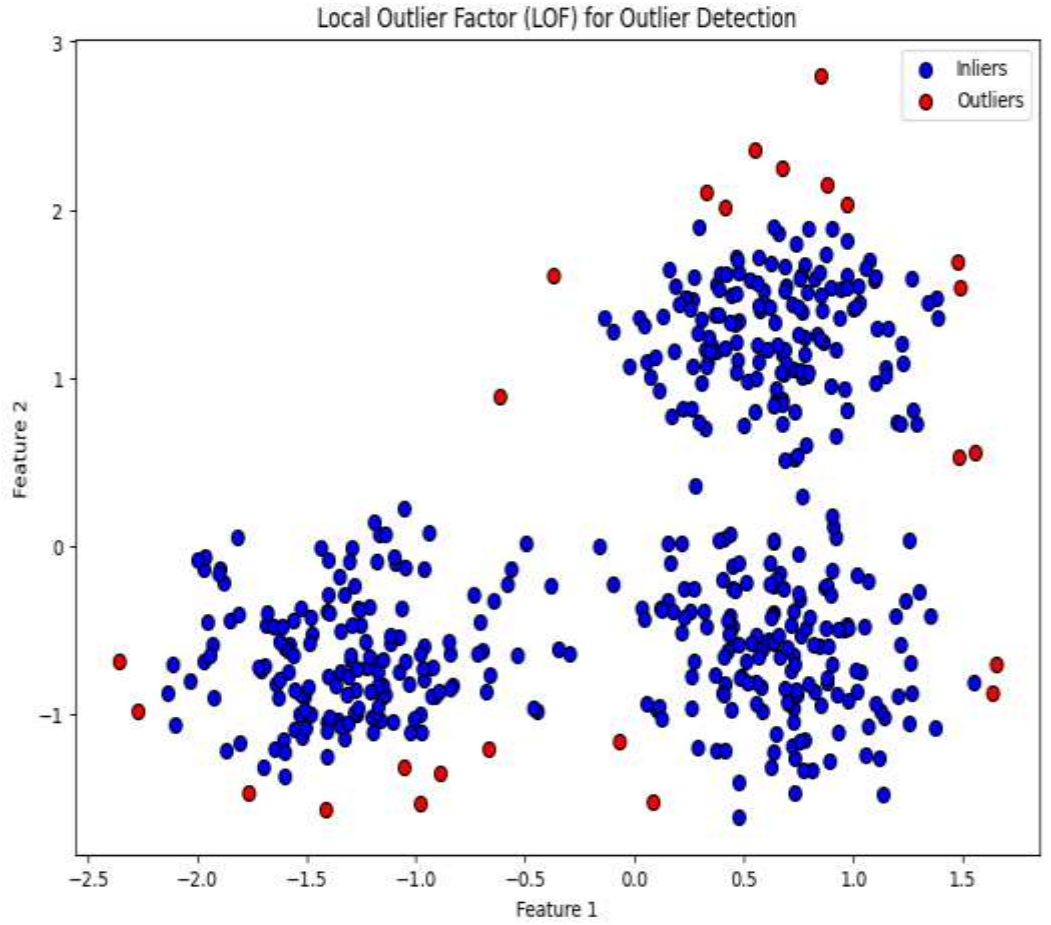
- **Winsorization**
  - Winsorization is a transformation technique where extreme values are replaced with the nearest non-outlier value. It is similar to capping, but instead of setting thresholds based on percentiles, it caps the outliers to specific values.
- **Handling Outliers:**
  - For extreme data points, you can set all values beyond a certain percentile (e.g., the 95th percentile) to the value at the 95th percentile.

EDA TECHNIQUES: HANDLING OUTLIERS USING DISTANCE BASED TECHNIQUES

- **KNN**: Detects outliers by looking at the distance to nearby points.
- **LOF**: Compares local densities of points to their neighbors to find outliers.
- **DBSCAN**: Identifies outliers as noise points that do not belong to any cluster.
- **Mahalanobis Distance**: Detects multivariate outliers based on the distance from the mean and the covariance structure.
- **Distance to Centroid**: Detects outliers based on the distance from the centroid of the data.
- **K-Means Clustering**: Identifies outliers as points farthest from their assigned cluster centroid.
- **One-Class SVM:** Learns a boundary around normal points and flags those outside as outliers.

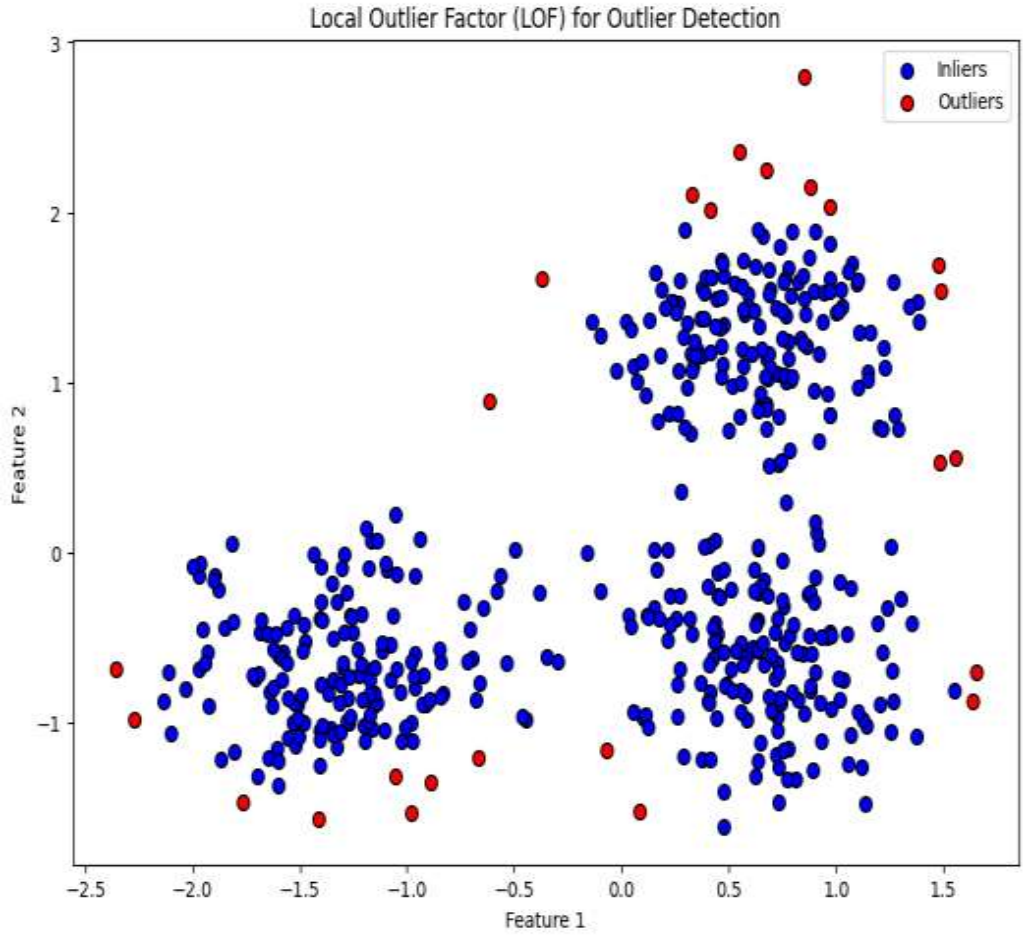
EDA TECHNIQUES: HANDLING OUTLIERS USING DISTANCE BASED TECHNIQUES

- **K-Nearest Neighbors (KNN)** for Outlier Detection, the K-Nearest Neighbors (KNN) algorithm can be adapted for outlier detection by considering the distance of a point to its nearest neighbors.
- **How It Works:**
  - For each data point, calculate the distance to its k nearest neighbors.
  - If the average distance between a point and its nearest neighbors is large compared to other points, that point is considered an outlier.
- **Steps**:
  - For a data point XX, find its k nearest neighbors based on a chosen distance metric (e.g., Euclidean or Manhattan distance).
  - Calculate the average distance between XX and its neighbors.
  - Define a threshold for classifying a point as an outlier based on the distribution of distances (e.g., points with the highest 5% of distances could be flagged as outliers).
- **Handling Outliers:**
  - Remove data points that are flagged as outliers.
  - Impute outliers by replacing them with the average or median value of their nearest neighbors.
- **When to Use:**
  - Works well when the dataset has a natural notion of proximity, such as spatial or time-series data.
  - Can handle multivariate outliers by considering the relationships between multiple features.



EDA TECHNIQUES: HANDLING OUTLIERS USING DISTANCE BASED TECHNIQUES

- **Local Outlier Factor (LOF)**
  - The **Local Outlier Factor (LOF)** algorithm is a more sophisticated version of KNN-based outlier detection. It compares the **local density** of a data point with that of its neighbors. A point is considered an outlier if its local density is much lower than that of its neighbors.
- **How It Works:**
  - LOF measures how isolated a point is relative to its surroundings. This is done by comparing the **local density** of a point to the local densities of its neighbors.
  - Points with a **significantly lower density** than their neighbors are flagged as outliers.
- **Steps:**
  1. For each point XXX, calculate its **local density** based on the distances to its neighbors.
  2. Compute the ratio of the density of XXX to the average density of its neighbors.
  3. A high ratio indicates that XXX is less dense (isolated) than its neighbors, meaning it is an outlier.
- **Handling Outliers:**
  - LOF assigns an outlier score to each data point. Points with scores above a certain threshold can be **removed** or **flagged** for further inspection.
- **When to Use:**
  - Useful when data points are clustered, and you want to detect **local** outliers within groups rather than global outliers.
  - Works well with **multivariate data**.



EDA TECHNIQUES: HANDLING OUTLIERS USING DISTANCE BASED TECHNIQUES

Feature	KNN	LOF
Type	Classification algorithm adapted for outlier detection	Density-based outlier detection algorithm
Detection Principle	Distance-based comparison to k neighbors	Local density comparison with neighbors
Sensitivity to Density	Less sensitive; treats all neighborhoods uniformly	More sensitive; adapts to local density variations
Performance	May struggle with varying densities	More robust in datasets with varying densities
Computational Complexity	High due to distance calculation to all points	High, but better suited for complex density structures
Interpretability	Simpler interpretation based on distance	Provides a relative score for outlierness based on local density

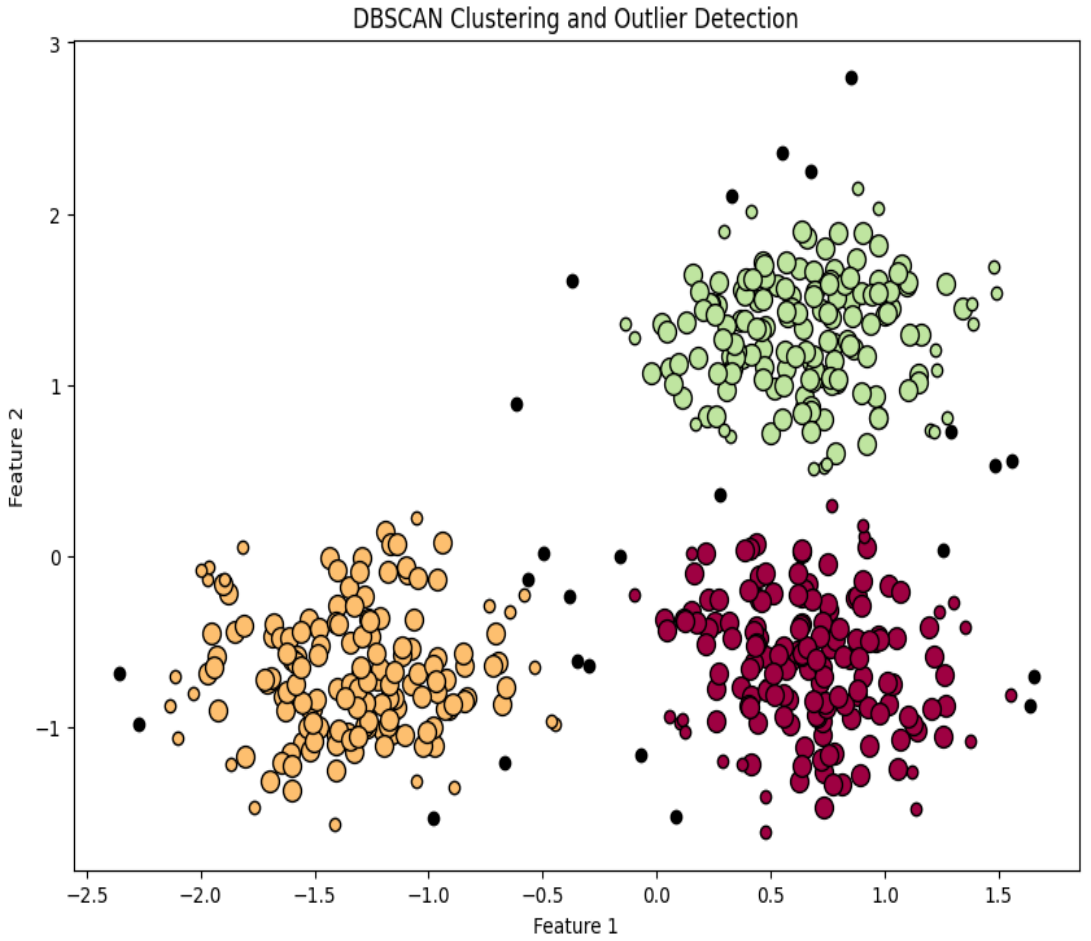
EDA TECHNIQUES: HANDLING OUTLIERS USING DISTANCE BASED TECHNIQUES

- Outlier Score Calculation:
  - **KNN**: The outlier score is based on a simple distance-based approach. It calculates the distance to the k nearest neighbors and can classify points based on this distance.
  - **LOF**: LOF takes a more sophisticated approach by considering the local reachability density of a point relative to its neighbors. It computes an outlier factor that compares the density of a point to the density of its neighbors, which allows it to effectively account for varying densities in the data.
- Local Density:
  - **KNN**: While it may calculate local density, it does not normalize or account for variations in density among different neighborhoods. The outlier detection can be less effective in datasets with clusters of varying density.
  - **LOF**: Specifically designed to account for local density variations. It uses the concept of Local Reachability Density (LRD) and calculates an outlier factor based on this density. This makes LOF more robust in detecting outliers in datasets where the distribution is not uniform.

- Neighborhood-Based:
  - Both methods rely on the concept of neighborhoods defined by distance to determine how far a data point is from others.
- Use of Distance Metrics:
  - Both methods commonly use distance metrics (like Euclidean distance) to measure how close points are to each other.
- K Nearest Neighbors:
  - Both use the idea of "k nearest neighbors" to evaluate points in the context of their local neighborhood.

EDA TECHNIQUES: HANDLING OUTLIERS USING CLUSTERING

- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**
  - **DBSCAN** is primarily a clustering algorithm, but it can also be used to detect outliers as it inherently identifies points that do not belong to any cluster.
- **How It Works:**
  - DBSCAN groups points that are close together into clusters based on a **distance threshold**.
  - Points that are not part of any cluster (because they are too far from any other points) are classified as **noise** or outliers.
- **Steps:**
  1. Set a **distance threshold (epsilon)** and a **minimum number of points (minPts)** required to form a dense region (cluster).
  2. DBSCAN will create clusters of points that are within the epsilon distance from each other.
  3. Any point that does not belong to a cluster is classified as an **outlier**.
- **Handling Outliers:**
  - **Remove** points labeled as noise by DBSCAN.
  - Optionally, investigate these outliers further to determine if they are errors or legitimate, rare cases.
- **When to Use:**
  - Effective for detecting **global outliers** that do not fit into any cluster.
  - Works well for **non-globular** clusters and datasets where you expect some points to naturally be considered noise (e.g., geographical or spatio-temporal data).



EDA TECHNIQUES: HANDLING OUTLIERS USING CLUSTERING

- **K-Means**
  - K-Means is a centroid-based clustering method, which assigns each data point to the nearest centroid. However, it is sensitive to outliers because it minimizes the squared distance between points and centroids, making outliers pull the centroids toward them.
- **Handling Outliers with K-Means:**
  - **Limitation**: K-Means does not inherently detect outliers. Outliers can distort the centroids and affect clustering quality.
- **Common Strategies to Handle Outliers in K-Means:**
  - **Preprocessing (Outlier Removal)**: Before applying K-Means, outliers can be detected and removed using other methods such as Z-scores, LOF, or distance-based approaches.
  - **Robust K-Means Variants**:
    - **K-Medoids (PAM)**: Instead of centroids, K-Medoids selects actual data points as cluster centers, making it less sensitive to outliers.
    - **Trimmed K-Means**: A modified version where a certain proportion of the most distant points from the centroids is trimmed (considered as outliers) during the fitting process.

EDA TECHNIQUES: HANDLING OUTLIERS USING CLUSTERING

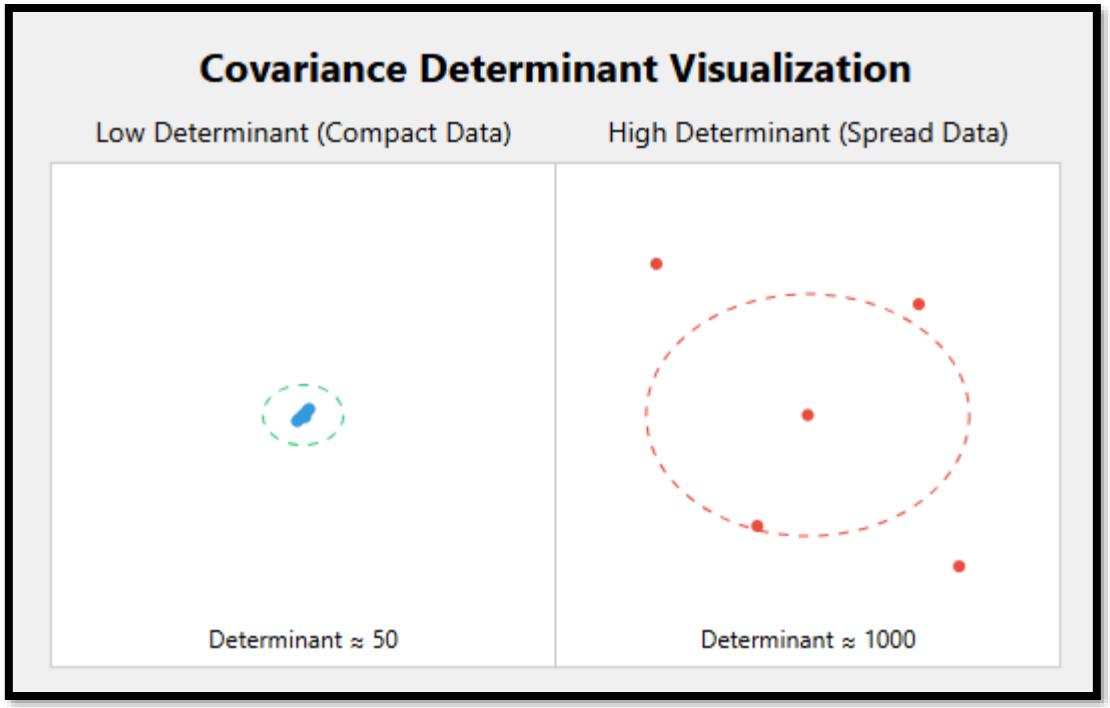
Clustering Method	Outlier Handling Capability	Technique
K-Means	Sensitive to outliers, needs preprocessing	Preprocess data (e.g., LOF, Z-score), or use robust variants like K-Medoids
DBSCAN	Naturally handles outliers	Identifies points in low-density regions as noise
Gaussian Mixture Models	Can detect outliers with low probability	Use a likelihood threshold to flag outliers
Agglomerative Clustering	Can handle outliers with distance thresholds	Set a distance threshold, points far from clusters are outliers

EDA TECHNIQUES: HANDLING OUTLIERS USING ADVANCED STATS

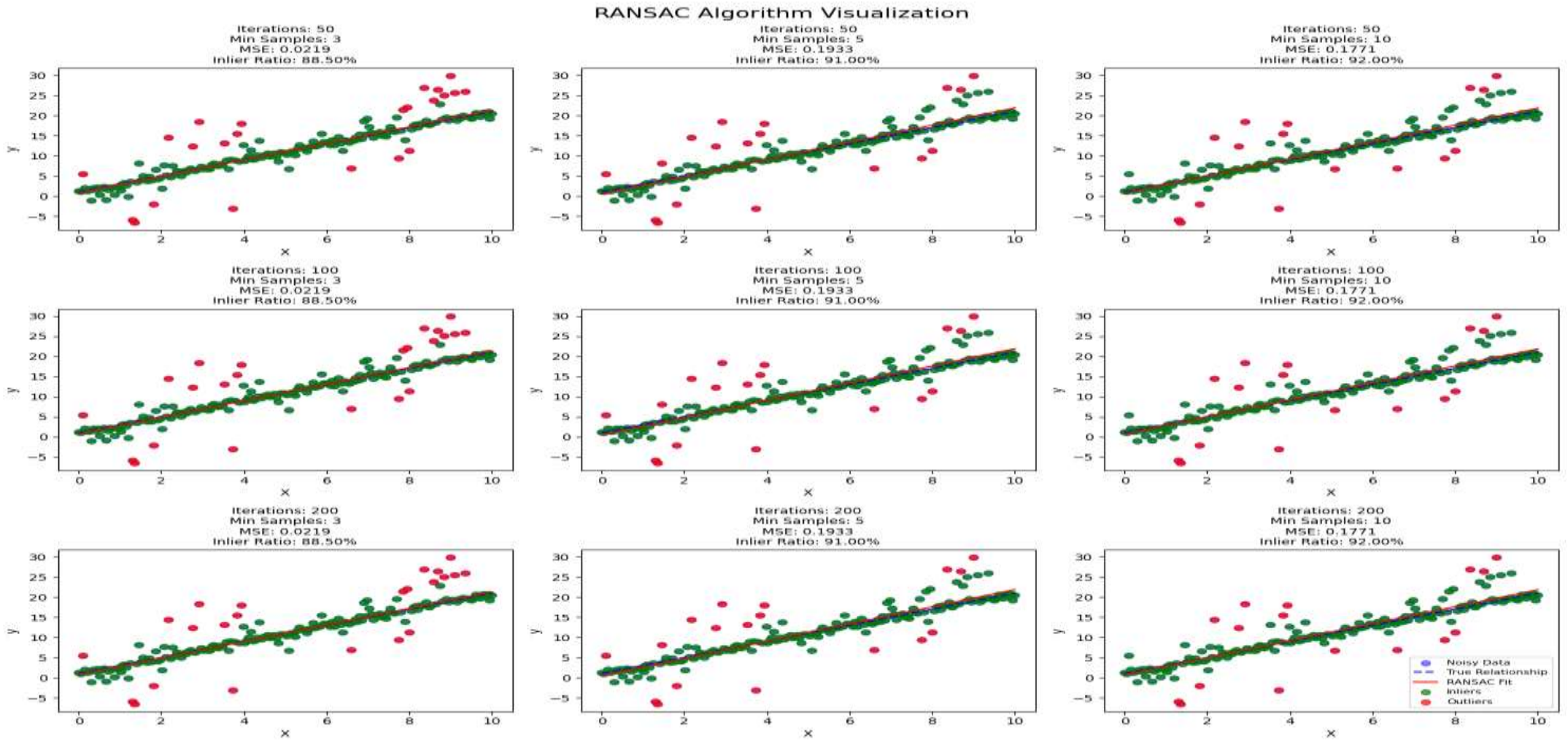
- **Robust Statistical Methods**
  - Robust statistical methods are designed to be less sensitive to outliers by using estimators that are not overly influenced by extreme values.
  - These techniques work well when the data contains both noise and significant outliers.
  - **A) Robust Regression (RANSAC)**
    - **RANSAC (Random Sample Consensus)** is an iterative algorithm that fits a model to the data while ignoring outliers.
    - It **randomly selects a subset of the data**, fits a model, and then tests how well the model fits the rest of the data.
    - The model that fits the largest number of points is selected, and the points not fitting the model are treated as outliers.
  - **B) Robust Covariance Estimation (Minimum Covariance Determinant)**
    - **Minimum Covariance Determinant (MCD)** is a robust estimator of the covariance matrix, used in multivariate data to detect outliers.
    - It computes the covariance matrix from a subset of the data that has the smallest **determinant**, minimizing the effect of outliers.
    - **Use Case:** Multivariate outlier detection in datasets with correlations between variables.

