

Machine Learning for Finance

Prepared by Dr. Diaa Salama

FNT633
Data Preprocessing , Data Normalization

Lecture 4





مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY

Machine Learning for Finance

Lecture

FNT633

Data Preprocessing , Data Normalization

Prepared by
Dr. Diao Salama

www.midocean.ae

- Data
- Data Preparation (Preprocessing)
- Data Integration and Normalization

Week

05

Data

■ OBJECTS AND ATTRIBUTES

- data object represents an entity
- Also, sample, example, instance, data point, or object (in a DB : Data Tuple)-
- e.g. customers, students, patients, books

■ Attribute

- data field, representing a characteristic or feature of a data object
- Also noun attribute, dimension, feature, and variable (DB and DM
- DWs, ML. Statistics)
- e.g. name, age, salary, gender. grade.

■ Attribute (feature) vector

- - A set of attributes that describe an object

What is Data?

- Collection of data objects and their attributes
- An attribute is a property or characteristic of an object
 - Examples: eye color of a person, temperature, etc.
 - Attribute is also known as variable, field, characteristic, dimension, or feature
- A collection of attributes describe an object
 - Object is also known as record, point, case, sample, entity, or instance

Objects

Attributes

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- **NOMINAL ATTRIBUTES**
- Symbol or names of things
- Each value represents category, code, or state
 - also referred to as categorical
 - e.g. hair color, marital status, customer ID
 - Possible to be represented as numbers (coding)

Attributes Types : (Binary Attributes)

- Nominal with only two values representing two states or categories: 0 or 1 (absent or present)
- Also, Boolean (true or false)
- Qualitative
- Symmetric:
 - both states are equally valuable and have the same weight-
 - e.g. gender
- Asymmetric-
 - states are not equally important
 - e.g. medical test outcomes

- * Qualitative*
 - Values have a meaningful order or ranking, but magnitude between successive values is not known
 - e.g. professional rank, grade, customer satisfaction-
 - Useful for, data reduction of numerical attributes

Attributes Types :(Numeric Attributes)

- Quantitative
- Interval-scaled:*

 - measured on a scale of equal-size units-
 - e.g. temperature, year-
 - Do not have a true zero point-
 - Not possible to be expressed as multiples

- Ratio-scaled:-

 - have a true zero point-
 - A value can be expressed as a multiple of another-
 - e.g. years of experience, weight, salary

Discrete and Continuous Attributes

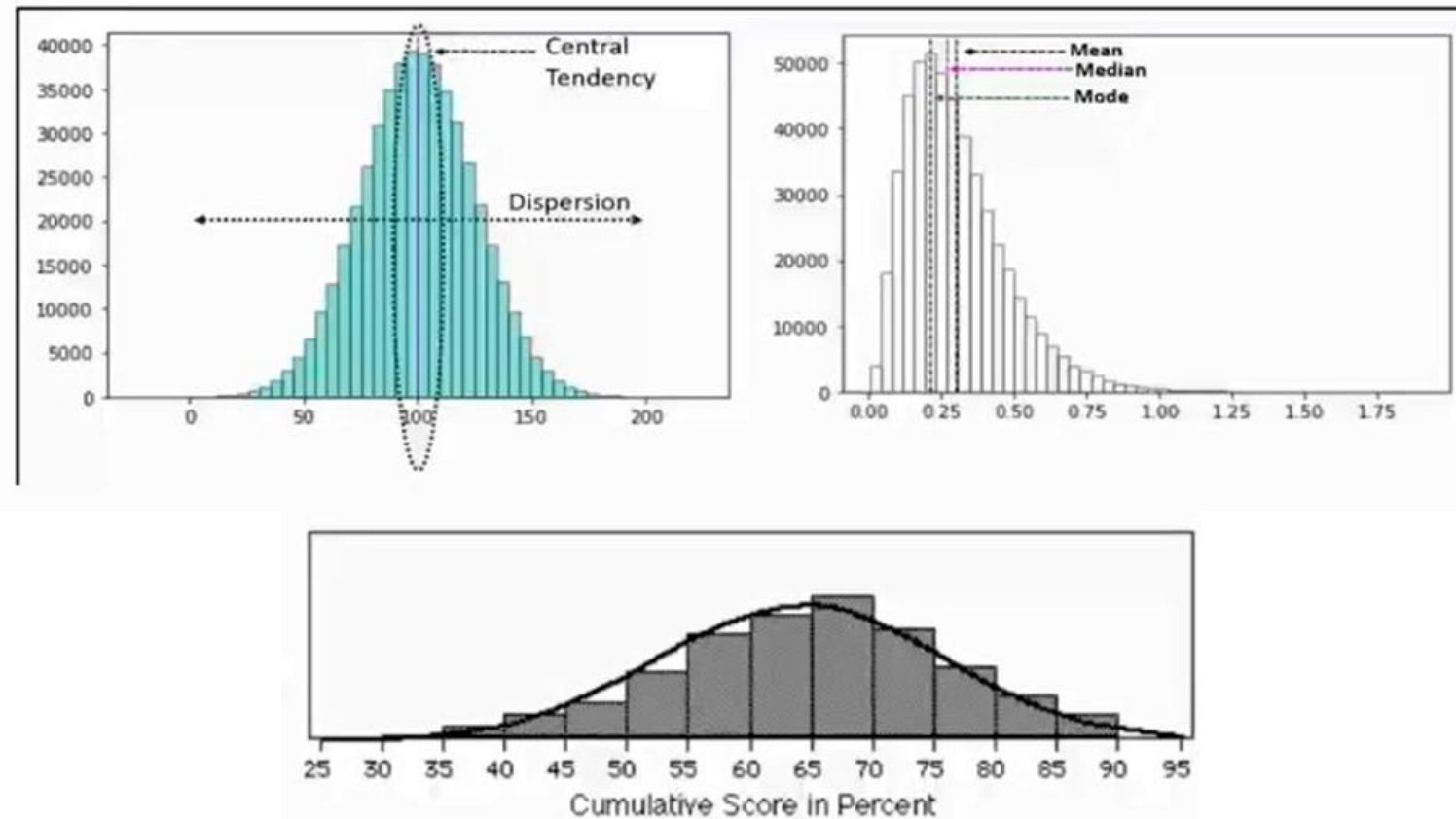
- Discrete Attribute:
 - has a finite or countably infinite set of values, integers or otherwise
 - - e.g. hair color, smoker, medical test, Customer_ID-
 - Customer_ID is countably infinite à infinite values but one-to-one correspondence with natural numbers*
- If an attribute is not discrete, it's continuous-
 - e.g. height, weight, age