EDA TECHNIQUES: HANDLING OUTLIERS USING ADVANCED STATS

- The determinant in covariance estimation is a scalar value that provides crucial information about the spread and characteristics of a multivariate dataset. Here are the key insights:
- **Mathematical Definition**:
  - For a covariance matrix, the determinant represents the "volume" of the data's spread in multi-dimensional space
  - It measures how much the data is dispersed or concentrated
  - Calculated **as the product of the eigenvalues of the covariance matrix**
- **Role in MCD Algorithm**: MCD seeks to minimize the determinant of the covariance matrix
  - Finds the subset **of h observations** with the smallest covariance determinant
  - This approach helps identify the most consistent subset of data



EDA TECHNIQUES: HANDLING OUTLIERS USING ADVANCED STATS

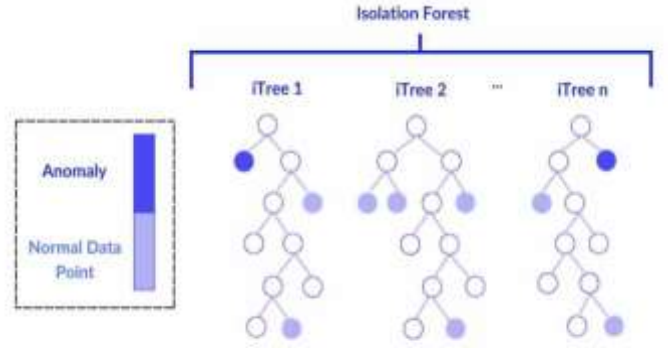


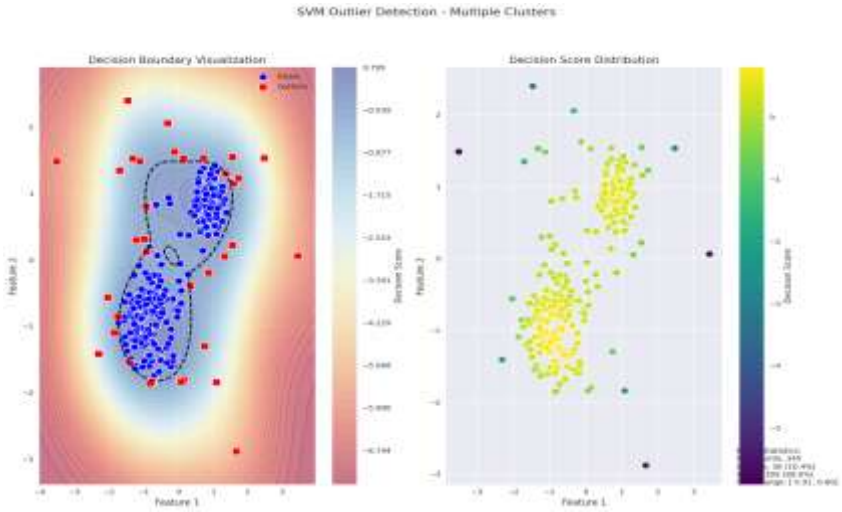
EDA TECHNIQUES: HANDLING OUTLIERS USING ADVANCED STATS

- **C) Elliptic Envelope**
  - **Elliptic Envelope** is a multivariate statistical method that assumes the data is distributed in a Gaussian distribution (elliptical shape).
  - It estimates the parameters of the Gaussian distribution and then identifies points that deviate significantly from the elliptical boundary as outliers.
  - **Use Case:** Multivariate outlier detection, useful in financial fraud detection, anomaly detection in sensor networks.
- **D) Principal Component Analysis (PCA) for Outlier Detection**
  - PCA is a dimensionality reduction technique that can also be used to detect outliers. It transforms the data into a lower-dimensional space, and points that do not conform to the dominant patterns of variability are likely outliers.
  - **How It Works:**
    - PCA identifies directions of maximum variance in the data (principal components). Points that lie far from the principal components are considered outliers.
  - **Use Case:** High-dimensional data, such as in image analysis or bioinformatics

EDA TECHNIQUES: HANDLING OUTLIERS USING ADVANCED STATS

- **E) Isolation Forest**
  - **Isolation Forest** is an ensemble-based anomaly detection method that is particularly well-suited for detecting outliers in high-dimensional data.
    - It isolates points by randomly partitioning the data.
    - Points that are easily isolated (i.e., require fewer partitions) are considered outliers.
  - **How It Works:**
    - It builds multiple trees (isolation trees) by randomly splitting features.
    - **Outliers are data points that are isolated quickly (short paths in the trees).**
  - **Use Case:** High-dimensional data, time series data, fraud detection, network intrusion detection.
- **F) One-Class SVM**
  - **One-Class Support Vector Machine (SVM)** is an unsupervised method that learns the boundary of normal points and identifies points that fall outside this boundary as outliers.
  - **How It Works:**
    - It attempts to separate the data from the origin using a hyperplane, and points lying outside the hyperplane boundary are classified as outliers.
  - **Use Case:** Outlier detection in high-dimensional data, anomaly detection in time series.





EDA TECHNIQUES: HANDLING OUTLIERS USING ADVANCED STATS

- **How One-Class SVM Works**
  - **Objective**: The goal of One-Class SVM is to find a boundary in the feature space that encapsulates most of the training data points, typically close to the origin (in feature space transformed by a kernel function).
  - **Boundary Creation**: By using the kernel trick, the One-Class SVM maps the data points into a higher-dimensional space where it becomes easier to separate the "normal" points from the origin. The algorithm tries to find the maximum-margin boundary (a hyperplane or other kernel-based structure) that best encloses the majority of the data points.
  - **Decision Function**: After training, the model uses this boundary to decide whether a new data point belongs to the same distribution as the training data or is an outlier. If the new point falls outside this boundary, it is considered an anomaly or outlier.

- Key Parameters in One-Class SVM
  - **Kernel**:
    - Determines the shape of the decision boundary. Common choices are linear, rbf (Radial Basis Function), and poly (polynomial). The rbf kernel is commonly used for outlier detection because it can capture non-linear boundaries.
    - **Nu (ν)**: This parameter controls the trade-off between false positives (normal points classified as outliers) and false negatives (outliers classified as normal). Higher nu values allow more points to be classified as outliers.
    - **Gamma**: Controls the influence of individual data points in the rbf kernel. Larger values make the boundary tighter around the training data points.

EDA TECHNIQUES: HANDLING OUTLIERS USING ADVANCED STATS

- # 1. INPUT DATA
- Input:
  - Data set `X = {x1, x2, ..., xn}` with points in d-dimensional space
  - Parameters:
    - gamma (for RBF kernel),
    - ν (fraction of points allowed as outliers)
- # 2. INITIALIZATION
  - Initialize:
    - gamma ← kernel parameter for RBF, which determines the influence of each point
    - ν ← fraction of outliers to allow (controls boundary flexibility)
    - alphas ← Initialize all alpha values to zero (one per data point)

- # 3. TRAINING
- For each data point xi in X:
  - - Compute the Kernel matrix `K(xi, xj)` using the RBF kernel:
    - **K(xi, xj) = exp(-gamma * ||xi - xj||^2)**
  - - Set up and solve the quadratic optimization problem:
    - Objective: **Maximize (sum of alphas) - (0.5 * sum of (alpha_i * alpha_j * K(xi, xj)))**
      - Subject to:
        - **a) sum(alpha) <= ν * n (limits the fraction of support vectors)**
        - **b) alpha_i >= 0 (ensures non-negativity)**
    - - Identify "support vectors" as points with non-zero alpha values
    - - Select any one support vector, say `x_sv`, to compute the bias term `b`

EDA TECHNIQUES: HANDLING OUTLIERS USING ADVANCED STATS

- # 4. COMPUTE BIAS TERM
  - Calculate the bias term `b` using the selected support vector
    - `x_sv`: - `b = 1 - sum(alpha_i * K(xi, x_sv))` for each support vector xi
    - - Note: This ensures that `x_sv` lies exactly on the decision boundary
- # 5. PREDICTION
  - Define decision function for a new point `x_new`:
    - f(x_new) = sum(alpha_i * K(xi, x_new)) - b
    - - If f(x_new) >= 0, classify `x_new` as an inlier (normal)
    - - Else, classify `x_new` as an outlier (anomaly) # END OF ALGORITHM

EDA TECHNIQUES: HANDLING OUTLIERS USING ADVANCED STATS

- **Step 1: Define a Basic Dataset**
  - Consider this simple 5-point dataset in 2 dimensions:
  - X=[(1,2);(2,1);(2,2);(3,3);(10,10)]
  - In this dataset, the first four points cluster near each other, while the last point (10,10) is far away, which we might expect to be classified as an outlier.
- **Step 2: Initialize Parameters**
  - Let's assume:
    - **Gamma** = 0.5 (determines the influence of each point in the RBF kernel)
    - **ν** = 0.2 (allows 20% of points to be considered as outliers)

- **Step 3: Training Phase**
- **Compute the Kernel Matrix**
  Using the **RBF kernel**, we compute K(xi,xj)=exp(−γ·||xi−xj||2) j||2) for each pair of points:
  - Example calculations:
    - K(x1,x2)=exp(−0.5·||(1,2)−(2,1)||2)=0.3679
    - K(x1,x3)=exp(−0.5·||(1,2)−(2,2)||2)=0.6065
    - Completing the kernel matrix involves calculating these values for each pair, but for simplicity, we can just understand the general approach here.
- **Optimize Alphas and Identify Support Vectors**
  - Solve the quadratic optimization problem to maximize the margin.
  - After optimization, let's assume we find that only some points (like points 1, 3, and 4) have non-zero alpha values.
  - These points are our support vectors.
- **Compute Bias Term b Choose any support vector, say x_sv=(2,2).**
  - Use the formula $b=1-\sum(\alpha i \cdot K(xi,xsv))$ over all support vectors.

EDA TECHNIQUES: HANDLING OUTLIERS USING ADVANCED STATS

- **Step 4: Prediction Phase**
- To predict whether a new point is an inlier or outlier:
  - **Define the Decision Function**
    For a new point x_new=(3,3), calculate:
    - $f(x\_new)=\sum(\alpha i \cdot K(xi, x\_new))-b$
    - where xi are the support vectors.
    - This result will determine if x_new is inside the boundary (inlier) or outside (outlier).
  - **Classify the Point**
    - If $f(x\_new) \geq 0$: x_new is an **inlier**.
    - If $f(x\_new)<0$: x_new is an **outlier**.

- **Example Predictions**
  - For x_new=(10,10):
    - The decision function will likely yield $f(xnew)<0$, classifying it as an outlier.
  - For xnew=(2,1):
    - The decision function will likely yield $f(xnew)\geq 0$, classifying it as an inlier.
- This example shows the general approach, applying the core principles of One-Class SVM step-by-step.

ANALYZING PAIRWISE RELATIONSHIPS

- **Correlation**: Measures the strength and direction of a linear relationship between two variables (e.g., Pearson's correlation).
- **Covariance**: Shows the direction of the linear relationship between variables but is less intuitive than correlation.
- **Regression**: Estimates the relationship between dependent and independent variables (e.g., linear regression).
- **Scatter Plots**: A visual method to observe relationships between two continuous variables.
- **Chi-Square Test**: Examines the relationship between categorical variables.

ANALYZING PAIRWISE RELATIONSHIPS

1 Correlation (Pearson)

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{1}$$

Alternative form:

$$r = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} \tag{2}$$

2 Covariance

$$Cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \tag{3}$$

3 Chi-Square Test

The chi-square (χ^2) test statistic for independence:

$$\chi^2 = \sum_{i=1}^{r}\sum_{j=1}^{c}\frac{(O_{ij} - E_{ij})^2}{E_{ij}} \tag{4}$$

4 Linear Regression

Simple linear regression equation:

$$\hat{y} = b_0 + b_1 x \tag{5}$$

Slope (b_1) calculation:

$$b_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{6}$$

Y-intercept (b_0) calculation:

$$b_0 = \bar{y} - b_1 \bar{x} \tag{7}$$

ANALYZING PAIRWISE RELATIONSHIPS

Respondent	Gender	Movie Preference
1	Male	Action
2	Female	Comedy
3	Male	Comedy
4	Female	Comedy
5	Male	Action
6	Male	Comedy
7	Female	Comedy
8	Female	Action
9	Male	Action
10	Female	Comedy
...	...	...

ANALYZING PAIRWISE RELATIONSHIPS

- Suppose we want to test if there is an association between **gender** and **preference for a type of movie** (action or comedy). We have a **contingency table** from the previous survey.
- **Step 1: State Hypotheses**
  - **Null Hypothesis H0**: Gender and movie preference are independent (no association).
  - **Alternative Hypothesis H1**: Gender and movie preference are not independent (there is an association).

	Action	Comedy	Total
Male	30	20	50
Female	10	40	50
Total	40	60	100

ANALYZING PAIRWISE RELATIONSHIPS

- **Step 2: Calculate Expected Frequencies**
  - The **expected frequency** for each cell is calculated as:
    - Eij = (Row Total_i * Column Total_j)/ Total
      - (Male, Action): E11 = (50*40)/100
      - (Female, Comedy): E22 = (50*60)/100
- Step 3: Calculate the Chi-Squared Statistic
  - X2 = 16.66

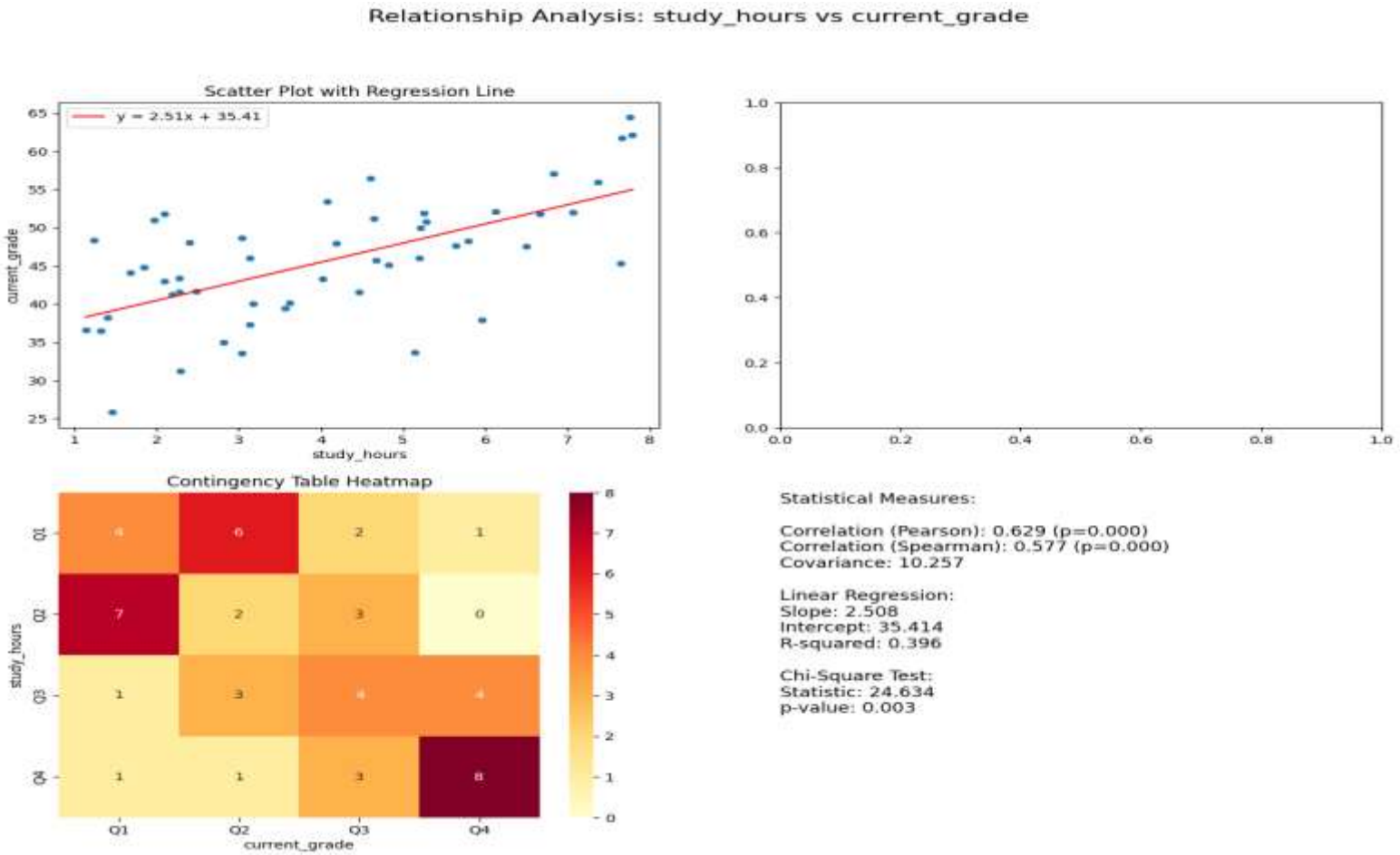
	Action (Expected)	Comedy (Expected)
Male	20	30
Female	20	30

ANALYZING PAIRWISE RELATIONSHIPS

- **Step 4: Determine Degrees of Freedom and Critical Value**
  - **Degrees of Freedom**: $(\text{Rows}-1)\times(\text{Columns}-1)=(2-1)\times(2-1)=1$
  - For a **0.05 significance level**, the critical value with 1 degree of freedom is 3.84.
- **Step 5: Make a Decision**
  - Since $\chi 2=16.66$ is greater than the critical value 3.84, we **reject the null hypothesis**.
  - This indicates there is likely an association between gender and movie preference.

	Action (Expected)	Comedy (Expected)
Male	20	30
Female	20	30

ANALYZING PAIRWISE RELATIONSHIPS: EXAMPLE

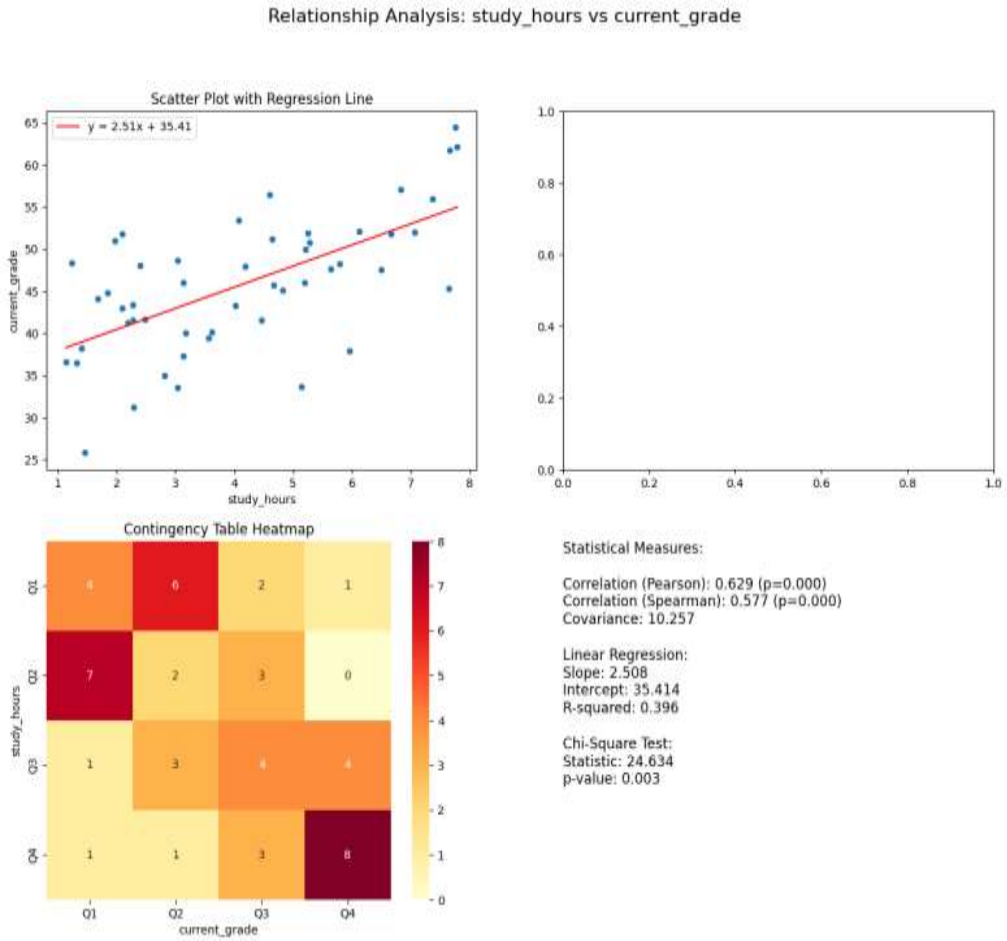


Sample Student Data:

	student_id	study_hours	attendance	prev_grade	current_grade
0	1	3.621781	98.783385	51.414313	40.145339
1	2	7.655000	91.005313	78.638469	61.715075
2	3	6.123958	97.579958	64.146019	52.079113
3	4	5.190609	95.793094	72.885681	45.963106
4	5	2.092130	83.915999	90.840491	51.830053

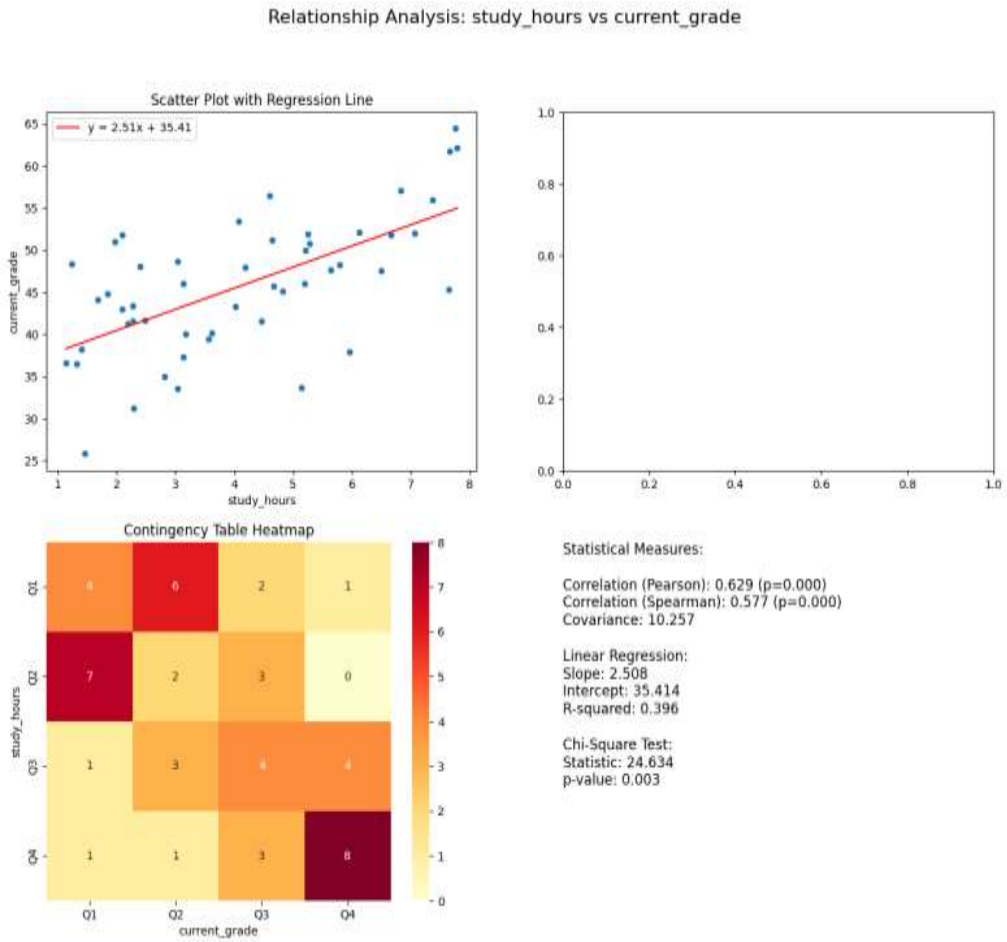
ANALYZING PAIRWISE RELATIONSHIPS: EXAMPLE

- **Data Generation**: Created realistic student data with:
  - Study hours (1-8 hours)
  - Attendance (60-100%)
  - Previous grades (50-95)
  - Current grades (calculated based on other variables)
- **Relationship Analysis**:
  - **Correlation**: Both Pearson (linear) and Spearman (rank) correlations
  - **Covariance**: Measures how variables change together
  - **Regression**: Linear regression with slope, intercept, and R-squared
  - **Chi-Square Test**: For testing independence of binned variables



ANALYZING PAIRWISE RELATIONSHIPS: EXAMPLE

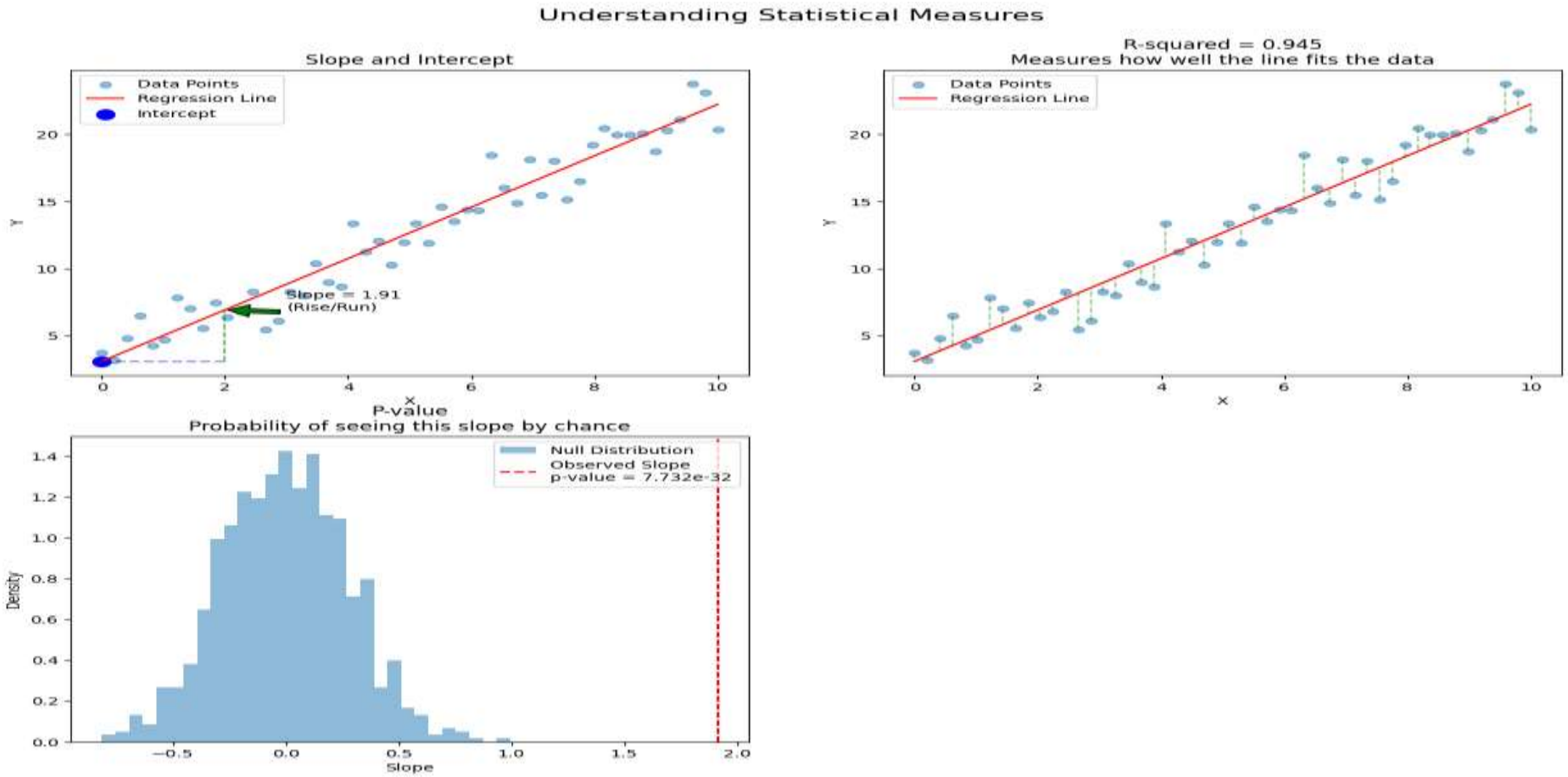
- **Visualizations**: Scatter plot with regression line
  - Joint plot showing distributions
  - Heatmap of contingency table
  - Summary statistics panel
  - Overall correlation matrix
- The example shows how:
  - Study hours positively correlate with grades
  - The relationship includes some natural variation
  - Different statistical measures capture different aspects of the relationship



ANALYZING PAIRWISE RELATIONSHIPS: STATISTICAL MEASURES

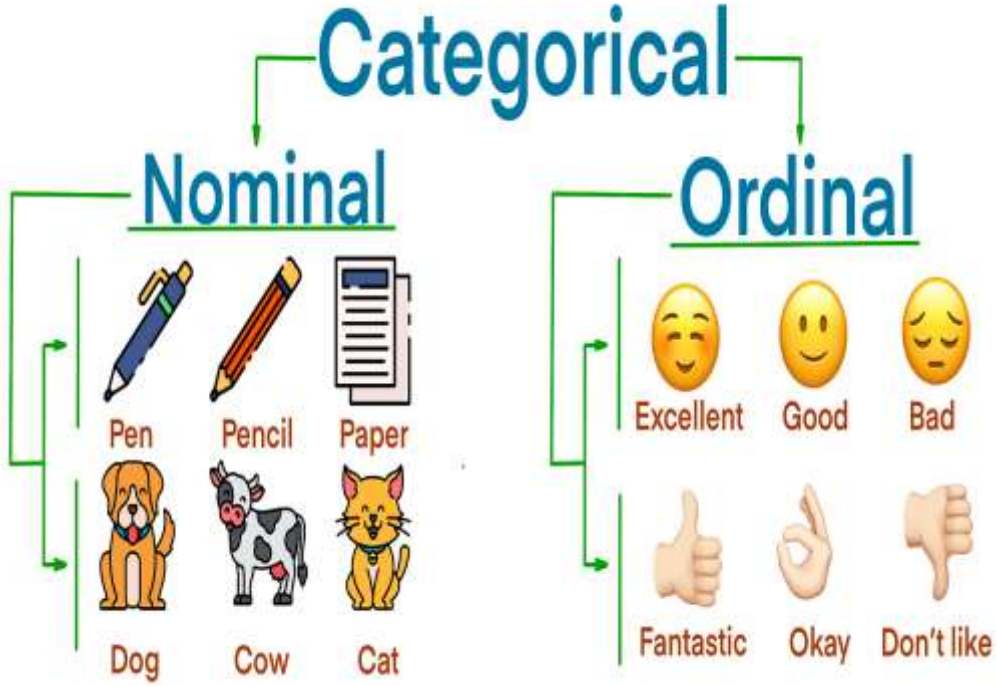
- **Slope (β_1)**:
  - The steepness of the relationshipHow much Y changes when X increases by 1 unit
  - Example: slope = 2 means Y increases by 2 units for each unit increase in X
- **Intercept (β_0)**:
  - Where the line crosses the Y-axisThe predicted Y value when X = 0
  - Example: intercept = 3 means Y starts at 3 when X = 0
- **R-squared (R^2):**
  - Measure of fit qualityRanges from 0 (poor fit) to 1 (perfect fit)
  - Example: R^2 = 0.75 means the model explains 75% of the variation
- **P-value:**
  - Probability of seeing this relationship by chance
  - Smaller values indicate stronger evidenceUsually compare to 0.05 threshold
  - Example: $p < 0.05$ suggests significant relationship
- Standard Error:
  - Uncertainty in our slope estimate
  - Smaller values indicate more precise estimatesUsed for confidence intervals

ANALYZING PAIRWISE RELATIONSHIPS: STATISTICAL MEASURES



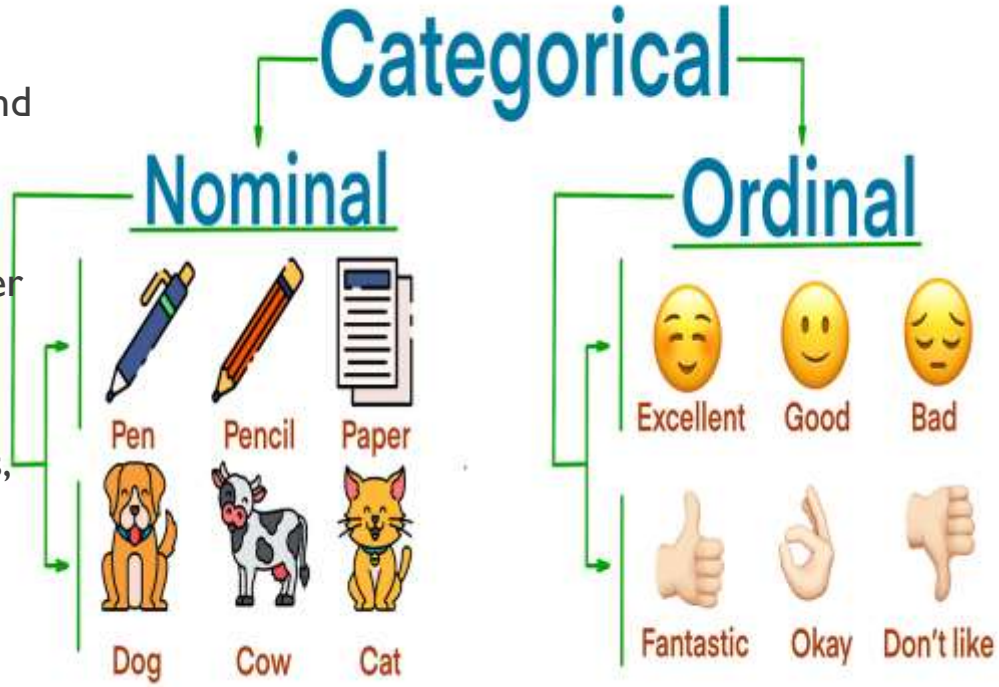
EDA TECHNIQUES: CATEGORICAL DATA

- Categorical data represents discrete values like labels or groups (e.g., gender, product type).
- Analyzing such data is key to understanding relationships and patterns.
- Exploring categorical data is crucial for extracting insights, preparing it for machine learning models, and understanding its role in the broader dataset.



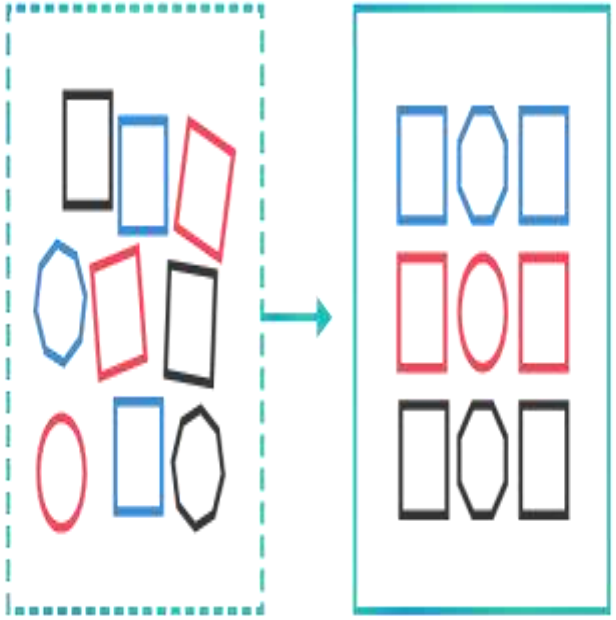
EDA TECHNIQUES: CATEGORICAL DATA

- **Different operations:**
  - **Frequency Tables:** A basic technique where the count of occurrences for each category is calculated. This helps in assessing the distribution and dominance of categories.
  - **Chi-Square Test:** A statistical method used to determine if there is a significant association between two categorical variables. It tests whether the observed frequencies deviate from expected ones.
  - **Contingency Tables:** Also known as cross-tabulation, this method summarizes the relationship between two or more categorical variables, helping to identify interactions.
  - **Encoding Methods or Transformation**:
    - One-Hot Encoding: Converts each category into a binary vector, suitable for machine learning models that can't process categorical variables directly.
    - Label Encoding: Assigns a unique integer to each category, which is efficient but assumes an ordinal relationship between categories



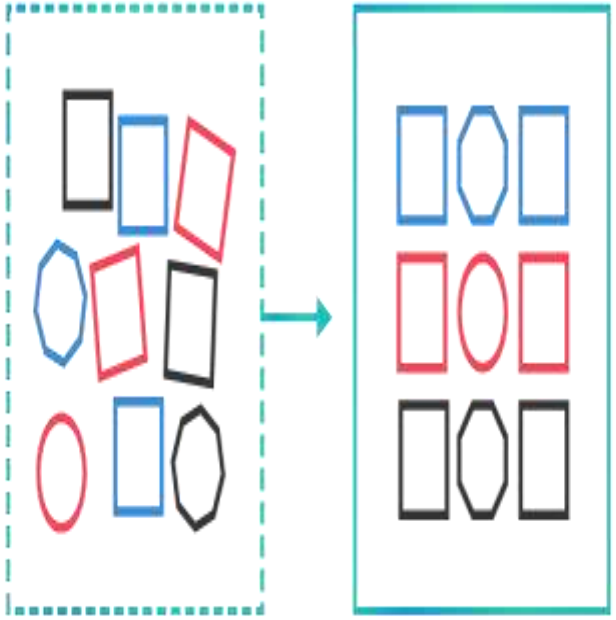
DATA TRANSFORMATION

- Data transformation is a key step in preparing raw data for analysis or modeling in data science.
- It involves converting data into a suitable format to improve accuracy and efficiency in subsequent stages.
- Data transformation techniques ensure that the data is in the right form for modeling, while also preserving its essential patterns and relationships.



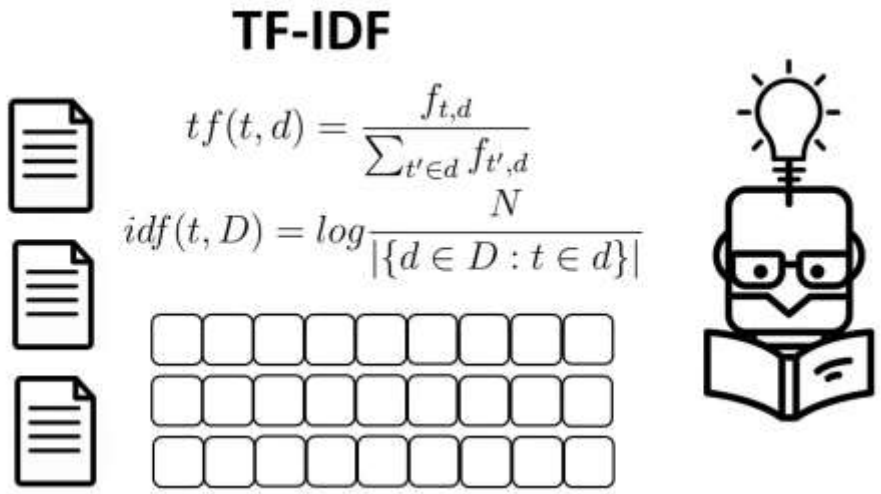
DATA TRANSFORMATION

- **Discretization**:
  - This process converts continuous data into discrete intervals or bins. It is often used when algorithms need categorical inputs, or when reducing the complexity of continuous variables.
  - For example, converting age into age groups (e.g., 0-18, 19-35, 36-60).
- **Handling Temporal Data**:
  - Temporal or time-series data requires special treatment to capture trends, seasonality, or patterns over time.
  - Common transformations include resampling data (e.g., aggregating daily data to monthly), creating lag features to capture dependencies, and applying rolling statistics like moving averages.

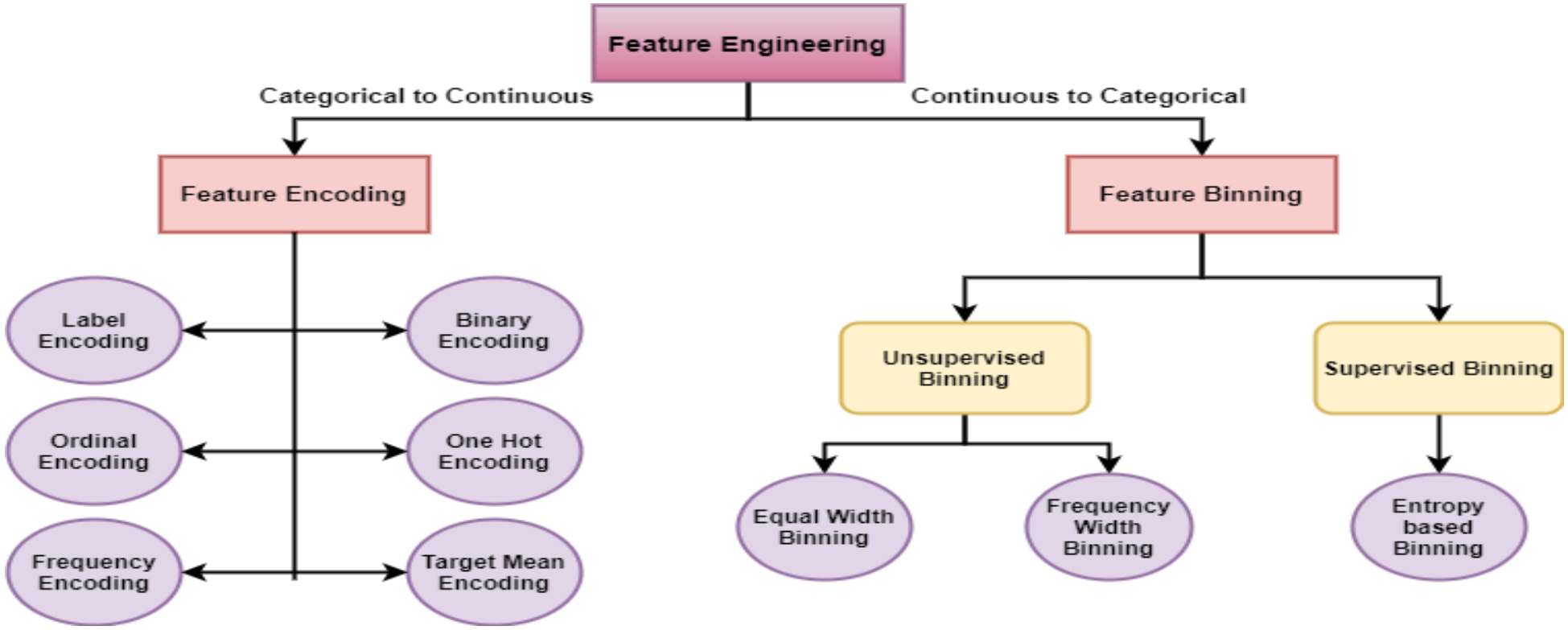


DATA TRANSFORMATION: TEXTUAL DATA

- **Managing Textual Data:**
  - Textual data, such as user reviews or social media posts, is unstructured and needs to be transformed into numerical representations.
  - Techniques include:
    - **Tokenization**: Splitting text into words or phrases.
    - **Bag-of-Words (BoW) or TF-IDF**: Converting text into numeric vectors based on word frequency or importance.
    - **Word Embeddings**: More advanced methods like Word2Vec or GloVe, which convert text into dense vector representations that capture word meanings.