Discrete and Continuous Attributes

- Discrete Attribute
 - Has only a finite or countably infinite set of values
 - Examples: zip codes, counts, or the set of words in a collection of documents
 - Often represented as integer variables.
 - Note: binary attributes are a special case of discrete attributes
- Continuous Attribute
 - Has real numbers as attribute values
 - Examples: temperature, height, or weight.
 - Practically, real values can only be measured and represented using a finite number of digits.
 - Continuous attributes are typically represented as floating-point variables.

Basic Statistical of data

To Understand This Distribution:



Measuring Central Tendency (Mean)

For N observations of numerical variable X: $x_1, x_2, \dots, x_N$

Mean: or average of values

$$\bar{x} = \frac{\sum_{i=1}^N x_i}{N} = \frac{x_1 + x_2 + \dots + x_N}{N}$$

Weighted Average : a weight is associated with each value

$$\bar{x} = \frac{\sum_{i=1}^N w_i x_i}{N} = \frac{w_1 x_1 + w_2 x_2 + \dots + w_N x_N}{N}$$

Problem: sensitivity to outlier values

- e.g. mean salary, mean student score

Solution: Trimmed mean à chop off extreme values at both ends

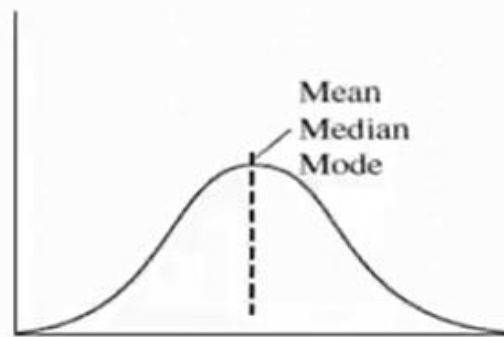
Measuring Central Tendency (**Median ,Mode**)

- Median
 - middle value in set of ordered values*
 - N is odd à median is middle value of ordered set*
 - N is even à median is not unique à average of two middlemost values*
 - Expensive to compute for large # of observations
- Mode:
 - value that occurs most frequently in the attribute values*
 - Works for both qualitative and quantitative attributes .*
 - Data can be unimodal, bimodal, or trimodal - no mode?

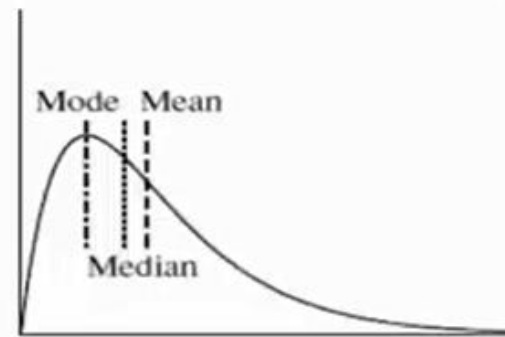
Example 1 : Central Tendency

■ Salary (in thousands of dollars), shown in increasing order: 30, 36, 47, 50, 52, 52, 56, 60, 63, 70, 70, 110

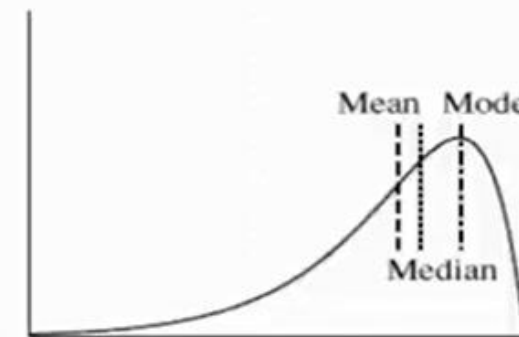
- Mean = 58,000
- Median = 54,000
- Mode = 52,000 and 70,000 – bimodal