DATA TRANSFORMATION: CATEGORICAL DATA



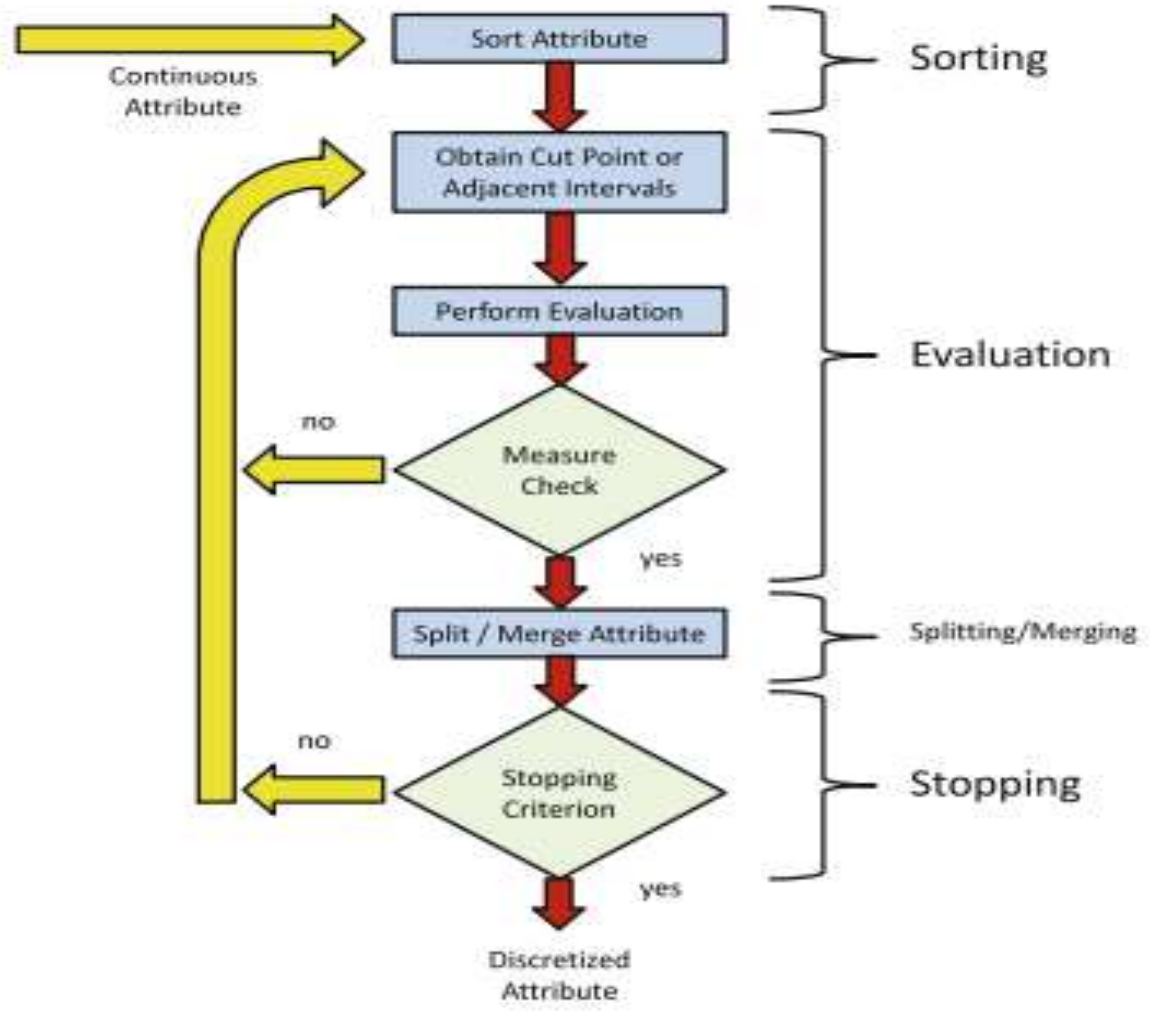
Feature Encoding techniques

DATA TRANSFORMATION: CATEGORICAL DATA

Encoding Technique	Example Categorical Feature	Method
One-Hot Encoding	Color (Red, Green, Blue)	Creates binary columns for each category (Color_Red, Color_Green, Color_Blue)
Label Encoding	Size (Small, Medium, Large)	Converts categories to numeric values (Small = 0, Medium = 1, Large = 2)
Target Encoding	Store Location	Replaces categories with the average target value for each category (avg sales for Downtown)
Frequency Encoding	City (e.g., City_A, City_B)	Replaces categories with their occurrence frequency (e.g., City_A = 50%)
Hashing Encoding	User IDs	Uses a hash function to convert categories into a fixed number of features

DATA TRANSFORMATION: NUMERICAL DATA

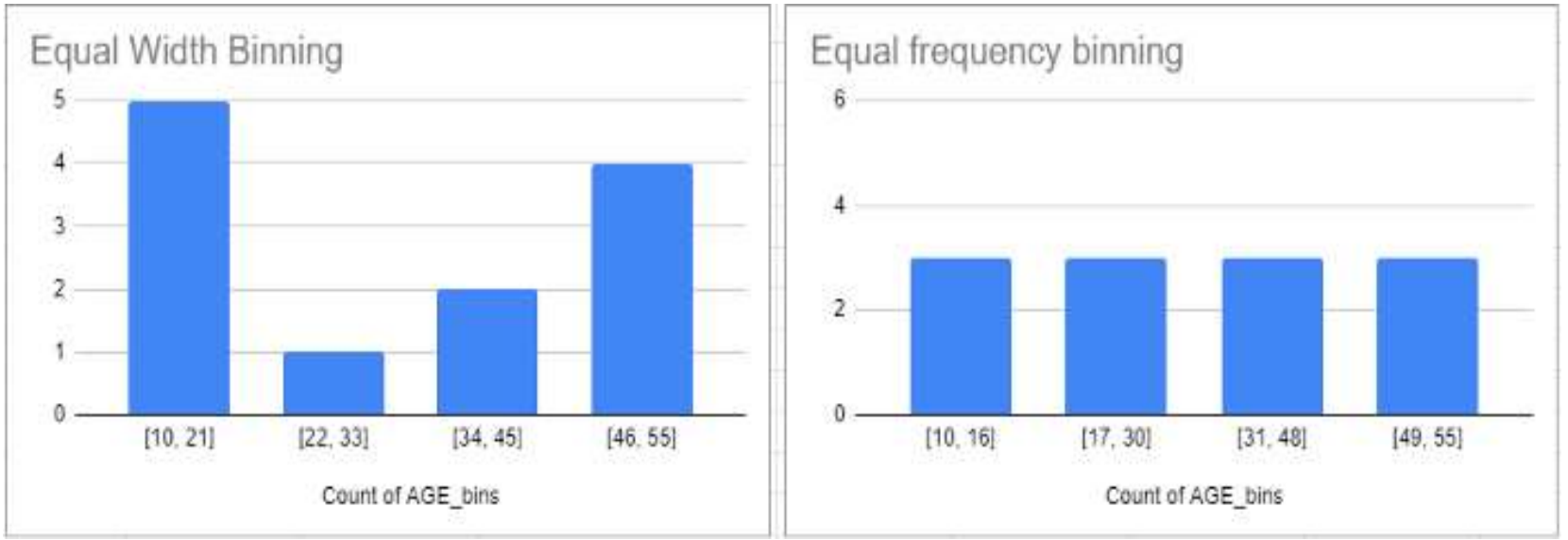
- **Discretization** consists of transforming quantitative values to discrete values by defining a number of intervals, then assigning each numerical value to one specific interval.



DATA TRANSFORMATION: NUMERICAL DATA

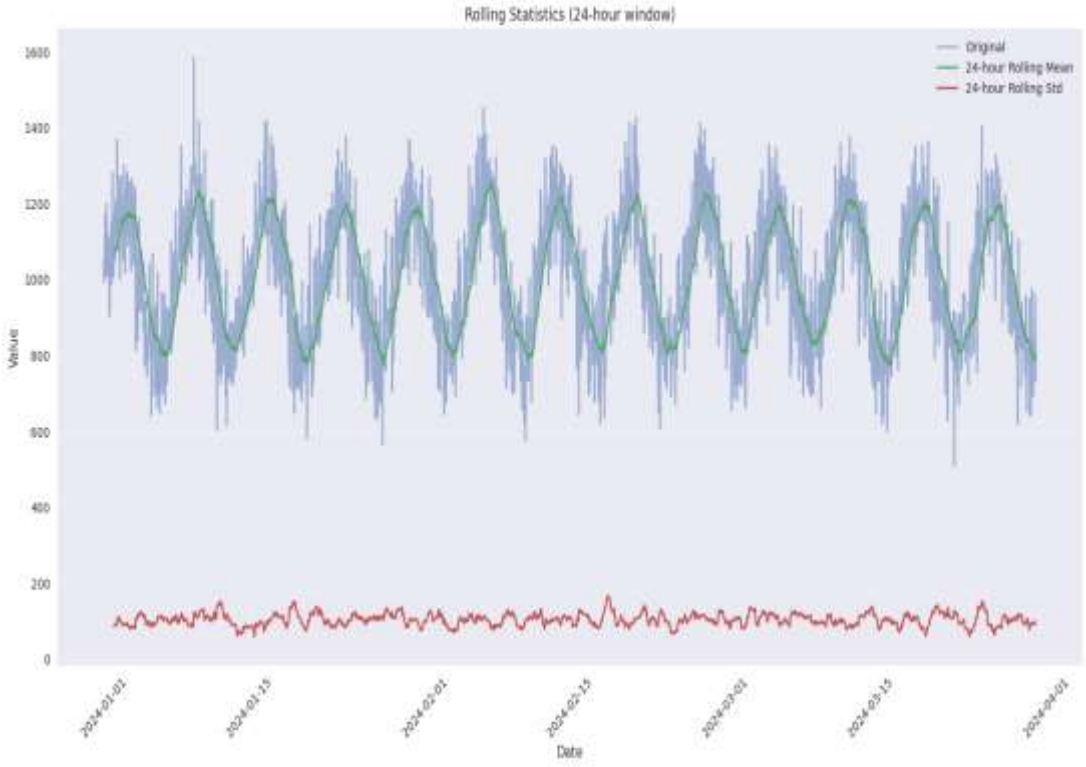
Transformation Method	Example Continuous Variable	Binning Description	Resulting Categorical Variable
Equal-Width Binning	Age	Divides the range of ages into equal intervals (e.g., 0-10, 11-20, 21-30)	Age Group (0-10, 11-20, 21-30)
Equal-Frequency Binning	Income	Divides the income data into bins with an equal number of observations (e.g., quartiles)	Income Quartiles (Q1, Q2, Q3, Q4)
Entropy-Based Binning	Temperature	Creates bins based on information gain, optimizing for predictive power (e.g., grouping temperatures for classifications)	Temperature Categories (Cold, Mild, Hot)

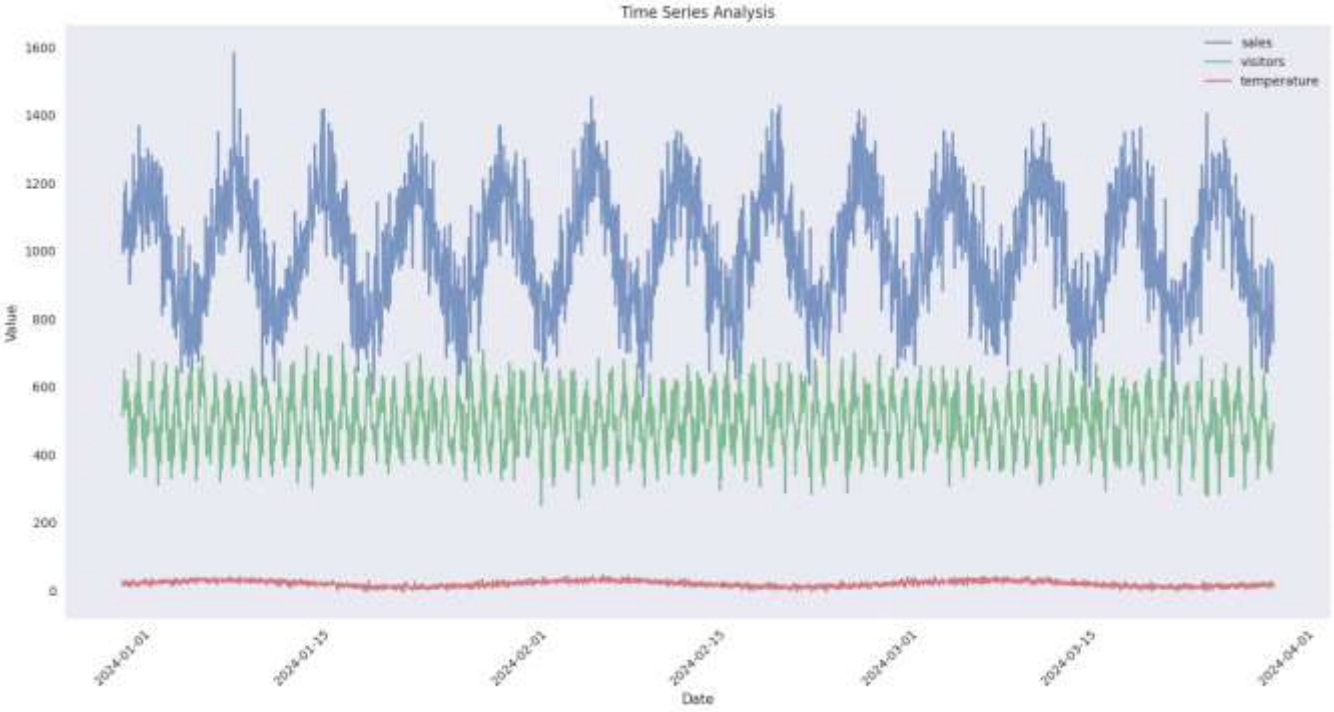
DATA TRANSFORMATION: NUMERICAL DATA



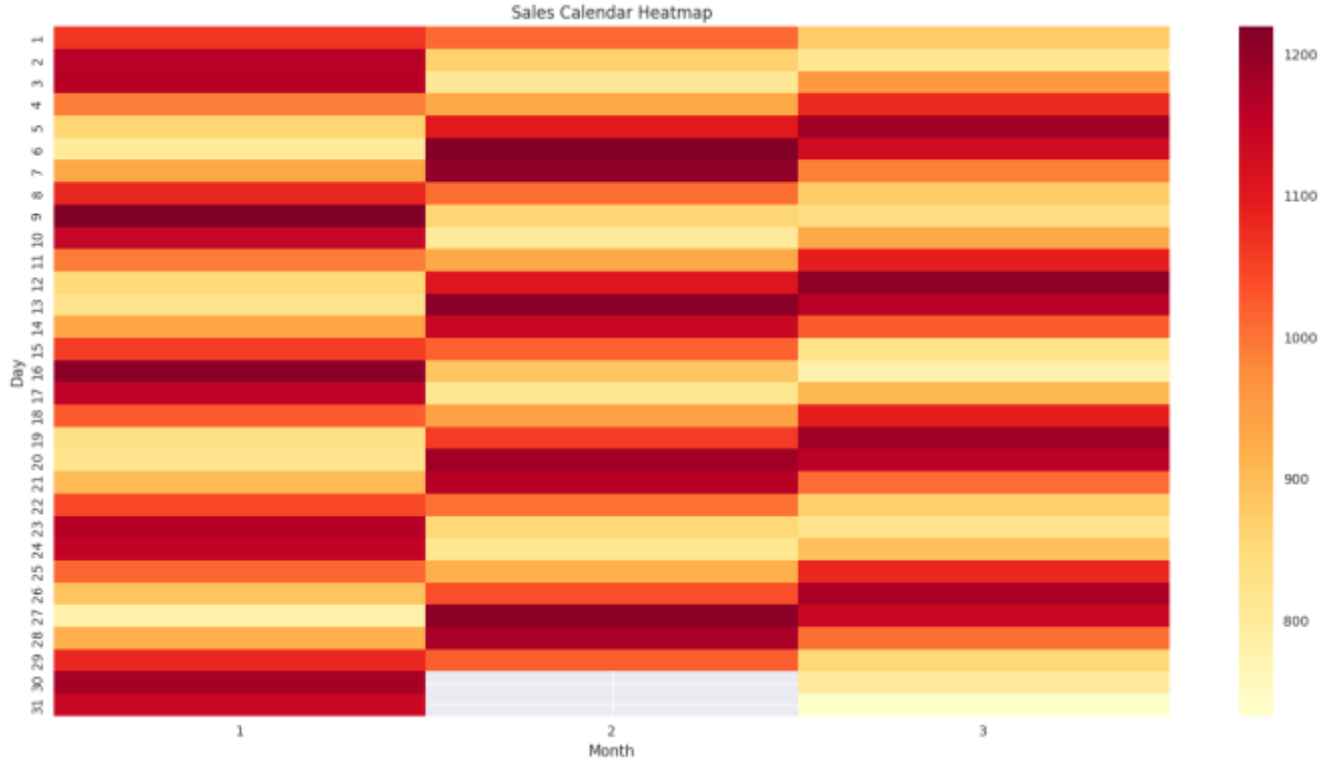
Comparison of different features discretization techniques

DATA TRANSFORMATION: EXAMPLE OF TEMPORAL DATA



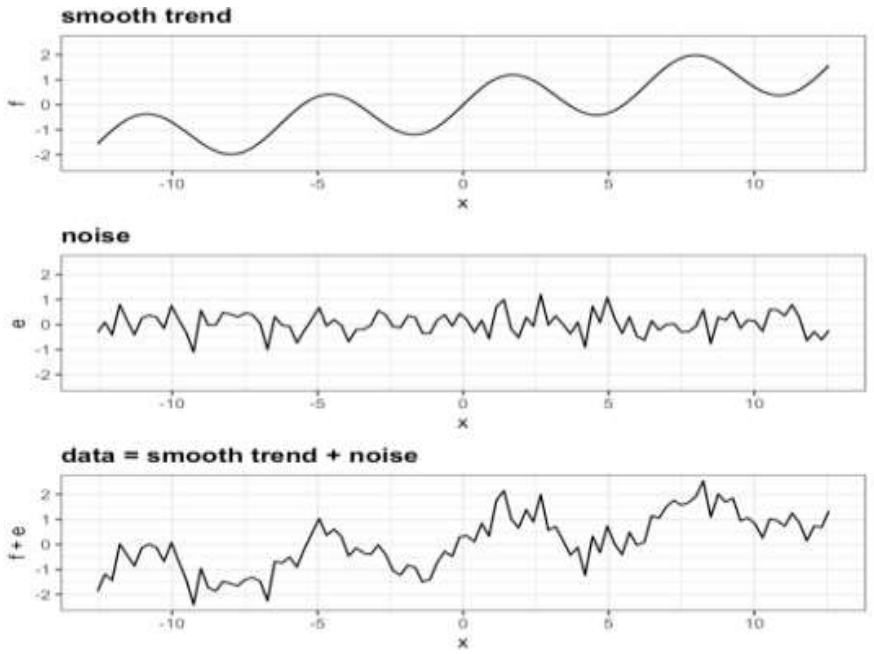


DATA TRANSFORMATION: EXAMPLE TEMPORAL DATA

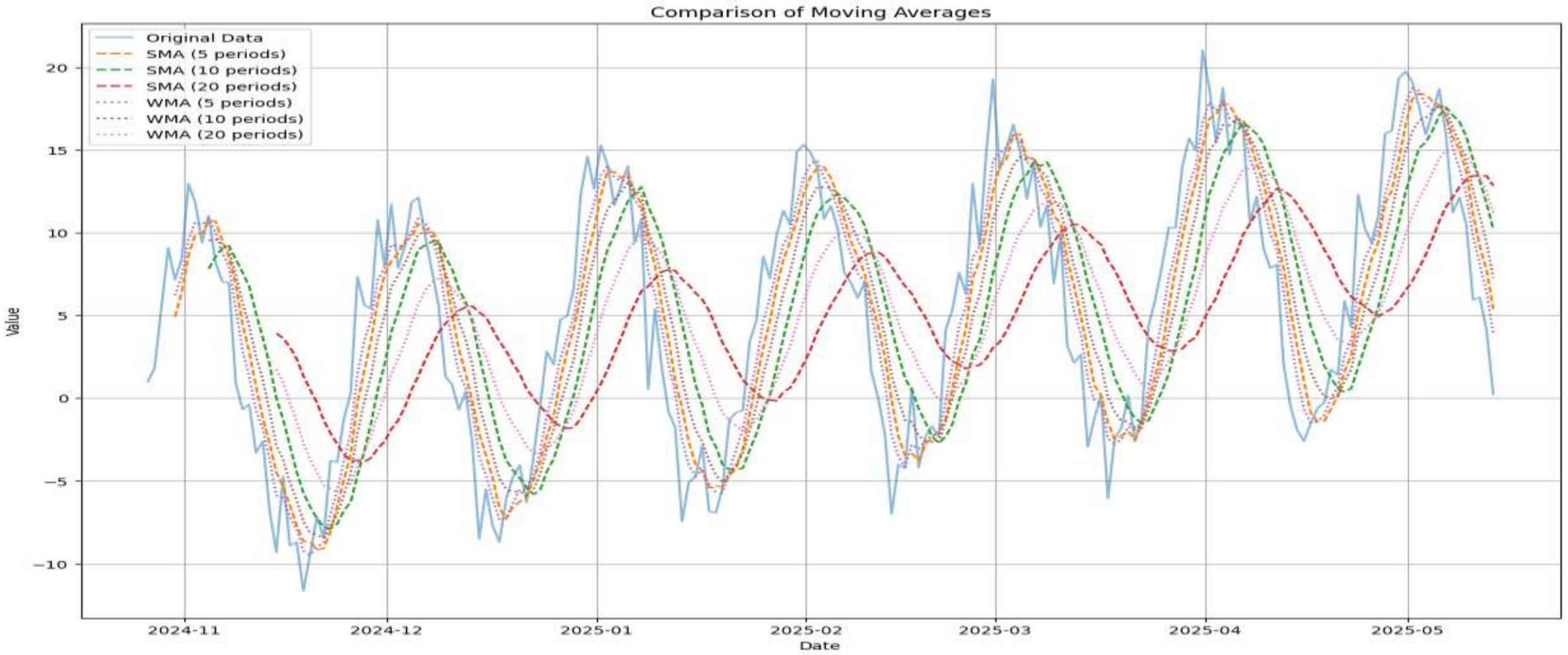


DATA SMOOTHING: BASIC TECHNIQUES

- The moving average is a commonly used statistical technique for smoothing time series data. It helps to identify trends by reducing noise and fluctuations in the data.
  - **Smoothing Using Simple Moving Average (SMA)**
  - **Smoothing Using Weighted Moving Average (WMA)**
- The choice between SMA and WMA depends on the specific needs of the analysis, with WMA providing a more responsive measure by giving greater importance to recent data.
- **Definition of Window**
  - In the context of moving averages, a **window** refers to the number of data points that are considered when calculating the average. For example, if the window size is 3, the moving average at each point in the data is calculated using the current point and the two previous points.



DATA SMOOTHING: BASIC TECHNIQUES



DATA SMOOTHING: BASIC TECHNIQUES

1. **Simple Moving Average (SMA):**
   - The **Simple Moving Average** is calculated by taking the arithmetic mean of the data points within the specified window.
   - Formula:

$$\text{SMA}_t = \frac{1}{n} \sum_{i=0}^{n-1} x_{t-i}$$

   where n is the window size, and x_t is the data point at time t.

2. **Weighted Moving Average (WMA):**
   - The **Weighted Moving Average** assigns different weights to each data point within the window. More recent data points typically receive higher weights.
   - Formula:

$$\text{WMA}_t = \frac{\sum_{i=0}^{n-1} w_i \cdot x_{t-i}}{\sum_{i=0}^{n-1} w_i}$$

   where w_i represents the weight assigned to the i-th data point.

DATA SMOOTHING: BASIC TECHNIQUES

Window Size = 3

- Calculate SMA for each point (starting from day 3, as it requires the previous two days):

Day	Sales	SMA (Window = 3)
1	10	-
2	15	-
3	20	(10 + 15 + 20)/3 = 15
4	25	(15 + 20 + 25)/3 = 20
5	30	(20 + 25 + 30)/3 = 25
6	35	(25 + 30 + 35)/3 = 30
7	40	(30 + 35 + 40)/3 = 35

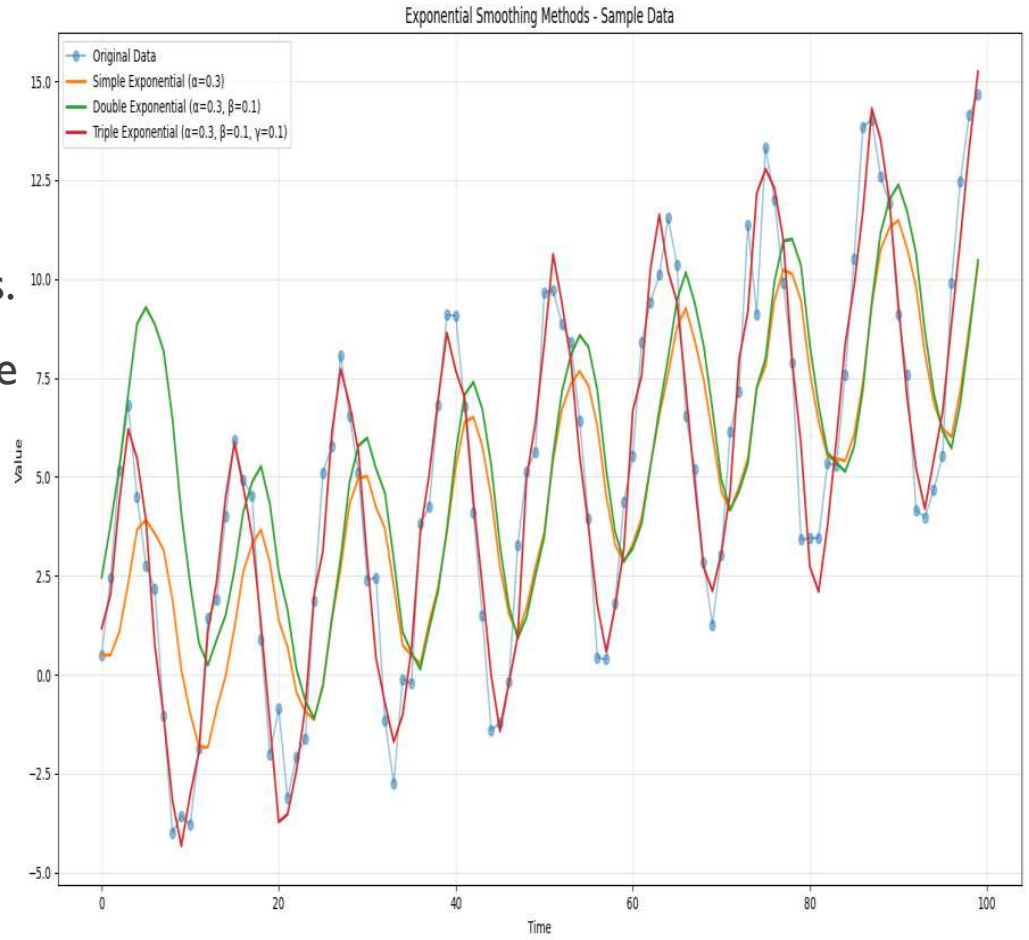
Window Size = 3 with Weights: [0.1, 0.3, 0.6]

- Calculate WMA for each point (starting from day 3):

Day	Sales	WMA (Window = 3)
1	10	-
2	15	-
3	20	(0.1 * 10 + 0.3 * 15 + 0.6 * 20) / (0.1 + 0.3 + 0.6) = 17
4	25	(0.1 * 15 + 0.3 * 20 + 0.6 * 25) / (0.1 + 0.3 + 0.6) = 23
5	30	(0.1 * 20 + 0.3 * 25 + 0.6 * 30) / (0.1 + 0.3 + 0.6) = 28
6	35	(0.1 * 25 + 0.3 * 30 + 0.6 * 35) / (0.1 + 0.3 + 0.6) = 33
7	40	(0.1 * 30 + 0.3 * 35 + 0.6 * 40) / (0.1 + 0.3 + 0.6) = 38

DATA SMOOTHING: EXPONENTIAL METHODS

- Exponential smoothing is a widely used forecasting method that applies weighted averages to estimate future values of a time series.
- Unlike simple moving averages, exponential smoothing assigns more weight to recent observations, making it an effective tool for data with trends and seasonal variations.
- Exponential smoothing, through its simple, double, and triple variants, provides robust methods for forecasting time series data while accounting for trends and seasonality.