(a) Symmetric data



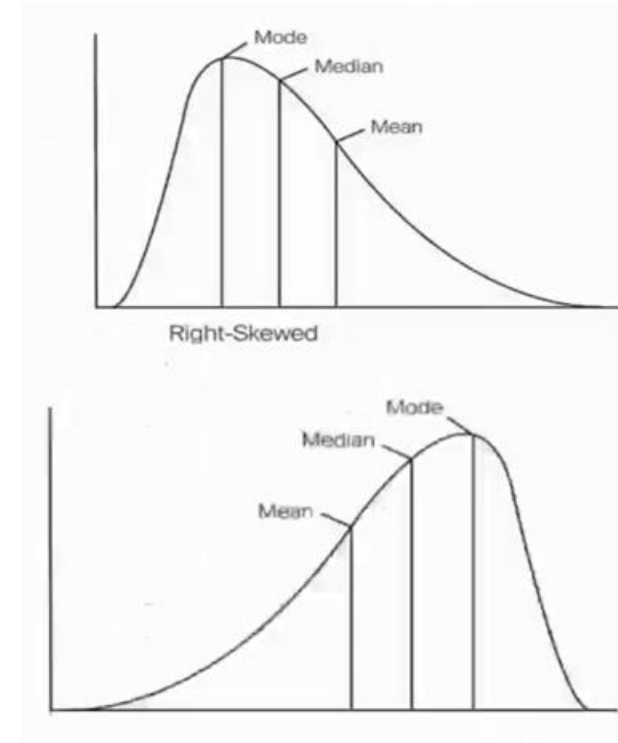
(b) Positively skewed data



(c) Negatively skewed data

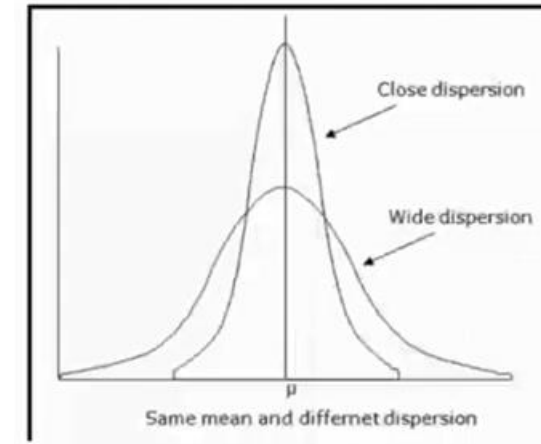
Central Tendency (+V and -V Skewed)

- Right Skewed or Positive Skewed-
 - long tail that extends to the right or positive side of the x axis, same as the below plot.
 - mean greater than the mode
 - median greater than the mode
 - mean greater than median
- Left Skewed or Negative Skewed-
 - which is left skewed have a long tail that extends to the left or negative side of the x axis, same as the below plot.
 - mean less than the mode
 - median less than the mode
 - mean less than median



- Variance & SD

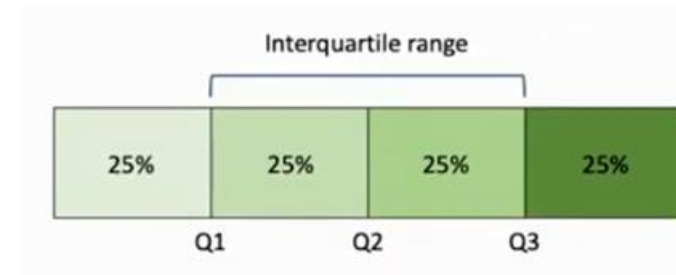
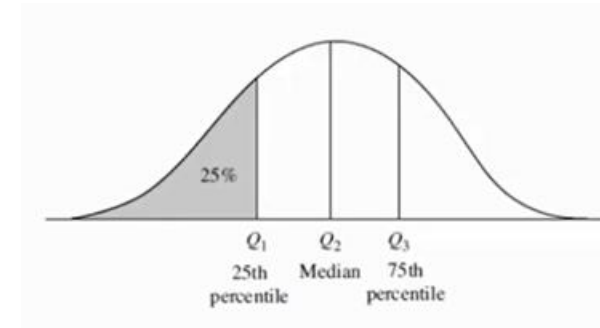
- Indicate how spread out a data distributed is
- Low SD: data observations tend to be very close to the mean
- High SD: data is spread out over a large range of value