DATA SMOOTHING: EXPONENTIAL METHODS

1. Simple Exponential Smoothing (SES)

Simple Exponential Smoothing is used for time series data without trend or seasonality. It calculat a forecast based on the last observation, weighted by a smoothing factor, α ($0 < \alpha < 1$).

- **Formula:**

$$F_t = \alpha \cdot Y_{t-1} + (1 - \alpha) \cdot F_{t-1}$$

where:

- F_t = forecast for time t
- Y_{t-1} = actual value at time $t - 1$
- F_{t-1} = forecast at time $t - 1$

Importance of α:

- A high α (close to 1) gives more weight to recent values, making the model more responsive.
- A low α (close to 0) gives more weight to older values, stabilizing the forecast.

DATA SMOOTHING: EXPONENTIAL METHODS

2. Double Exponential Smoothing (Holt's Method)

Double Exponential Smoothing, or Holt's method, is used for time series data with trends. It incorporates both level and trend in the model.

- **Formulas:**
  - Level:

$$L_t = \alpha \cdot Y_t + (1 - \alpha) \cdot (L_{t-1} + T_{t-1})$$

  - Trend:

$$T_t = \beta \cdot (L_t - L_{t-1}) + (1 - \beta) \cdot T_{t-1}$$

  - Forecast:

$$F_{t+h} = L_t + h \cdot T_t$$

  where h is the number of periods to forecast, and β is the smoothing factor for the trend.

Importance of α and β:

- α adjusts the weight of observations for the level.
- β adjusts the weight of observations for the trend.

DATA SMOOTHING: EXPONENTIAL METHODS

3. Triple Exponential Smoothing (Holt-Winters Method)

Triple Exponential Smoothing, or Holt-Winters method, is used for time series data with both trend and seasonality. It can be additive or multiplicative.

- **Formulas (additive):**
  - Level:

$$L_t = \alpha \cdot (Y_t - S_{t-m}) + (1 - \alpha) \cdot (L_{t-1} + T_{t-1})$$

  - Trend:

$$T_t = \beta \cdot (L_t - L_{t-1}) + (1 - \beta) \cdot T_{t-1}$$

  - Seasonality:

$$S_t = \gamma \cdot (Y_t - L_t) + (1 - \gamma) \cdot S_{t-m}$$

  - Forecast:

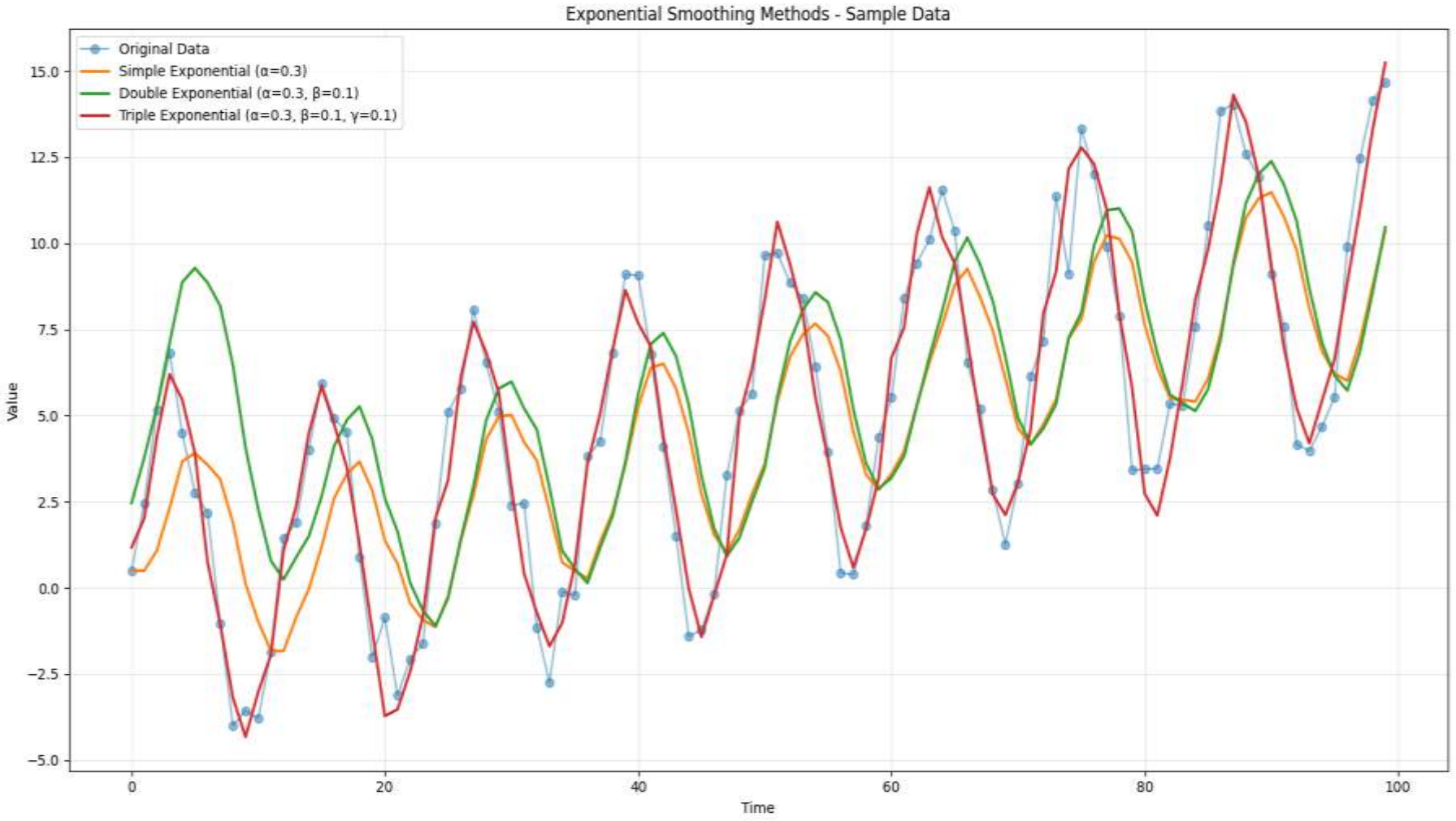
$$F_{t+h} = L_t + h \cdot T_t + S_{t-m+h}$$

  where S_{t-m+h} is the seasonality for future periods.

Importance of α, β, and γ:

- α adjusts the weight of observations for the level.
- β adjusts the weight for the trend.
- γ adjusts the weight for seasonality.

DATA SMOOTHING: EXPONENTIAL METHODS

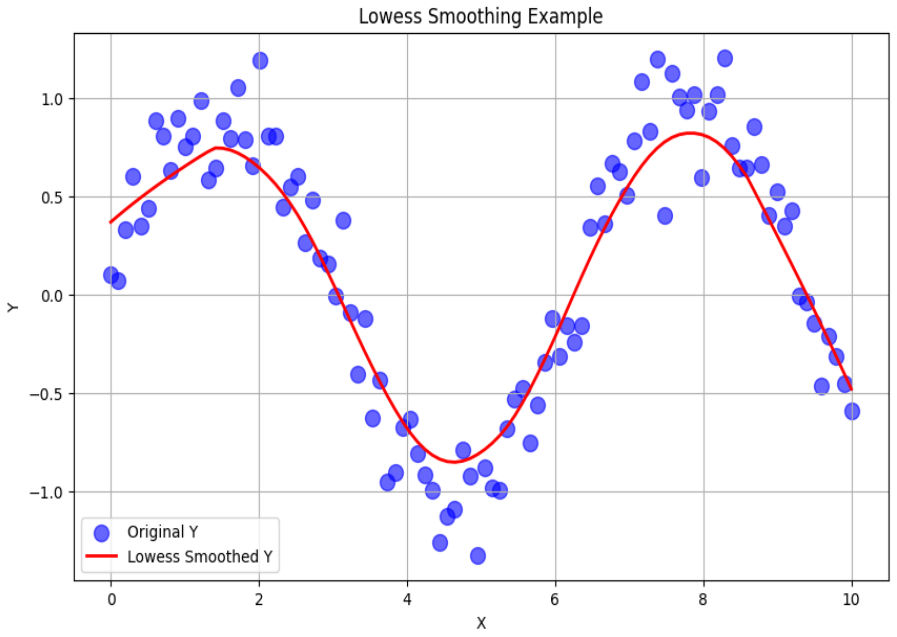


Error Metrics:

Method	MSE	RMSE
Simple Exponential	12.50	3.54
Double Exponential	17.12	4.14
Triple Exponential	1.03	1.01

DATA SMOOTHING:
LOCAL WEIGHTING TECHNIQUES: LOWESS SMOOTHING

- **What is Lowess Smoothing?**
  - **Definition**: Lowess (Locally Weighted Scatterplot Smoothing) is a non-parametric technique for smoothing data, especially in scatter plots, to observe trends.
  - **Purpose**: Creates a smooth curve by fitting local regressions to subsets of data, helping identify underlying patterns in noisy datasets.
- **How Does It Work?**
  - **Local Regression**: Calculates a weighted least squares fit for each data point.
  - **Weighting**: Nearby points have higher weights, and distant points have lower weights.
  - **Flexible Fit**: Adjusts to local data variations, making it suitable for nonlinear relationships.
- **Key Applications**
  - **Data Smoothing**: Reduces noise in time-series or spatial data.
  - **Trend Analysis**: Useful in economics, biology, and social sciences to reveal underlying trends



DATA SMOOTHING:
LOCAL WEIGHTING TECHNIQUES: LOWESS SMOOTHING

1. **Select a Neighborhood Size (Bandwith):**
   - Choose a bandwidth (usually a proportion of the data points, e.g., 0.3) that determines how many neighboring points will be used for the smoothing calculation.
2. **Fit Local Regression:**
   - For each point X_i, fit a local regression using weighted least squares. The weight is determined by the distance of each point to X_i, often using a tricube weight function:

$$w_j = \left(1 - \left(\frac{|X_j - X_i|}{h}\right)^3\right)^3 \quad \text{for } |X_j - X_i| < h$$

where h is the bandwidth.

X	Original Y	Smoothed Y (Lowess)
1	2.5	2.6
2	3.0	3.1
3	3.5	3.4
4	5.0	4.8
5	4.5	4.7
6	6.0	5.8
7	7.5	6.9
8	8.0	7.5
9	8.5	8.1
10	9.0	8.8

DATA SMOOTHING:
LOCAL WEIGHTING TECHNIQUES: LOWESS SMOOTHING

3. **Calculate Weights:**
   - For each X_i, compute the weights for all X_j based on their distance to X_i.
4. **Perform Weighted Least Squares:**
   - Solve for the coefficients of the local regression:

$$\hat{Y}_i = \arg\min_{\beta_0,\beta_1} \sum_j w_j (Y_j - (\beta_0 + \beta_1 X_j))^2$$

   This can be done using matrix operations.
5. **Predict Smoothed Values:**
   - The predicted value $\hat{Y}_i$ for each X_i is obtained using the fitted model.

X	Original Y	Smoothed Y (Lowess)
1	2.5	2.6
2	3.0	3.1
3	3.5	3.4
4	5.0	4.8
5	4.5	4.7
6	6.0	5.8
7	7.5	6.9
8	8.0	7.5
9	8.5	8.1
10	9.0	8.8

DATA SMOOTHING:
LOCAL WEIGHTING TECHNIQUES: LOWESS SMOOTHING

Example Calculation for X = 5

1. **Select Bandwidth:** Let's say we choose $h = 2$ (considering 2 units on either side of $X = 5$).
2. **Calculate Weights:**
   - For $X = 5$:
     - Points used: $X = [3, 4, 5, 6, 7]$ (indexes 3, 4, 5, 6, 7).
     - Weights calculated using the tricube function.
3. **Calculate Weights Example:**
   - Assuming X_j are the indices around $X_i = 5$:
     - $X_3 = 3$: $w_3 = \left(1 - \left(\frac{|3-5|}{2}\right)^3\right)^3 = (1 - 0.5^3)^3 = 0.875$
     - $X_4 = 4$: $w_4 = \left(1 - \left(\frac{|4-5|}{2}\right)^3\right)^3 = (1 - 0.25^3)^3 = 0.9375$
     - $X_5 = 5$: $w_5 = 1$
     - $X_6 = 6$: $w_6 = 0.9375$
     - $X_7 = 7$: $w_7 = 0.875$

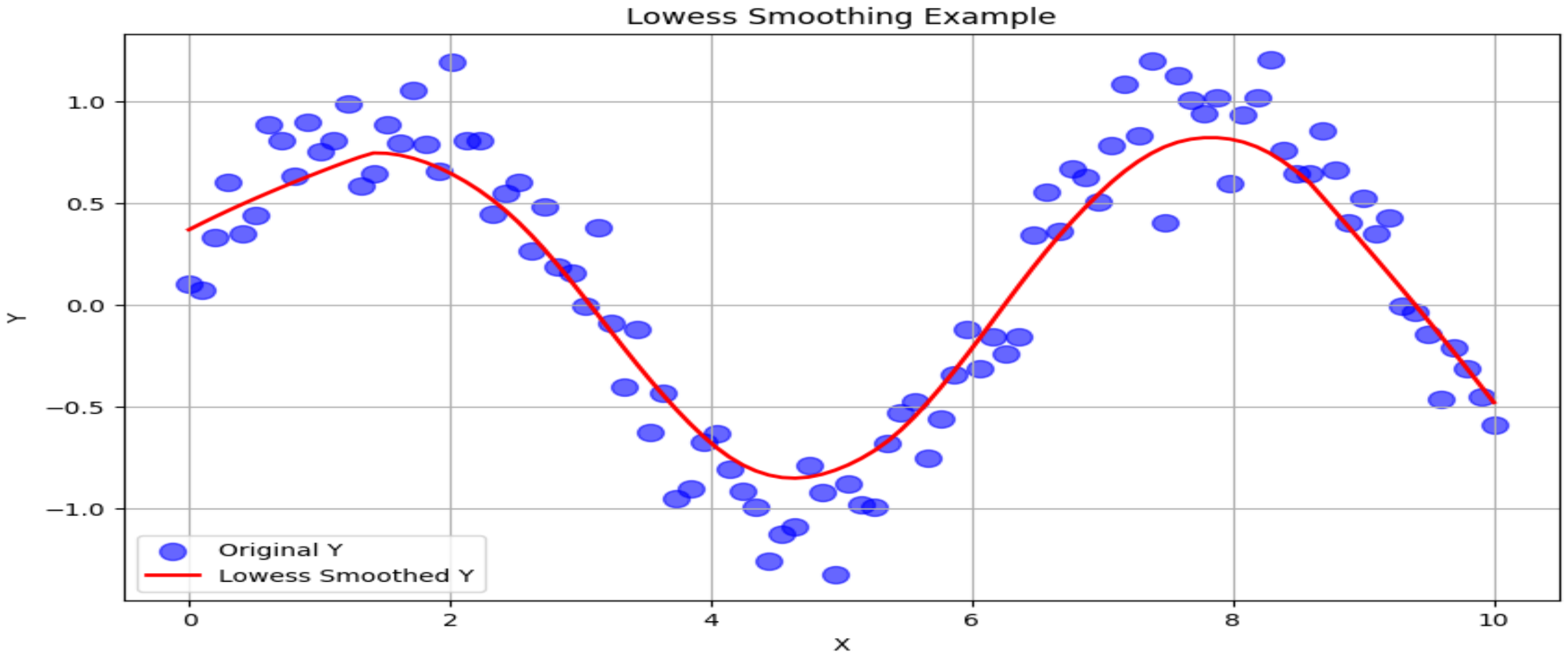
X	Original Y	Smoothed Y (Lowess)
1	2.5	2.6
2	3.0	3.1
3	3.5	3.4
4	5.0	4.8
5	4.5	4.7
6	6.0	5.8
7	7.5	6.9
8	8.0	7.5
9	8.5	8.1
10	9.0	8.8

DATA SMOOTHING:
LOCAL WEIGHTING TECHNIQUES: LOWESS SMOOTHING

4. **Perform Weighted Least Squares:**
   - Formulate the system of equations using the weighted values and solve for β_0 and β_1. This involves creating matrices for the weights and the X and Y values and using normal equations to find the best fit.
5. **Predict Smoothed Value:**
   - The resulting $\hat{Y}_5$ would be calculated as part of the weighted least squares fit.

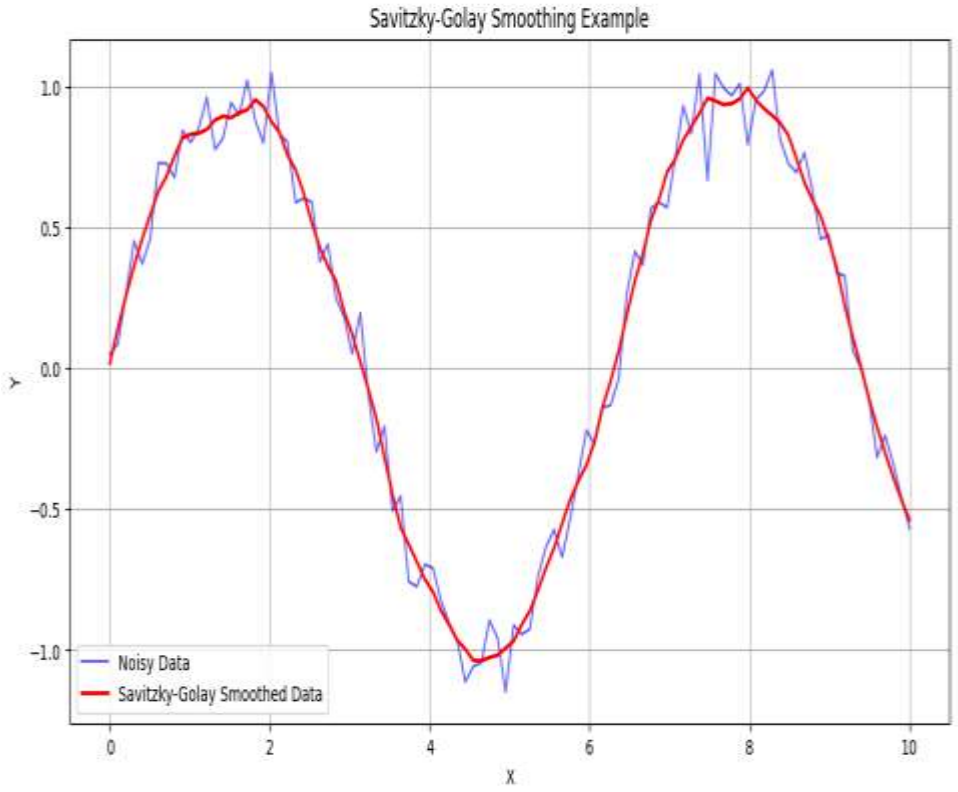
X	Original Y	Smoothed Y (Lowess)
1	2.5	2.6
2	3.0	3.1
3	3.5	3.4
4	5.0	4.8
5	4.5	4.7
6	6.0	5.8
7	7.5	6.9
8	8.0	7.5
9	8.5	8.1
10	9.0	8.8

DATA SMOOTHING:
LOCAL WEIGHTING TECHNIQUES: LOWESS SMOOTHING



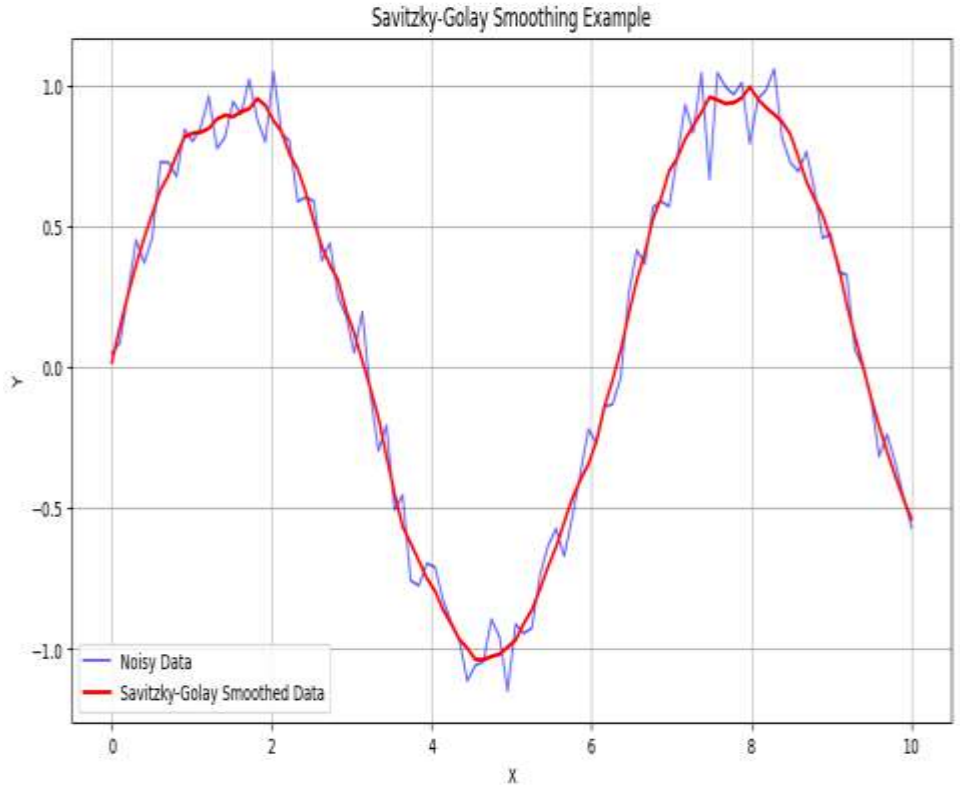
DATA SMOOTHING: **SAVITZKY-GOLAY FILTER**

- The **Savitzky-Golay (SG) Filter** is a digital filtering technique that smooths data by fitting successive sub-sets of adjacent data points with a **low-degree polynomial** by the method of least squares.
- This filter is particularly useful in signal processing and data analysis for reducing noise while preserving important features of the data, such as peaks and valleys.
- **Key Features**
  - **Preserves Features**: Unlike simple averaging filters, the **SG** filter maintains the shape and features of the data, making it especially effective for data with sharp changes or high-frequency noise.
  - **Polynomial Fitting**: The filter fits a polynomial to a specified number of data points (the window size) around each point in the dataset, allowing for local smoothing.
  - **Customizable**: The degree of the polynomial and the size of the moving window can be adjusted according to the needs of the data being analyzed.