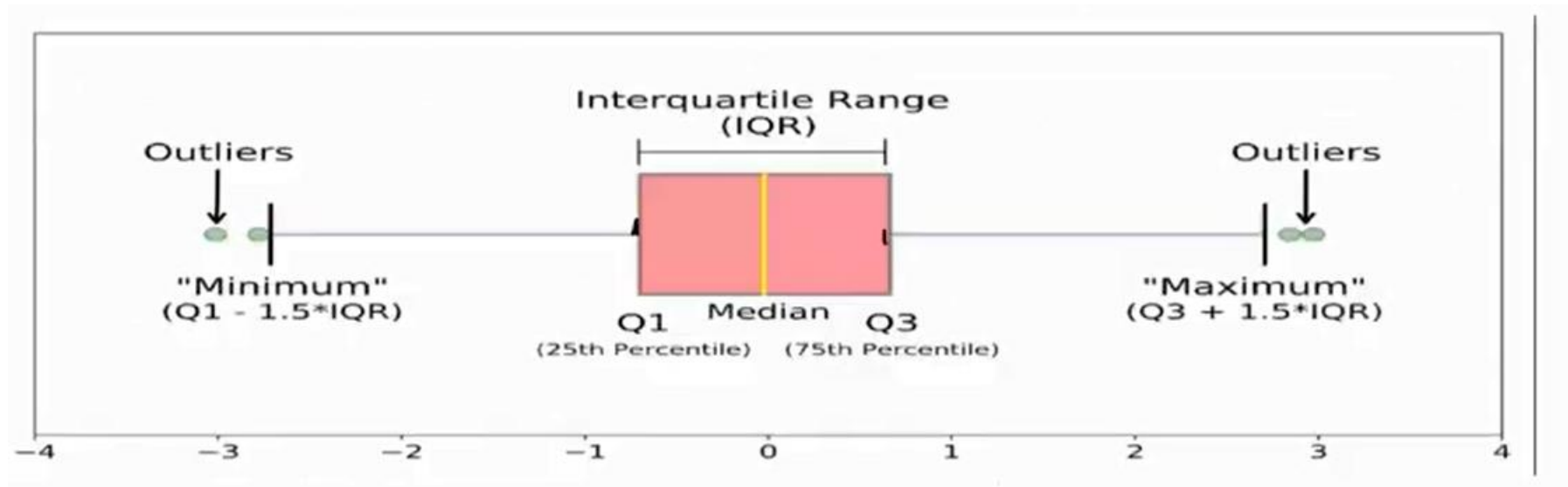
$$\sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 = \left(\frac{1}{N} \sum_{i=1}^N x_i^2 \right) - \bar{x}^2$$

$$\text{SD} = \sigma$$

- For N observations of numerical variable X: $X_1, X_2, \dots, X_N$
- Rang: difference between the largest and smallest values
 - Quantiles:
 - points taken at regular intervals of a data distribution, dividing it into (almost) equal-size consecutive sets
 - Interquartile Range:= $Q_3 - Q_1$ * the middle 50% of values when
 - ordered from lowest to highest



Five Number Summary



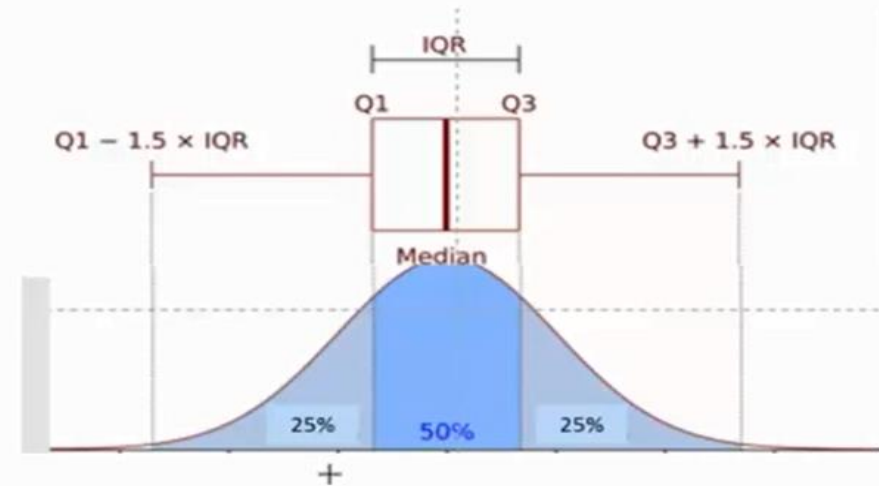
Five Number Summary

Five-Number Summary:

- Median (Q2), quartiles Q1 and Q3, & smallest and largest individual observations – in order

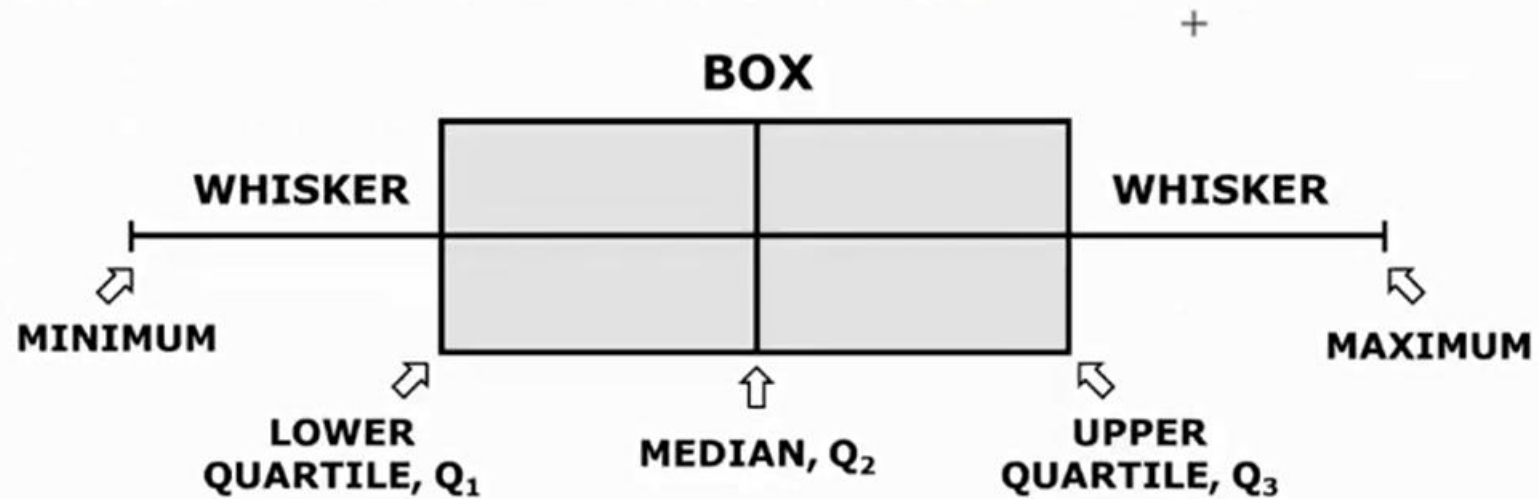
Boxplots:

- Visualization technique for the five-number summary
- **Whiskers** terminate at min & max OR the most extreme observations within
- The whiskers are the two lines outside the box, that go from the minimum to the lower quartile (the start of the box) and then from the upper quartile (the end of the box) to the maximum.
 - $1.5 \times \text{IQR}$ of the quartiles – with remainder points (outliers) plotted individually



Five Number Summary

Figure 4.5.2.1 Building a box and whisker plot

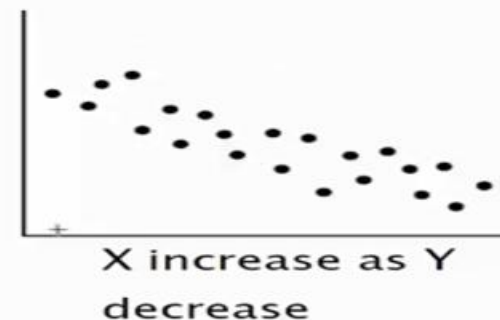


- Suppose that a hospital tested the age and body fat data for 18 randomly selected adults with the following results: Draw the boxplots for age and %fat.

Age	23	23	27	27	39	41	47	49	50	52	54	54	56	57	58	58	60	61
%fat	9.5	26.5	7.8	17.8	31.4	25.9	27.4	27.2	31.2	34.6	42.5	28.8	33.4	30.2	34.1	32.9	41.2	35.7

- Problem Solution:
- Draw the boxplots for age and %fat.
 - For Age
 - $Q1=39$, median= 51, $Q3=57$, min=23, max=61-
 - $IQR= 57-39= 18$, $1.5 IQR= 27^*$
 - newMin= $39-27= 12$, newMax= $57+27= 84$
 - For Fat
 - $Q1=26.5$, median= 30.7, $Q3=34.1$, min=7.8, max=42.5
 - $IQR= 34.1 - 26.5= 7.6$, $1.5 IQR= 11.4$
 - new Min= $26.5-11.4 = 15.1$, newMax= $34.1+11.4= 45.5$

- 1. Histograms
- 2. Scatter Plots
 - each pair of values is treated as a pair of coordinates and plotted as points in plane
 - X and Y are correlated if one attribute implies the other
 - positive, negative, or null (uncorrelated)
 - For more attributes, we use a scatter plot matrix



Week

05

Data Preparation (Data Preprocessing)

Reasons for Data Cleansing

- The data to be analyzed may be:
 - Incomplete; where the data is missing (empty record)
 - Noisy; where data may contain errors (data that have incorrect values -Gender Z)or outlier values (Outliers are data values that deviate from expected values of the rest of the data set) (Age 2500)
 - Inconsistent; where data may contain discrepancies in the values (age – birthdate) , (Total-price and (unit-price and quantity).

How can Data be Cleaned?

- Filling-in Missing Values
- Smoothing Noisy Data (Validation and correction – Standardization)
- Identifying and Removing Outliers Using Inter-quartile Range
- Resolving Inconsistency (age – birthdate)

There are several ways to deal with missing data:

- Replace the missing value with some default value
- Replace the missing value with the field mean for the fields that take numerical values or the mode (if exists) for the fields that take categorical values
- Replace the missing values with a value generated at random from the field distribution observed

Handling Outliers

Customer ID	Zip	Gender	Income	Age	Marital Status	Transaction Amount
1001	10048	M	75000	C	M	5000
1002	J2S7K7	F	-40000	40	W	4000
1003	90210		10000000	45	S	7000
1004	6269	M	50000	0	S	1000
1005	55101	F	99999	30	D	3000

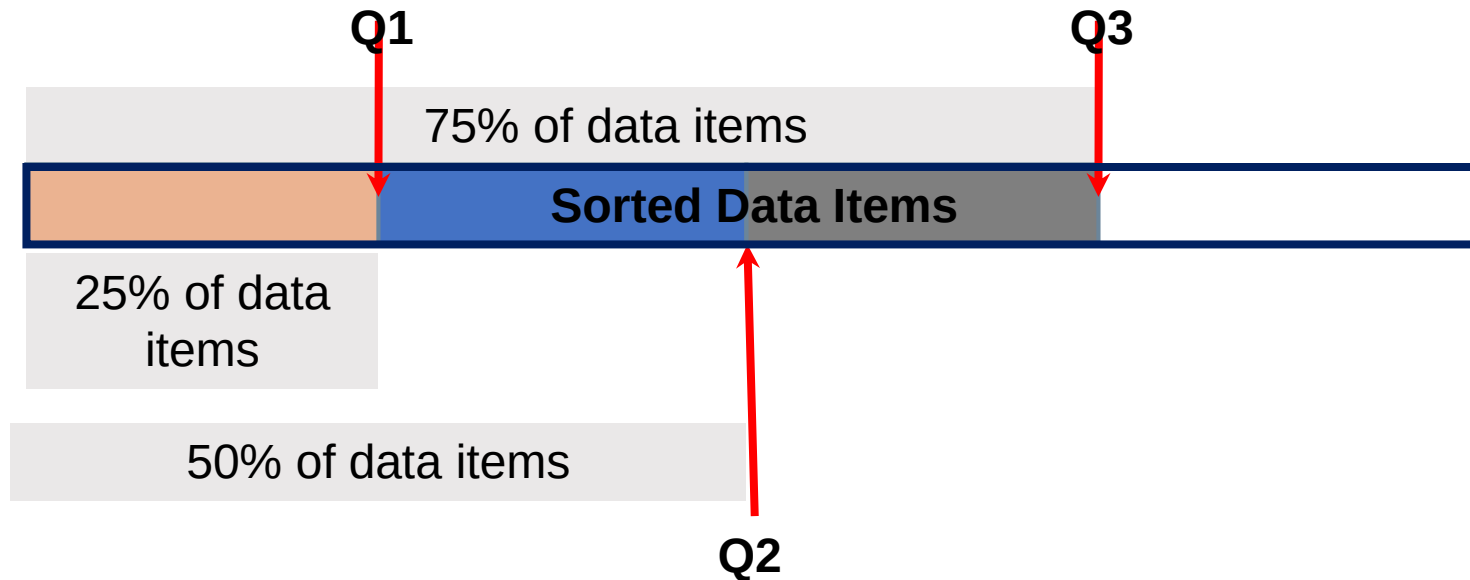
Sample Table

Data Set

Possible Outlier Values

- Outliers are data values that deviate from expected values of the rest of the data set
- Outliers are extreme values that lie near the limits of the data range or go against the trend of the remaining data.
- Normally, outliers need **more investigation** to make sure that they don't occur due to mistakes during data entry

Handling Outliers Using Inter-quartile Range



- A Quartile is any of the three values which divide the sorted data set into four equal parts
- First quartile (Q1) cuts off lowest 25% of data
- Second quartile (Q2) cuts data set in half (it is the median of the data set)
- Third quartile (Q3) cuts off highest 25% of data, or lowest 75%

Computing Q1, Q2, and Q3

- to compute Q1, Q2, and Q3, use the following method:
 - Order the given data set in ascending order.
 - Use the median to divide the ordered data set into two halves. This median is second quartile (Q2). Exclude this median (if it is one of the data items) from any further computation.
 - The first quartile (Q1) value is the median of the lower half of the data.
 - The third quartile (Q3) value is the median of the upper half of the data.

- Compute the Inter-Quartile Range (**IQR**) as follows:

$$\text{IQR} = Q3 - Q1$$

- A data value is an outlier if:
 - its value is $\leq (Q1 - 1.5 \cdot \text{IQR})$, or
 - its value is $\geq (Q3 + 1.5 \cdot \text{IQR})$.

- Noisy data are the kind of data that have incorrect values
- Some reasons for noisy data:
 - Data collection instruments may be faulty
 - Human or computer errors may occur during data entry
 - Transmission errors may occur
 - Technology limitations like buffer size, may occur during data-entry

Customer ID	Zip	Gender	Income	Age	Marital Status	Transaction Amount
1001	10048	M	75000	C	M	5000
1002	J2S7K7	F	-40000	40	W	4000
1003	90210		10000000	45	S	7000
1004	6269	M	50000	0	S	1000
1005	55101	F	99999	30	D	3000

Examples of Noisy Data

Smoothing Noisy Data

- By smoothing noisy data we can correct the errors
- Smoothing noisy data is performed by:
 - Validation and correction
 - Standardization

- Data inconsistency means that different data items contain discrepancies in their values
- It can occur when different data items depend on other data items and their values don't match; for example:
- Age and Birth-date; age can be computed from the birth-date, hence the value of Age must match the value computed from the birth-date
- City and Phone-area-code; each city has certain area-code
- Total-price and (unit-price and quantity); total-price can be computed from the unit-price and quantity
- These dependencies can be utilized to detect errors and substitute missing values or correct wrong values

Data Inconsistency Example

Client_Code	Place	Phone	Quantity	Unit_Price	Total_Price
101	Cairo	02-99XXXXXX	10	120	1200
102	Cairo	03-99XXXXXX	10	234	2340
102	Alexadria	03-99XXXXXX	20	400	600
104	Port-Said	03-99XXXXXX	30	100	300