DATA SMOOTHING: **SAVITZKY-GOLAY FILTER**

- **Applications**
  - **Spectroscopy**: Commonly used to smooth spectroscopic data, enhancing the resolution of spectral features.
  - **Signal Processing**: Used in various signal processing applications to reduce noise in measured data while preserving signal characteristics.
  - **Time Series Analysis**: Effective in smoothing time series data for better visualization and analysis.



DATA SMOOTHING: **SAVITZKY-GOLAY FILTER**

Step 1: Choose Parameters

- **Window Length**: Let's choose a window length of 5 (we will use 5 points for fitting).
- **Polynomial Order**: We'll use a polynomial of order 2 (quadratic).

Step 2: Apply Savitzky-Golay Filter

Calculate the Polynomial Coefficients

1. **Select the Window**: For each point $Y[i]$, we will use the points $Y[i-2]$, $Y[i-1]$, $Y[i]$, $Y[i+1]$, and $Y[i+2]$.
2. **Fit a Polynomial**: For each window, we fit a polynomial using the least squares method.

X	Y (Noisy Data)
0.0	0.05
1.0	0.84
2.0	0.75
3.0	0.60
4.0	0.95
5.0	0.84
6.0	0.30
7.0	0.40
8.0	0.90
9.0	0.70
10.0	0.00

DATA SMOOTHING: **SAVITZKY-GOLAY FILTER**

Example Calculation for $Y[2]$ (Using Points $Y[0], Y[1], Y[2], Y[3], Y[4]$):

- Data Points: $X = [0, 1, 2, 3, 4]$ and $Y = [0.05, 0.84, 0.75, 0.60, 0.95]$
- Polynomial Form: We fit a quadratic polynomial $P(x) = a_0 + a_1 x + a_2 x^2$.
- Matrix Representation: Set up the following equations based on least squares:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0.05 \\ 0.84 \\ 0.75 \\ 0.60 \\ 0.95 \end{bmatrix}$$

Using a numerical library (or performing the calculations by hand), we can solve for a_0, a_1, a_2.

X	Y (Noisy Data)
0.0	0.05
1.0	0.84
2.0	0.75
3.0	0.60
4.0	0.95
5.0	0.84
6.0	0.30
7.0	0.40
8.0	0.90
9.0	0.70
10.0	0.00

DATA SMOOTHING: **SAVITZKY-GOLAY FILTER**

Step 3: Calculate Smoothed Values

For each center point (e.g., $Y[2]$), calculate the polynomial value using the coefficients derived from the fitting:

1. For $Y[2]$: After calculating coefficients (for example $a_0 = 0.7, a_1 = -0.1, a_2 = 0.05$), compute:

$$P(2) = a_0 + a_1 \cdot 2 + a_2 \cdot (2^2)$$

Step 4: Repeat for Other Points

Continue this process for each point in the dataset where you can apply the window (i.e., $Y[2]$ to $Y[7]$).

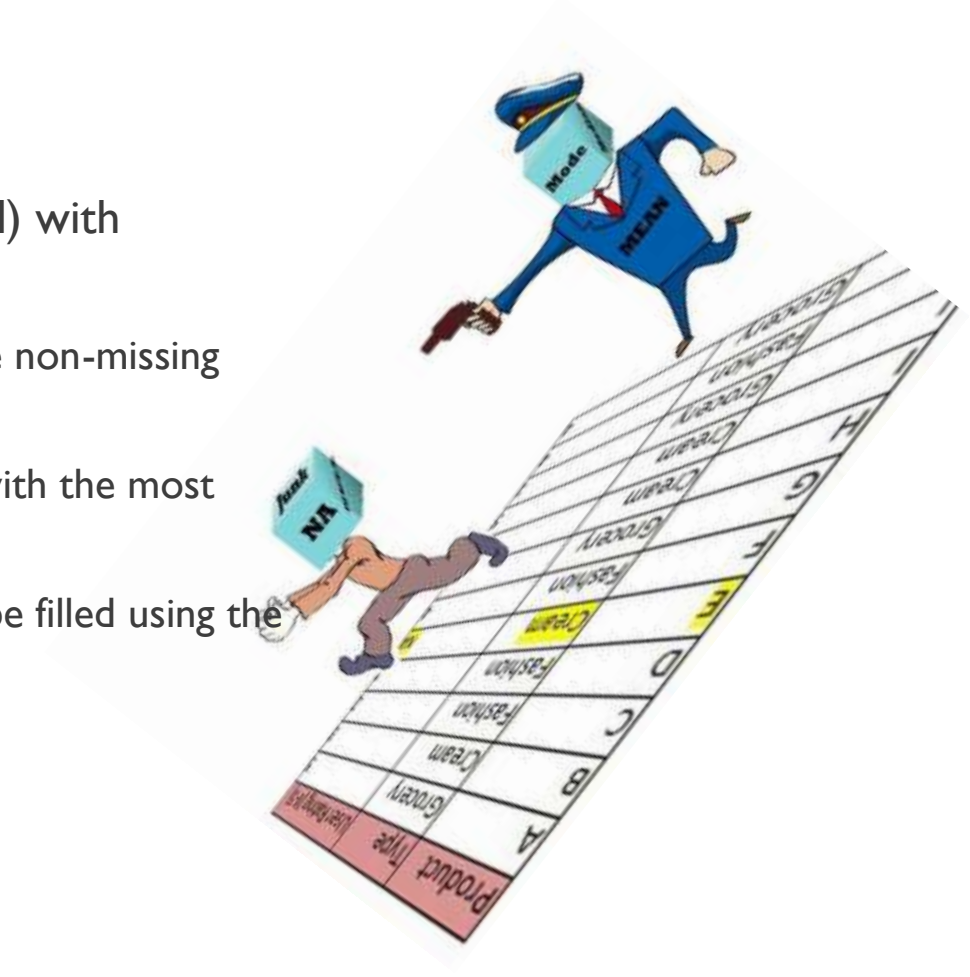
X	Y (Noisy Data)
0.0	0.05
1.0	0.84
2.0	0.75
3.0	0.60
4.0	0.95
5.0	0.84
6.0	0.30
7.0	0.40
8.0	0.90
9.0	0.70
10.0	0.00

BASIC DATA PREPROCESSING

- How do I clean up the data?—Data Cleaning.
- How do I provide accurate data?—Data Transformation.
- How do I incorporate and adjust data?—Data Integration.
- How do I unify and scale data?—Data Normalization.
- How do I handle missing data?—Missing Data Imputation.
- How do I detect and manage noise?—Noise Identification.

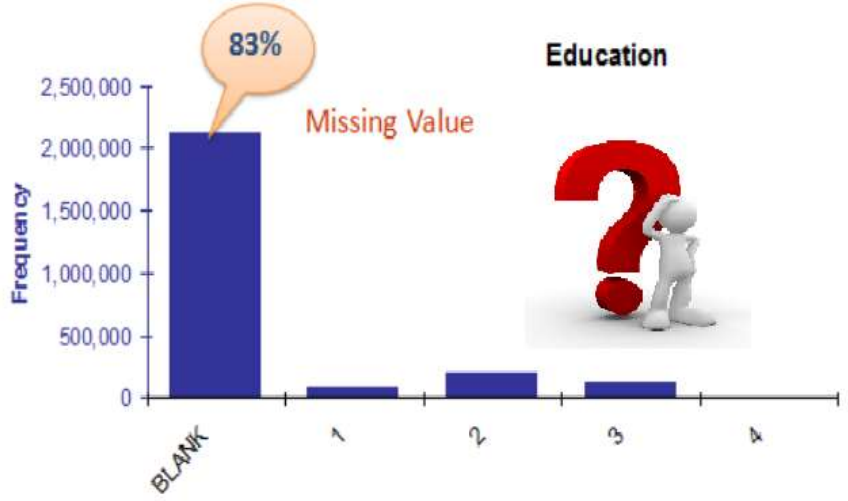
BASIC DATA PREPROCESSING: MISSING VALUES

- **Imputation:** Instead of dropping data, missing values can be filled (imputed) with statistical measures such as:
  - Mean/Median Imputation: Replace missing values with the mean or median of the non-missing data.
  - Mode Imputation: Used for categorical data, where missing values are replaced with the most frequent category.
  - Forward/Backward Filling: For time series or sequential data, missing values can be filled using the value from the previous or next time step.



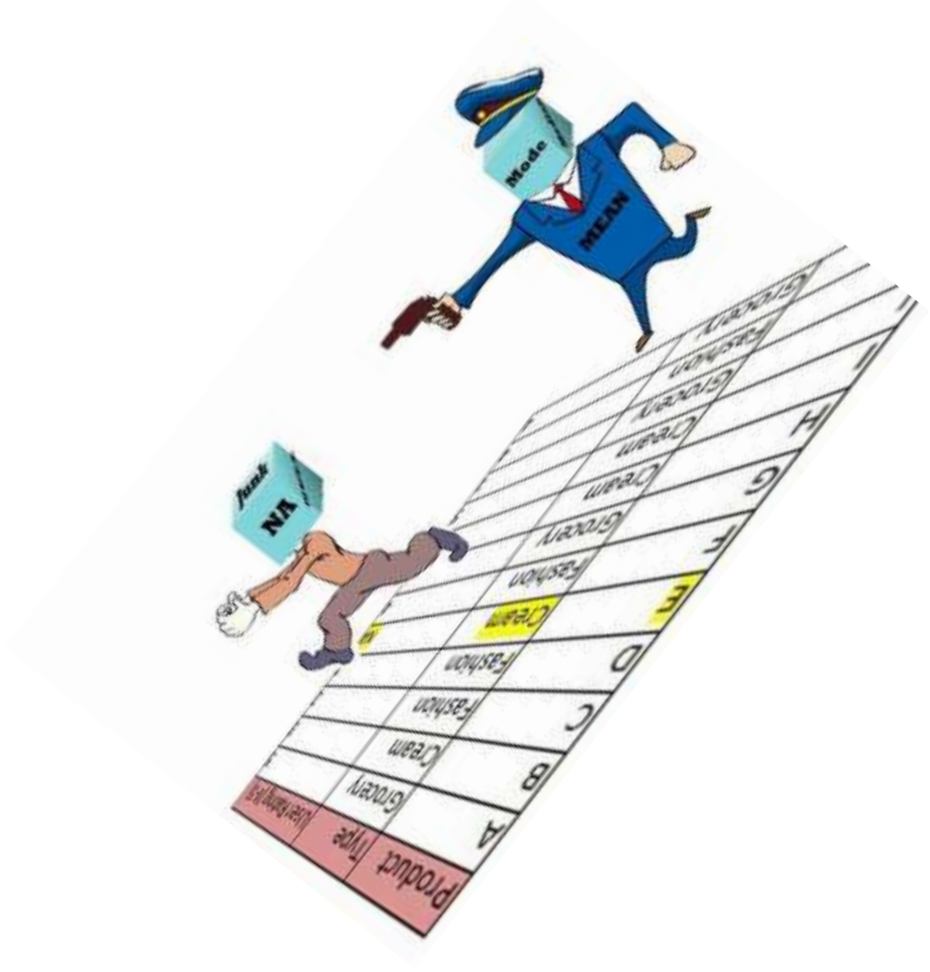
BASIC DATA PREPROCESSING: MISSING VALUES

- **Missing Data Identification:** Before processing missing data, it's crucial to identify which variables contain missing values and to what extent.
  - Missing data can occur due to errors in data collection, transmission, or entry.
- **Dropping Missing Data:** In some cases, especially when the proportion of missing values is large or they are unimportant to the analysis, missing rows or columns may be removed entirely. This method is straightforward but may result in data loss.

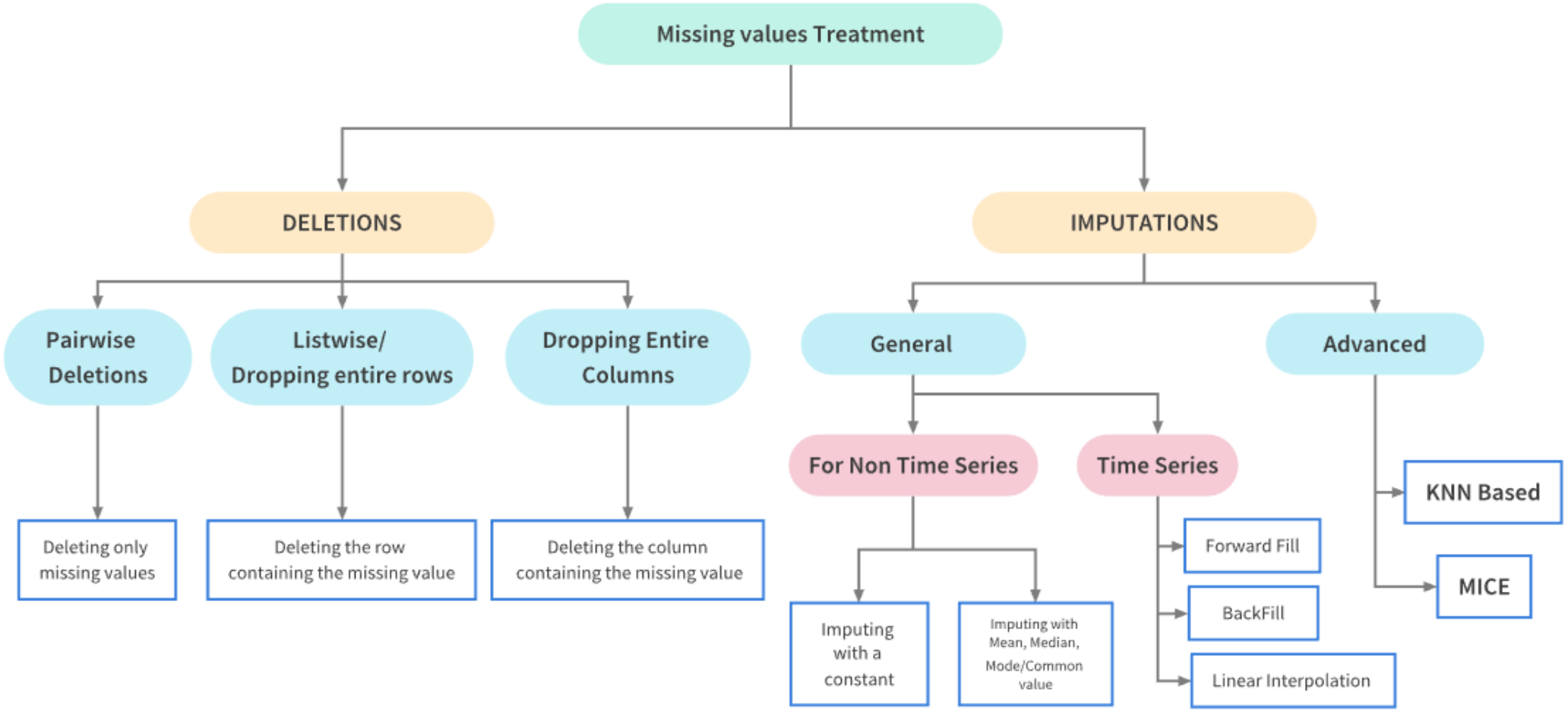


BASIC DATA PREPROCESSING: MISSING VALUES

- **Advanced Imputation:**
  - **More complex methods**, such as using algorithms like K-Nearest Neighbors (KNN) or regression models, predict missing values based on the relationships with other variables. Or using machine learning based methods such as: (KNNI, WKNNI, KMI, FKMI, SVMI, SVDI…etc.).
  - Or using probabilistic based methods such as: (EM algorithm).
  - More recent techniques have appeared such as GMC, MLP, ANN, SOM, Bayesian models …etc.
  - **Indicator Variables**: For categorical data, a new category labeled "Missing" can be introduced. This approach ensures no data is lost while accounting for missing values.

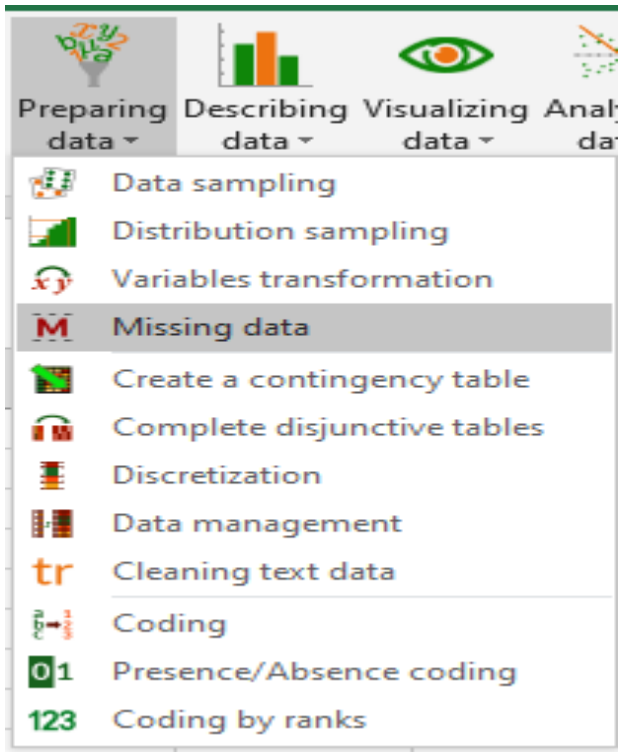


BASIC DATA PREPROCESSING: MISSING VALUES



BASIC DATA PREPROCESSING: MISSING VALUES

- Surprisingly, Excel offers some magic !



BASIC DATA PREPROCESSING: MISSING VALUES

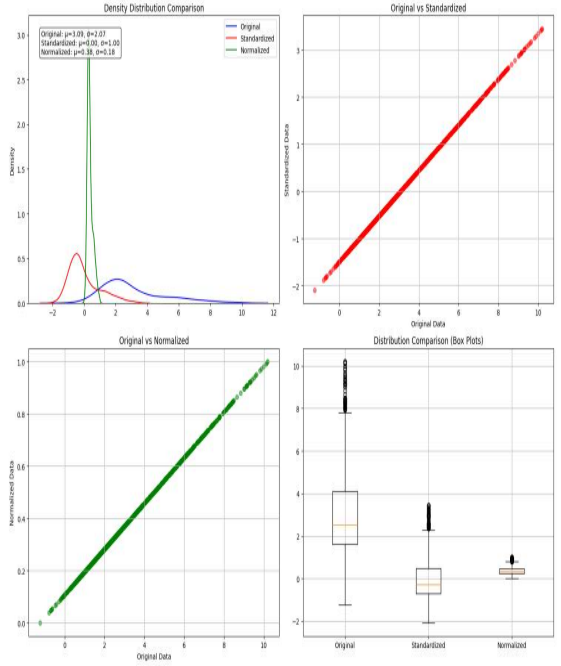
- Missing value imputation using Weka:
  - weka.filters.unsupervised.attribute.MissingValuesImputation - for imputing missing values
  - weka.filters.unsupervised.attribute.MissingValuesInjection - for injecting missing values

BASIC DATA PREPROCESSING

- Common Techniques of data preprocessing include methods to adjust data ranges, scales, and structures.
- **Standardization** and **normalization** are among the most widely used methods for these adjustments.
- Such techniques help to address inconsistencies, missing values, and varying scales within data.
  - Hence, it enhances the model performance and ensures more reliable, accurate outcomes.

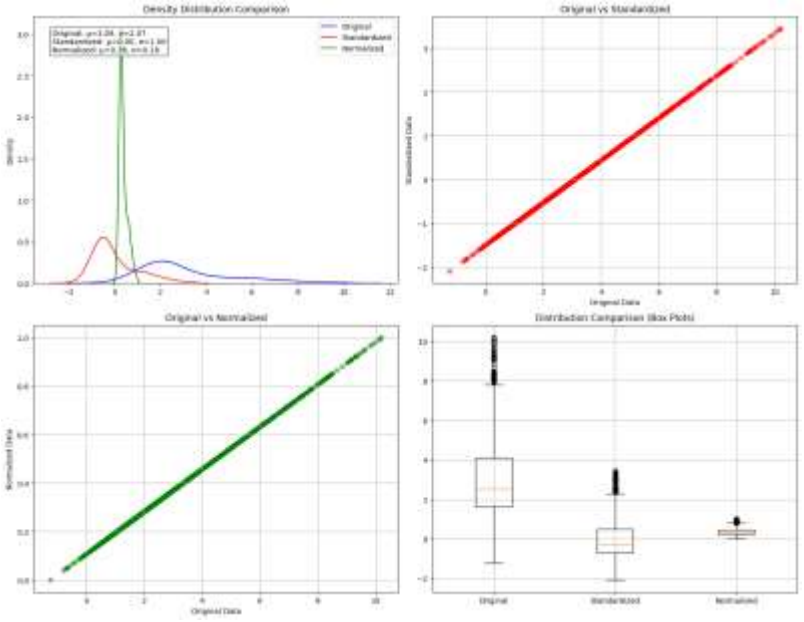
BASIC DATA PREPROCESSING: NORMALIZATION AND STANDARDIZATION

- Two key techniques for preparing data: **Normalization** and **Standardization**.
- **Normalization** adjusts the data scale to a specific range, typically [0,1] or [-1,1].
  - Formula:
    - $X' = (X - Xmin)/(Xmax - Xmin)$
  - **Purpose**: Keeps values within a fixed range to manage varying data scales.
  - Commonly used in algorithms **like k-Nearest Neighbors and Neural Networks**, where feature scaling impacts model performance.



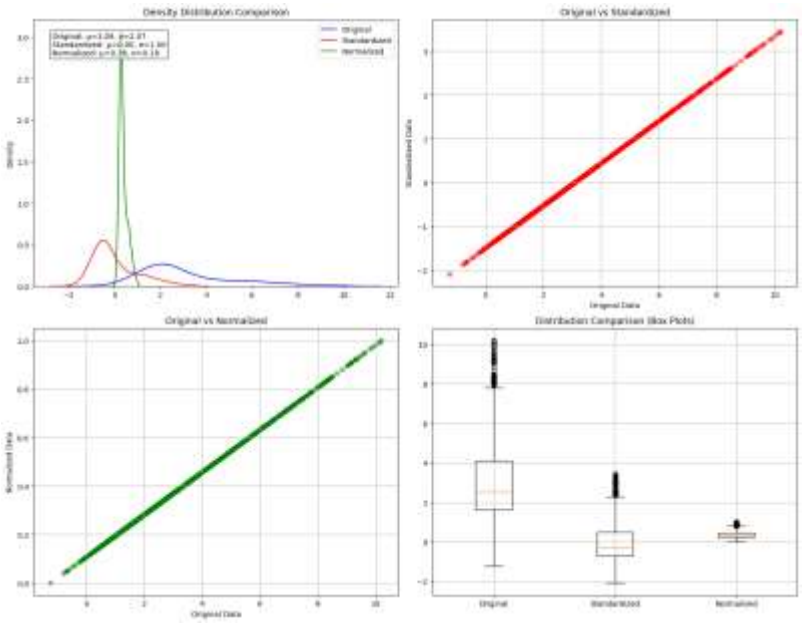
BASIC DATA PREPROCESSING: NORMALIZATION AND STANDARDIZATION

- **Standardization** transforms data to have a mean of 0 and a standard deviation of 1, often resulting in a normal (Gaussian) distribution.
  - Formula:
    - $X' = (X - \mu)/\sigma$
  - Purpose:
    - Centers data around 0, allowing different features to contribute equally to the analysis.
  - Preferred in algorithms like **Support Vector Machines and Principal Component Analysis**, where data distribution affects model outcome.