Example of Inconsistent Data

Client_Code	Place	Phone	Quantity	Unit_Price	Total_Price
101	Cairo	02-99XXXXXX	10	120	1200
102	Cairo	03-99XXXXXX	10	234	2340
102	Alexadria	03-99XXXXXX	20	400	600
104	Port-Said	03-99XXXXXX	30	100	300

Data Inconsistency Marked in Red

Incorrect Area-Code

Total_Price doesn't equal (Quantity*Unit_Price)

Week

05

Data Integration , Transformation and Normalization

Methods for Collecting Raw Data

- Manually
 - Using Surveys
 - Using Observations
 - Based on Contributions from experts
- Using special computer peripherals equipments (instruments and/or sensors)
 - Bar-code readers
 - Computer-based medical imaging system
 - Remote sensing instruments
 - Atmosphere sensing instruments
- Web-based systems
 - Becomes very popular to collect clients' feedback via web
 - Examples are online polls and questionnaires

Data Sources Classifications

- Internal data sources
 - Internal sources are data sources that are housed within a company
 - An organization's internal data are about people, products, services, and processes
- Personal data sources
 - Employees may document their own expertise by creating personal data. These data are not necessarily just facts, but may include concepts, thoughts, and opinions. They include, for example, opinions about what competitors are likely to do, and certain rules and formulas developed by the users
- External data sources
 - There are many sources for external data, ranging from commercial databases to data collected by sensors and satellites
 - Internet and commercial database sources also considered as external sources

- Data Integration means that we need to merge data coming from multiple sources
- This is normally happened in data warehousing and data mining systems

- Entity Identification Problem (we have attributes (coming from different data sources) with different names that represent the same entity (**Student ID** , **Student no**)
- Redundancy (An attribute is redundant if it can be derived from another attribute (**correlation**))
- Data Value Conflicts (For the same entity, attribute values from different sources may differ (**meter, kilogram**)
- Duplication (there are two or more identical records for a given unique data **entry(discarding all duplication)**)

Entity Identification Problem

- Means that we have attributes (coming from different data sources) with different names that represent the same entity
- Example: we may have all the following attributes that represent the same entity:
 - Student_No
 - Bench_No
 - Student_Id
- All of these attributes could represent the same entity
- During data integration these attributes must be identified to reduce data redundancy and inconsistency

- An attribute is redundant if it can be derived from another attribute
- Some redundancy can be detected by correlation analysis

- Consider two attributes A and B coming from two different data sources. The correlation between attributes A and B can be measured by the correlation coefficient $r_{A,B}$:

$$r_{A,B} = \frac{\sum_{i=1}^n (A_i - \bar{A})(B_i - \bar{B})}{\{(n-1)\sigma_A\sigma_B\}}$$

Where

$$\bar{A} = \left(\frac{\sum_{i=1}^n A_i}{n} \right) \quad \bar{B} = \left(\frac{\sum_{i=1}^n B_i}{n} \right) \quad \sigma_A = \sqrt{\frac{\sum_{i=1}^n (A_i - \bar{A})^2}{n-1}} \quad \text{and} \quad \sigma_B = \sqrt{\frac{\sum_{i=1}^n (B_i - \bar{B})^2}{n-1}}$$

- Where n is the number of data items in A or B (assuming they have the same number of data items), $\bar{A}$ and $\bar{B}$ are the mean values of A and B, and σ_A and σ_B are the standard deviations of A and B

- If the $r_{A,B}$ is greater than 0 then A and B are positively correlated
- If A and B are positively correlated then the values of A increase as the values of B increase
- The higher the value of $r_{A,B}$ the more each attribute implies the other
- So, a high value of $r_{A,B}$ implies that either A or B may be removed as a redundancy but not both of them

- If the $r_{A,B}$ is 0, then A and B are independent, hence there is no redundancy and we cannot remove any of them
- If $r_{A,B}$ is less than 0 then A and B are negatively correlated
- This implies that the values of one attribute increase as the values of the other decrease and also we cannot remove any of the attributes

Data Value Conflicts

- For the same entity, attribute values from different sources may differ
- This may be due to differences in representation, scaling, or encoding
- Example: a weight attribute may be stored in metric units (meter, kilogram ...) in one system and British units (Pound, inch ...) in another
- The price of different hotels may involve not only different currencies but also different services and taxes
- Hence, these conflicts must be resolved before integrating data sources
- Resolving these kinds of conflicts is done by converting one unit to the other so that we use a unified unit for all integrated data sources
- Careful integration of data from multiple sources helps in reducing and avoiding redundancy and inconsistency in the resulting data sets

- Duplication means there are two or more identical records for a given unique data entry
- It occurs when you have the same record coming from different data sources
- Resolving it can be done by discarding all duplication and leaving single unique data entry for each entity

- Data Normalization means that the attribute data are scaled to fall within a small specified range such as -1.0 to $+1.0$, or 0.0 to 1.0
- Normalization is important for data classification and analysis by data mining techniques

- Data normalization is a critical preprocessing step in data analysis and machine learning to bring data into a standard format or scale, ensuring that different features contribute equally to the analysis and preventing issues caused by varying scales.
- There are various methods of data normalization, each suited to specific types of data or analysis. Here are some common types of data normalization:

- Min-Max Scaling (Normalization)
- Z-Score Standardization (Standard Scaling)
- Decimal Scaling
- Robust Scaling (Scaling by Median and IQR)
- Logarithmic Transformation
- Box-Cox Transformation
- Quantile Transformation
- Rank Transformation
- Unit Vector Transformation
- Sparse Data Normalization
- Image Normalization

Min-Max Normalization: Example

- Also known as feature scaling, it scales the data between a specified range, typically [0, 1].
 - Formula: $X_{\text{normalized}} = (X - X_{\text{min}}) / (X_{\text{max}} - X_{\text{min}})$
- Useful when you have features with varying ranges and want to bring them into a common scale.