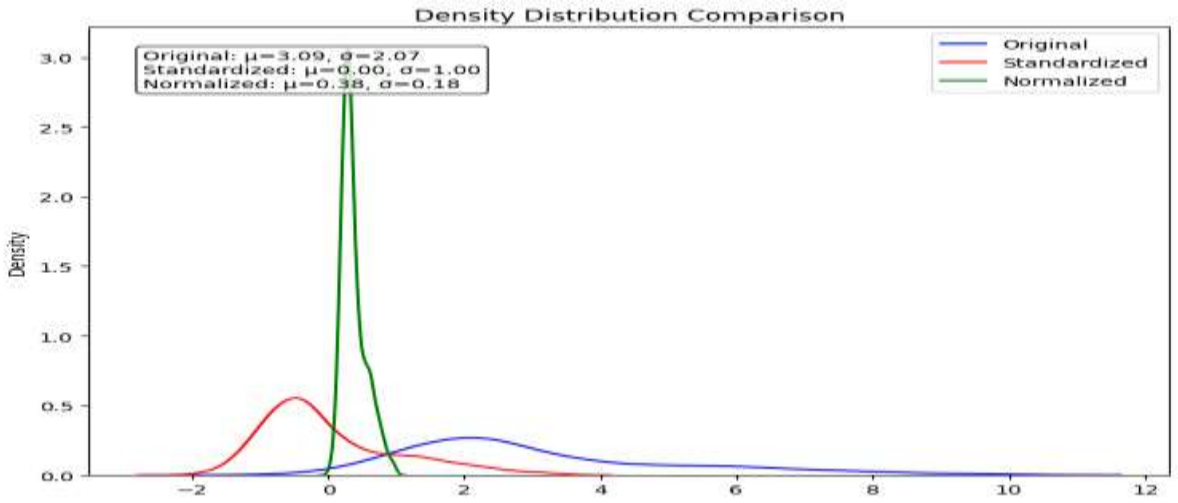


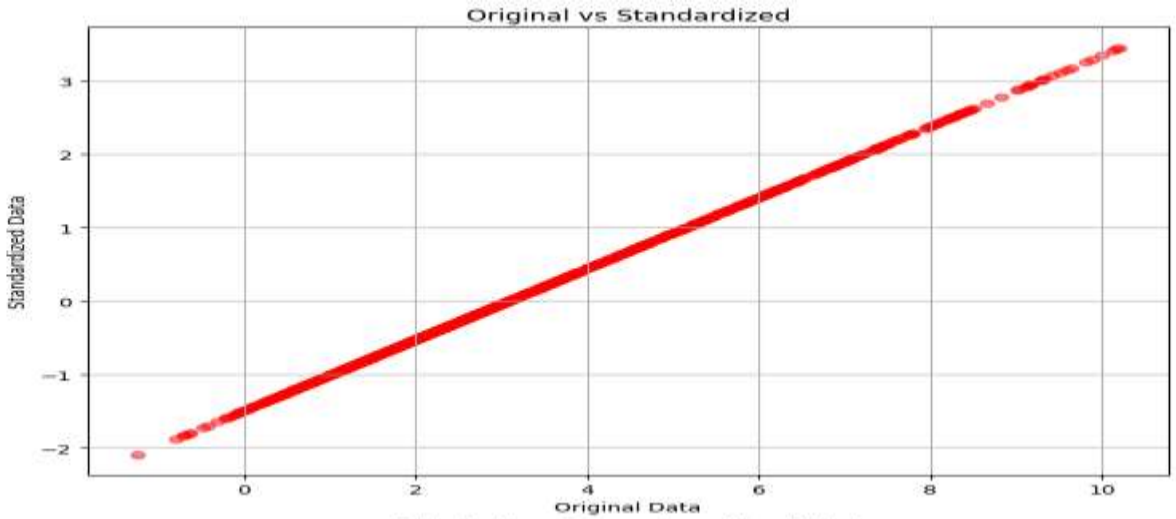
BASIC DATA PREPROCESSING: NORMALIZATION AND STANDARDIZATION

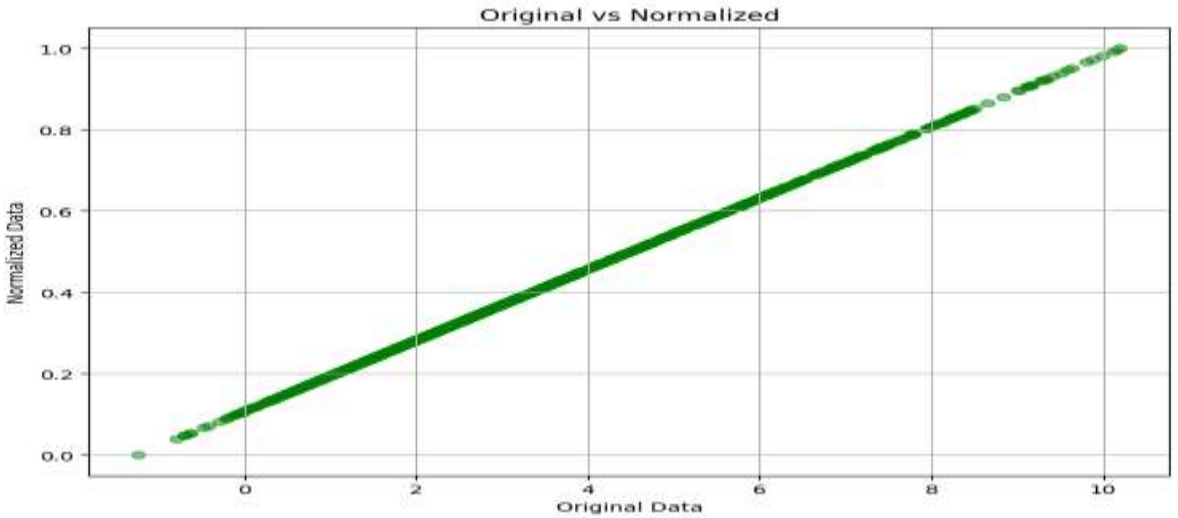
- Use **Normalization**:
  - When data does not have a Gaussian distribution.
  - For models sensitive to feature magnitude, like k-NN or Neural Networks.
- Use **Standardization**:
  - When data follows a Gaussian (normal) distribution.
  - For distance-based algorithms sensitive to data distribution, like SVMs and PCA.

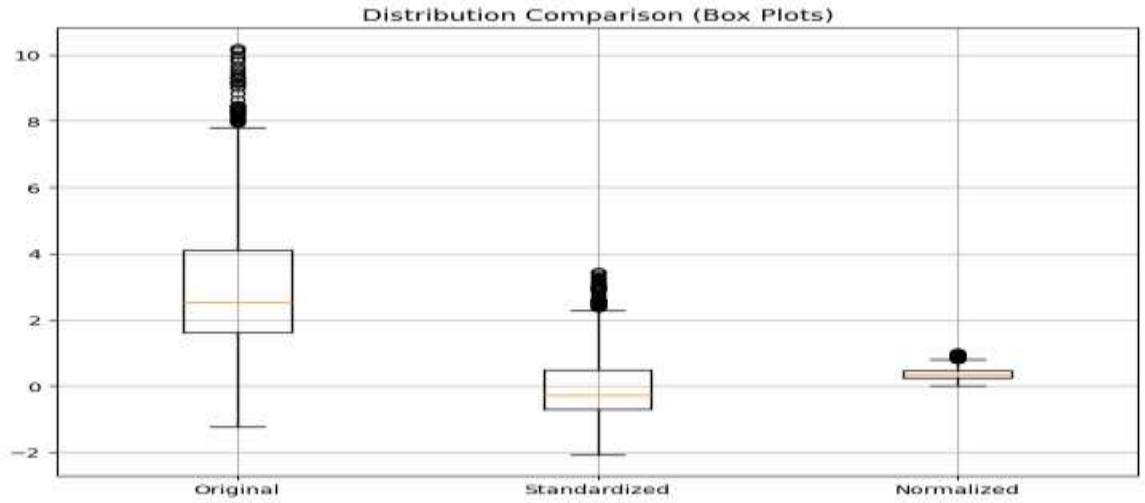


BASIC DATA PREPROCESSING: NORMALIZATION AND STANDARDIZATION









FEATURE ENGINEERING

- **Feature Engineering** is the process of creating, transforming, or selecting data features to improve model performance.
  - A crucial step in data science and machine learning pipelines, as good features can enhance accuracy, reduce complexity, and improve interpretability.
- **Types of Feature Engineering Techniques**
  - **Feature Selection**: Selecting the most relevant features from existing data.
  - **Feature Extraction**: Creating new features by transforming or combining existing ones.
  - **Feature Transformation**: Modifying features for better model compatibility (e.g., scaling, encoding).

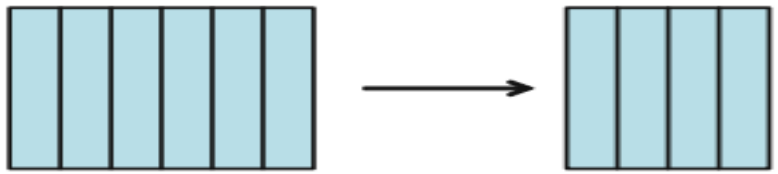
FEATURE ENGINEERING

- How do I reduce the dimensionality of data?—Feature Selection (FS).
- How do I remove redundant and/or conflictive examples?—Instance Selection (IS).
- How do I simplify the domain of an attribute?—Discretization.
- How do I fill in gaps in data?—Feature Extraction and/or Instance Generation.

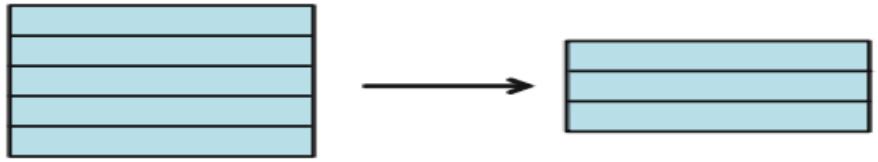
FEATURE ENGINEERING

- **Feature Selection:**
  - **Goal**:
    - Identify the most informative features that significantly impact the target variable.
  - **Benefits**:
    - Reduces dimensionality, minimizes overfitting, and enhances model interpretability.
  - **Basic Algorithms:**
    - Filter Methods:
      - Select features based on statistical tests (e.g., Chi-Square, ANOVA).
    - **Wrapper Methods:**
      - Evaluate subsets of features based on model performance (e.g., Recursive Feature Elimination).
    - **Embedded Methods:**
      - Feature selection integrated into model training (e.g., Lasso Regression).
  - **Example**
    - In a dataset with hundreds of features, selecting only the top 10 correlated with the target variable.

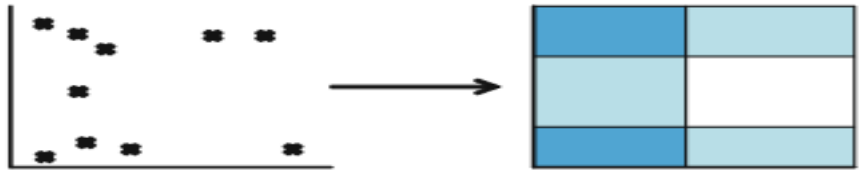
FEATURE ENGINEERING : FEATURE SELECTION



Instance Selection



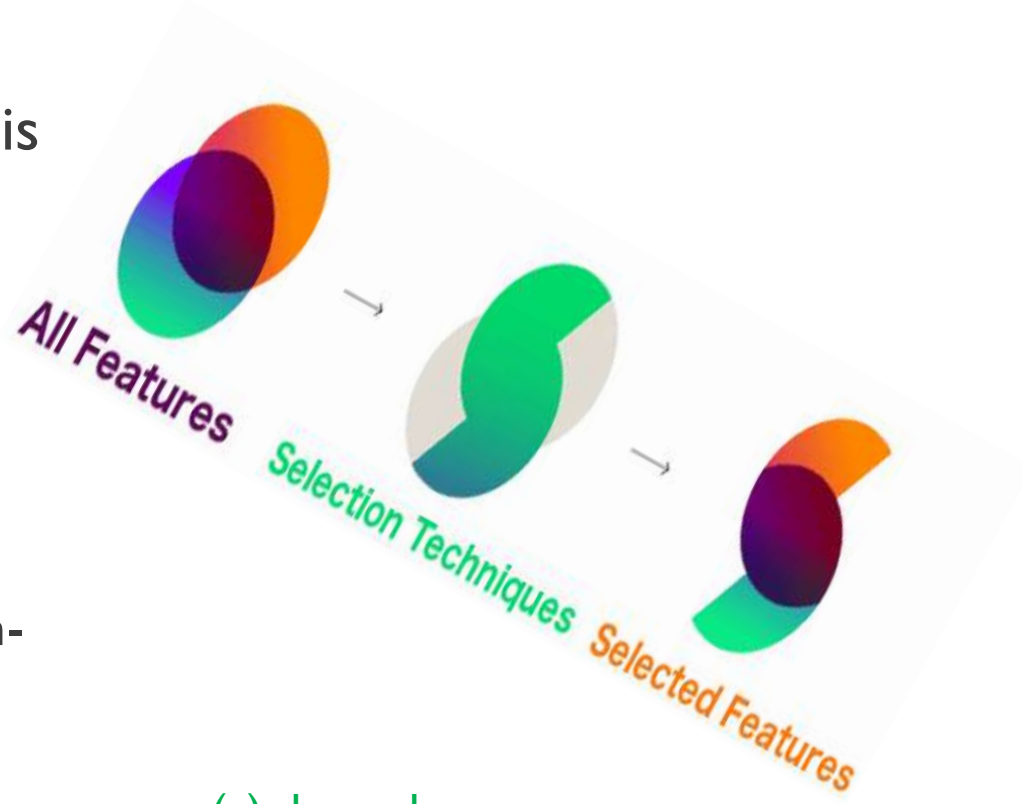
Discretization



- ✓ Speed up the processing of the DM algorithm.
- ✓ Improve data quality.
- ✓ Increase the performance of the DM algorithm.
- ✓ Make the results easier to understand

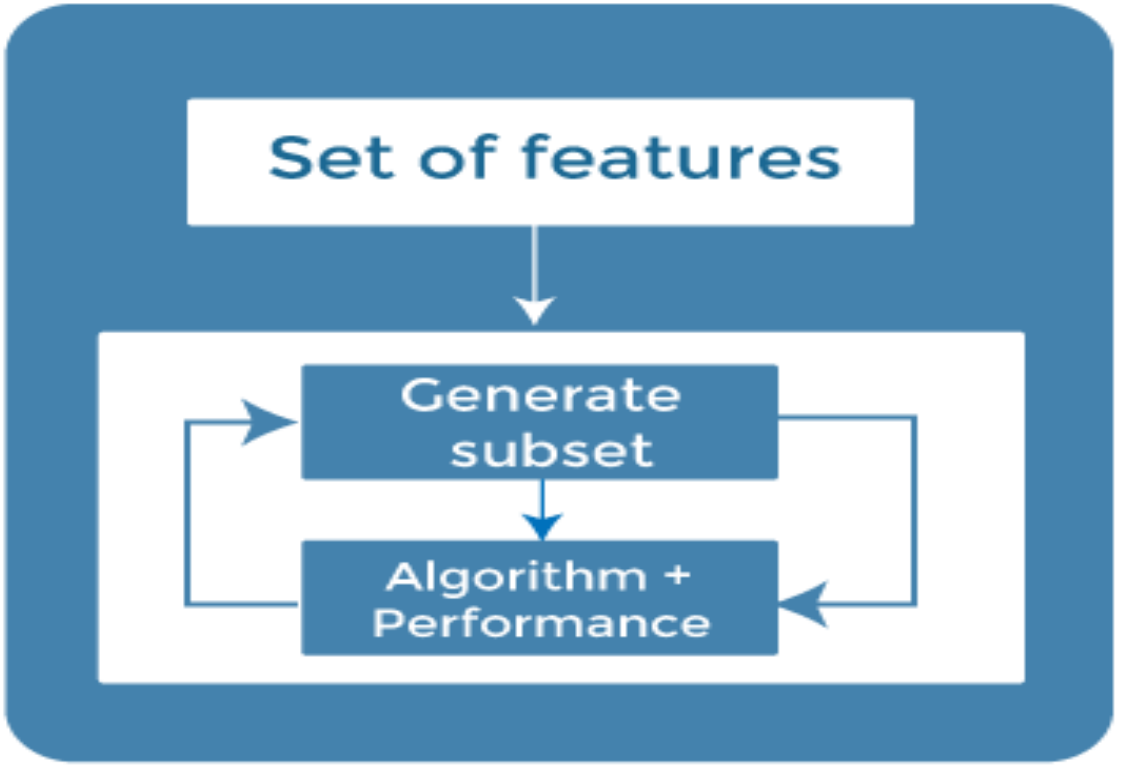
FEATURE ENGINEERING : FEATURE SELECTION

- **Feature selection (i.e., data condensation.)** is reducing the number of attributes as much as possible to speed up data processing, and enable visualization. This operation happens by removing redundant, irrelevant features.
  - Were mainly used for classification purposes.
  - Using: (a) exhaustive search, (b) heuristic search, (c) non-deterministic search.



➢ Selection criteria based on: (a) information measures, (b) distance measures, (c) dependency measures, (d) accuracy measures, consistency measures.

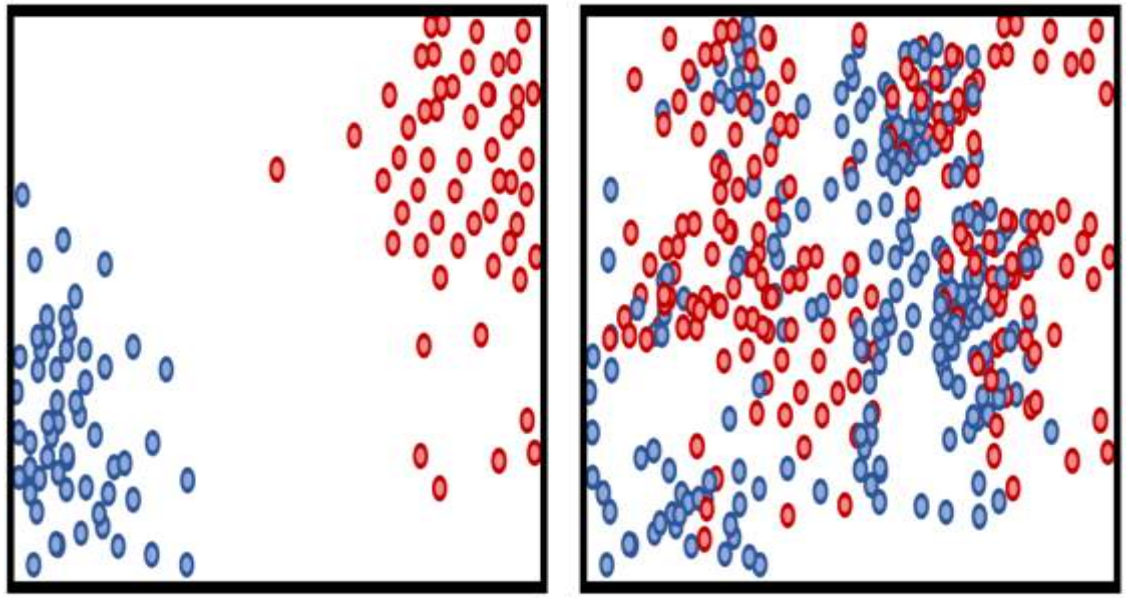
FEATURE ENGINEERING : FEATURE SELECTION



FEATURE ENGINEERING : FEATURE SELECTION

- *Feature selection using the information gain*
  - $H(X) = -\sum pi * \log2(pi)$
  - $H(X|Y) = \sum P(X=v) H(Y|X=v)$
  - IG(Y|X) = H(X) – H(X|Y)

- where:
  - Pi is the probability of the class i.
  - H(X) is the entropy of the target feature (class feature).
  - H(X|Y) is the conditional entropy of X in regard to the feature Y.
  - IG(Y|X) is the information gain of the feature Y by considering the target class.



FEATURE ENGINEERING : EXAMPLE OF FEATURE SELECTION

Row	Age_Group	Income_Level	Internet_Usage	Buys_Product	H(B)
1	Young	High	Low	No	-[4/8log(4/8)+4/8log(4/8)] = 0.30
2	Middle	Low	High	Yes	
3	Senior	Medium	Medium	No	
4	Young	Low	High	Yes	
5	Middle	High	Medium	No	
6	Senior	Low	Low	No	
7	Young	Medium	High	Yes	
8	Middle	Medium	Medium	Yes	

FEATURE ENGINEERING : EXAMPLE OF FEATURE SELECTION

- We calculate the information gain for the Age group, by splitting the data according to all unique values of the feature, and found the value 0.27

ID	P(v)	Age_Group	Buys_Product	H(B|a)	H(B|A)	IG(A|B)
1	3/8=0.375	Young	No	-[1/3*log(1/3)+2/3*log(2/3)] = 0.2764	0.2764*0.375+0.2764*0.375+0*0.25= 0.0312	0.3-0.0312 = 0.27
4		Young	Yes			
7		Young	Yes			
2	3/8=0.375	Middle	Yes	-[1/3*log(1/3)+2/3*log(2/3)]= 0.2764		
5		Middle	No			
8		Middle	Yes			
3	2/8=0.25	Senior	No	-[2/2*log(2/2)] =0		
6		Senior	No			

FEATURE ENGINEERING : EXAMPLE OF FEATURE SELECTION

- We calculate the information gain for the Income Level, by splitting the data according to all unique values of the feature, and found the value 0.27

ID	P(v)	Income_Level	Buys_Product	H(B|i)	H(B|I)	IG(I|B)
2	3/8 =0.375	Low	Yes	-[2/3*log(2/3)+1/3*log(1/3)] = 0.2764	0.375*0.2764+0.2764*0.375+0.25*0= 0.0312	0.3-0.0312 = 0.27
4		Low	No			
6		Low	Yes			
3	3/8=0.375	Medium	No	-[1/3*log(1/3)+2/3*log(2/3)]= 0.2764		
7		Medium	Yes			
8		Medium	Yes			
1	2/8= 0.25	High	No	-2/2*log(2/2)=0		
5		High	No			

FEATURE ENGINEERING : EXAMPLE OF FEATURE SELECTION

- We calculate the information gain for the Internet usage, by splitting the data according to all unique values of the feature, and found the value 0.2.
- By considering the previous calculations, we conclude that internet usage is the less important feature in comparison to the other ones.

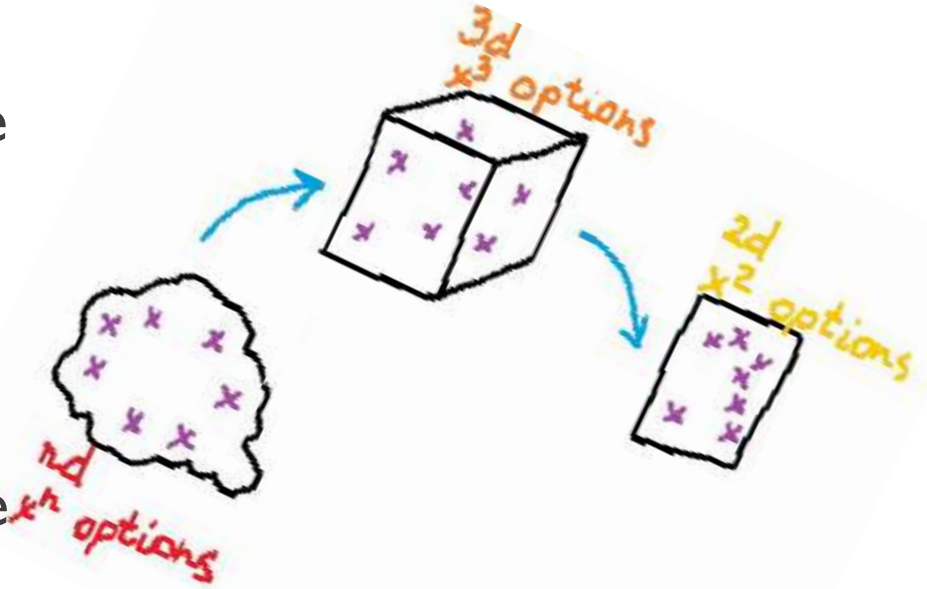
ID	P(v)	Internet_Usage	Buys_Product	P(B|i)	P(B|I)	IG(I|B)
1	2/8= 0.25	Low	No	2/2*log(2/2) = 0	0.25*0+0.375*0.2764+0.375*0=0.10	0.3-0.1=0.2
6		Low	No			
3	3/8= 0.375	Medium	No	[2/3*log(2/3)+1/3*log(1/3)] = 0.2764		
5		Medium	No			
8		Medium	Yes			
2	3/8= 0.375	High	Yes	3/3*log(3/3) = 0		
4		High	Yes			
7		High	Yes			

FEATURE ENGINEERING: FEATURE EXTRACTION

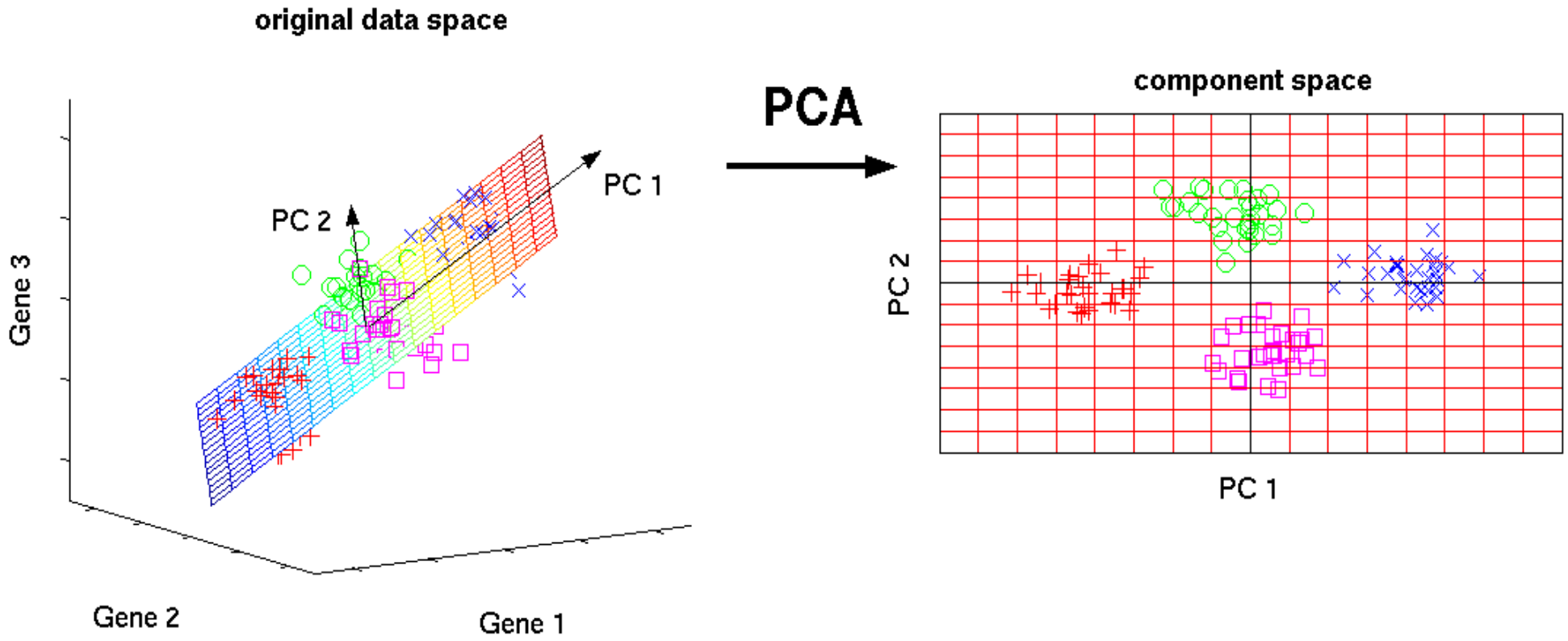
- **Feature Extraction:**
  - **Goal**:
    - Create new features that retain important information while simplifying the dataset.
  - **Benefits**:
    - Reduces dimensionality and improves performance, especially in high-dimensional data.
  - **Basic Algorithms**:
    - Principal Component Analysis (PCA):
      - Projects data into new axes to reduce dimensionality.
    - Independent Component Analysis (ICA):
      - Separates a multivariate signal into independent components.
    - t-SNE (t-Distributed Stochastic Neighbor Embedding):
      - Reduces high-dimensional data for visualization.
  - **Example**:
    - Applying PCA to transform a 100-dimensional dataset into a 5-dimensional representation, capturing most of the data's variance.

FEATURE ENGINEERING: FEATURE EXTRACTION

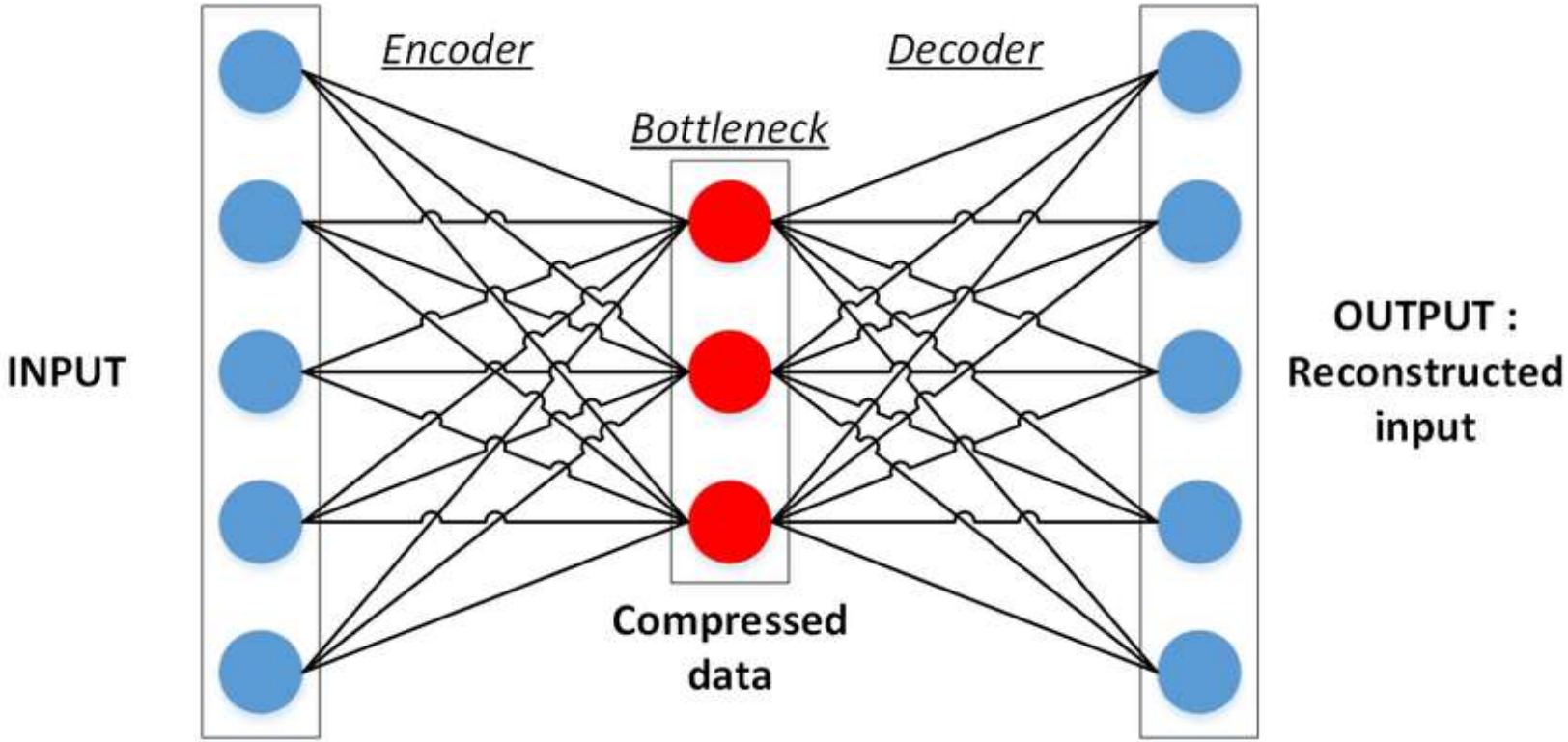
- **Feature Extraction/Instance Generation:** In the contrary of feature selection, in this operation original data contribute in the generation of the extracted features by aggregation, merging, or combinations following some specific functions (e.g., linear combinations, polynomial …etc.). Among these techniques we enumerate PCA, factor analysis,..etc.



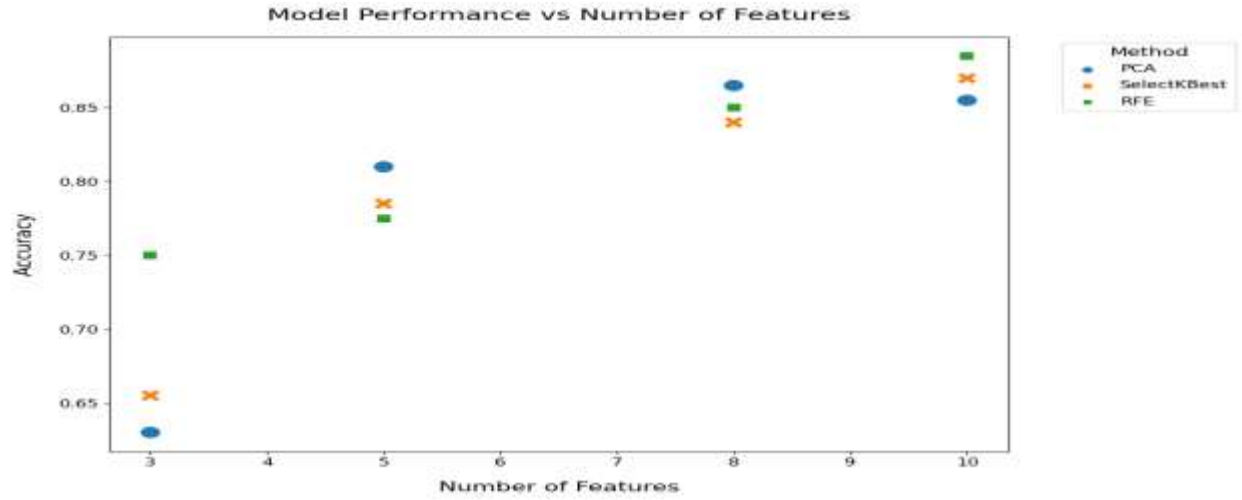
FEATURE ENGINEERING: FEATURE EXTRACTION

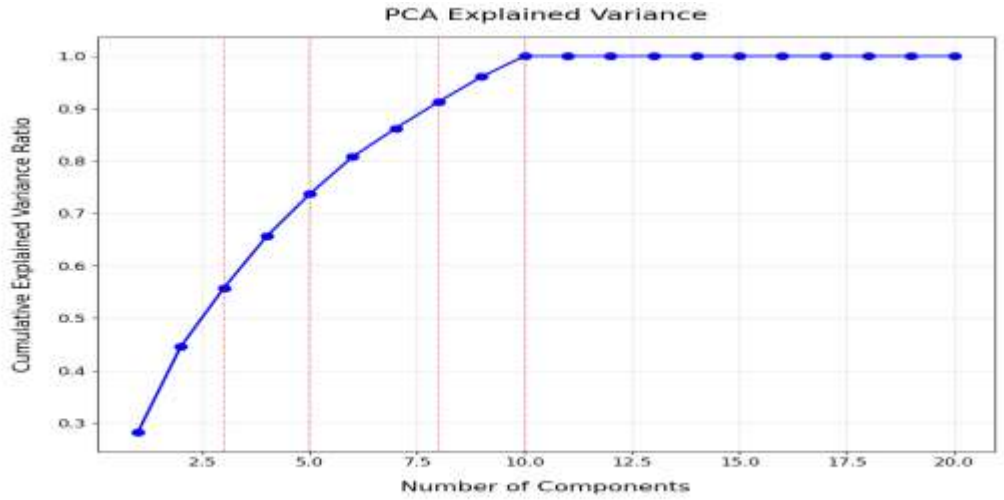


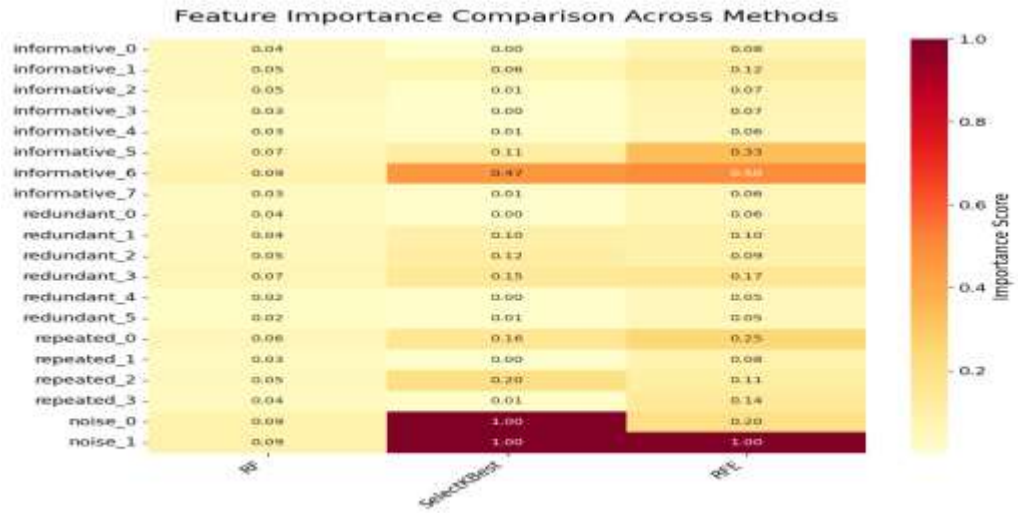
FEATURE ENGINEERING: FEATURE EXTRACTION

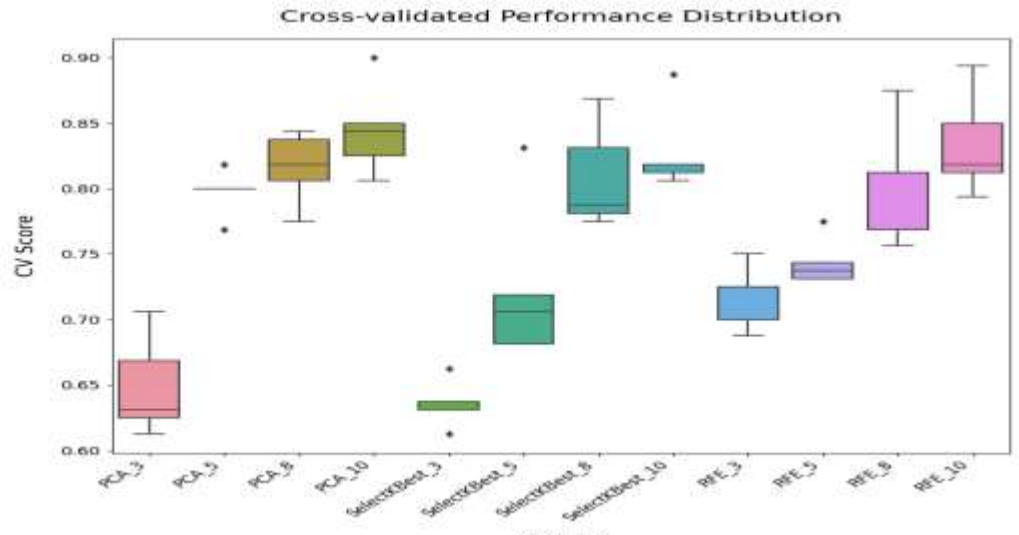


FEATURE ENGINEERING: COMPARISON









FEATURE ENGINEERING: INSTANCE SELECTION

- **Instance selection:** Many techniques have appeared under this category of operation holding the name **sampling**, it consists of randomly (i.e., or using some heuristics) selecting a subset of records for the data mining process. This operation should be performed very carefully to avoid the **bias problem**.

TREE-BASED FEATURE IMPORTANCE ALGORITHM

- **Tree-based models**, like Decision Trees, Random Forests, and Gradient Boosting, automatically rank features based on their importance to predictions.
- Feature importance tells us which features contribute most to the predictive power of the model.

TREE-BASED FEATURE IMPORTANCE ALGORITHM

- **Splitting**: Tree models split data at each node based on features that best separate classes or values.
- **Decrease in Impurity**: Each split reduces "impurity" (e.g., Gini impurity or variance).
- **Importance Score**: The feature that reduces impurity most often across the tree (or trees) gets a higher importance score.
- **Example**: In a Random Forest, the importance of a feature is the average impurity decrease across all trees where it was used for a split.

TREE-BASED FEATURE IMPORTANCE ALGORITHM

- **Limitations:**
  - **Bias towards Numerical Features**: Trees may favor features with more unique values (e.g., continuous variables).
  - **Doesn't Capture Interactions**: Individual feature importance might miss important feature interactions.
  - **Model-Specific:** Importance is based on the model used; it may differ across different algorithms.

TREE-BASED FEATURE IMPORTANCE ALGORITHM

- **Example**:

Hours_Studied	Attendance	Parental_Support	Pass_Fail
3	0.8	Yes	Fail
5	0.9	Yes	Pass
2	0.6	No	Fail
4	0.7	No	Fail
6	1.0	Yes	Pass

TREE-BASED FEATURE IMPORTANCE ALGORITHM

- **1. Building the Decision Tree**
  - The Decision Tree algorithm first **splits the data** on features that reduce impurity the most.
  - Impurity measures how mixed the target classes are at each node.
  - For classification, we often use **Gini Impurity** or **Entropy**.
  - The algorithm tries to split on **Hours_Studied**, **Attendance**, and **Parental_Support** and chooses the split that best separates the students who passed from those who failed.
- **2. Splitting to Reduce Impurity**
  - Assume that **Hours_Studied** provides the best first split. For example, splitting at **Hours_Studied** $\geq$ **4** might separate many passing students from failing ones.
  - At each split, the tree evaluates how much the impurity decreases when using each feature to split.
  - The **total decrease in impurity** is summed for each feature across the tree. The higher the total decrease for a feature, the more important that feature is considered.

TREE-BASED FEATURE IMPORTANCE ALGORITHM

- **3. Calculating Feature Importance Scores**
  - After the tree is built, we calculate a **feature importance score** for each feature.
  - The importance of a feature is calculated as:
    - Feature Importance=Total Impurity Decrease from this Feature / Total Impurity Decrease from all Features
- **4. Example of Feature Importance Calculation**
- Suppose our decision tree split as follows:
  - **Hours_Studied** reduced impurity by 0.30.
  - **Attendance** reduced impurity by 0.20.
  - **Parental_Support** reduced impurity by 0.05.
- The feature importance for each feature would be:
  - **Hours_Studied**: 0.30/(0.30+0.20+0.05)=0.6 (60%)
  - **Attendance**: 0.20/(0.30+0.20+0.05)=0.33 (33%)
  - **Parental_Support**: 0.05/(0.30+0.20+0.05)=0.1 (10%)

TREE-BASED FEATURE IMPORTANCE ALGORITHM

- **5. Resulting Feature Importance**
  - Based on the above calculations, the feature importance scores for each feature are:
    - **Hours_Studied**: 60%
    - **Attendance**: 33%
    - **Parental_Support**: 10%
  - **Interpretation**: **Hours_Studied** is the most important feature for predicting Pass/Fail status, followed by **Attendance**, and lastly **Parental_Support**.

TREE-BASED FEATURE IMPORTANCE ALGORITHM

- **Example of Calculating Gini Impurity**:
  - We calculate the gini impurity by considering the target class as follows:

$$\text{Gini Impurity} = 1 - \sum_{i=1}^{C} p_i^2$$

  - where:
    - C is the number of classes (in this case, 2: Pass and Fail),
    - pi is the proportion of examples in the node that belong to class i.
- The **lower the Gini Impurity**, the purer the node is, meaning it has a higher concentration of a single class (e.g., mostly "Pass" or mostly "Fail").
- So, the probabilities for each class are:
  - P(Pass) =2/5=0.4
  - P(Fail)=3/5=0.6
- Now, we plug these probabilities into the Gini formula:
  - Gini (before split) = 0.48

TREE-BASED FEATURE IMPORTANCE ALGORITHM

- Now, let's say we split the dataset on **Hours_Studied** at the threshold of **4 hours**, creating two child nodes:
  - **Node 1:** (Hours_Studied < 4):
    - Pass = 0, Fail = 3
    - P(Pass) = 0 and P(Fail) =1
    - **Gini Impurity for Node 1**: Gini=1−(02+12)=1−1=0\text{Gini} = 1 - (0^2 + 1^2) = 1 - 1 = 0Gini=1−(02+12)=1−1=0
  - **Node 2:** (Hours_Studied ≥ 4):
    - Pass = 2, Fail = 0
    - P(Pass)=1 and P(Pass) =1
    - **Gini Impurity for Node 2**: Gini=1−(1^2+0^2)= 1−1=0
    - Since both nodes have Gini Impurity values of **0**,

  , the split on Hours_Studied at 4 hours has created **pure nodes** (where each node contains only one class).

TREE-BASED FEATURE IMPORTANCE ALGORITHM

- To measure the **impurity reduction**, we need to calculate the **weighted average Gini Impurity after the split** and compare it to the Gini Impurity before the split.
- **Weighting each node's Gini Impurity by the number of samples**:
  - **Node 1**: 3 samples with Gini = 0
  - **Node 2**: 2 samples with Gini = 0
- **Weighted Gini After Split**:
  - Weighted Gini After Split=3/5×0+2/5×0=0
- **Impurity Reduction**:
  - Impurity Reduction=Gini Before Split−Weighted Gini After Split =0.48−0=0.48
- This **impurity reduction (0.48)** represents the **importance of the feature** "Hours_Studied" in this split.

TREE-BASED FEATURE IMPORTANCE ALGORITHM

For a given feature f, the importance score is computed as:

$$\text{Feature Importance (f)} = \sum_{t \in T_f} \frac{n_t}{N} \times \Delta\text{Impurity}_t$$

Where:

- T_f represents the set of nodes where feature f was used for a split,
- n_t is the number of samples in node t,
- N is the total number of samples in the dataset,
- $\Delta\text{Impurity}_t$ is the impurity reduction at node t from the split involving feature f.

CONCLUSION

- Exploratory Data Analysis (EDA) is a critical step in the data science process, serving as the foundation for building robust models and generating insights.
- Visualizing and summarizing the data helps identify underlying patterns, detect anomalies, and uncover relationships between variables.
- EDA also provides a clear understanding of data distributions and missing values, guiding decisions on data cleaning and preprocessing.
- Ultimately, EDA empowers data scientists to make informed decisions, improve model accuracy, and drive actionable business outcomes.

REFERENCES

- **Zheng, A. X., & Casari, A.** (2018). *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. O'Reilly Media.
- **Kelleher, J. D., & Tierney, B.** (2018). *Data Science: An Introduction*. Springer.