- Min-Max normalization performs a linear transformation on the original data
- Suppose that $\min_A$ and $\max_A$ are the minimum and maximum values of an attribute A
- Min-Max normalization maps any value v of A to v' in the range $[\text{new_min}_A, \text{new_max}_A]$ by computing:

$$v' = \left(\left[\frac{(v - \min_A)}{(\max_A - \min_A)} \right] \times [\text{new_max}_A - \text{new_min}_A] \right) + \text{new_min}_A$$

- Min-max normalization preserves the relationships among the original data values

Min-Max Normalization: Example

- Let the minimum and maximum values for the attribute income be \$12000 and \$98000
- To map income to the range [0.0, 1.0], we have:

$$\text{minA} = 12000 \quad \text{maxA} = 98000$$

$$\text{new_minA} = 0.0 \quad \text{new_maxA} = 1.0$$

- By min-max normalization, a value of \$73600 for income is transformed to:

$$\left(\frac{73600 - 12000}{98000 - 12000} \right) \times [1.0 - 0.0] + 0.0 = 0.716$$

- In Z-Score normalization,
- It scales the data to have a mean of 0 and a standard deviation of 1.
 - Formula: $X_{\text{standardized}} = (X - \text{mean}(X)) / \text{std}(X)$
- Useful when you want to center data around zero and have it in units of standard deviations.

- In Z-Score normalization, the values for an attribute A are normalized based on the mean and standard deviation of A
- A value v of A is normalized to v' by computing:

$$v' = \left(\frac{v - \bar{A}}{\sigma_A} \right) \quad \text{Where} \quad \bar{A} = \left(\frac{\sum_{i=1}^n A}{n} \right) \quad \text{and} \quad \sigma_A = \sqrt{\frac{\sum_{i=1}^n (A_i - \bar{A})^2}{n-1}}$$

- Where $\bar{A}$ is the mean value of A and σ_A is the standard deviation of A
- The Z-Score normalization method is useful when the actual minimum and maximum of attribute A are unknown, or when there are outliers that dominate the min-max normalization

Z-Score Normalization: Example

- Suppose that the mean and standard deviation of the values for the attribute income are \$54000 and \$16000, respectively
- With Z-Score normalization, a value of \$73600 for income is transformed to

$$\frac{73600 - 54000}{16000} = 1.225$$

Normalization by decimal scaling

- This method normalizes by moving the decimal point of values of attribute A
- The number of decimal points moved depends on the maximum absolute value of A
- A value v of A is normalized to v' by computing:

$$v' = \frac{v}{10^j}$$

- Where j is the smallest integer such that $\text{Max}(|v'|) < 1$

Normalization by decimal scaling: Example

- Let the values of A range from -986 to 917
- The maximum absolute value of A is 986
- To normalize by decimal scaling, we divide each value by 1000, which means $j = 3$
- Hence, -986 normalizes to -0.986

Robust Scaling (Scaling by Median and IQR):

- Let Scales data based on the median and the interquartile range (IQR) to minimize the effect of outliers.
 - Formula: $X_{\text{robust}} = (X - \text{median}(X)) / \text{IQR}(X)$
- Useful when your data has outliers, and you want a scaling method that is robust to extreme values.

Robust Scaling (Scaling by Median and IQR):

- Robust scaling, also known as scaling by the median and interquartile range (IQR), is a data normalization technique that minimizes the effect of outliers on your data. It's particularly useful when your dataset contains extreme values that can significantly affect other scaling methods like standardization (z-score).
- Let's walk through a solved example to demonstrate how to perform robust scaling:

Robust Scaling (Scaling by Median and IQR):

- to Ages (in years):
- 18, 21, 22, 25, 29, 31, 34, 35, 42, 170
- Step 1: Collect Data
- Suppose you have a dataset representing the ages of individuals in a small town, and it contains some outliers
- Step 2: Calculate the Median and IQR
- First, calculate the median (M) and the interquartile range (IQR) of your data. The IQR is the range between the 75th percentile (Q3) and the 25th percentile (Q1).
- Median (M): To find the median, first sort your data and find the middle value. In this case, the sorted ages are:
- The middle value is 29, so the median (M) is 29.

Robust Scaling (Scaling by Median and IQR):

- Step 3: Perform Robust Scaling Now that you have the median (M) and the IQR, you can scale your data using the robust scaling formula:

- $X_{\text{robust}} = (X - M) / IQR$

- Apply this formula to each data point:

- For 18 years old:

- $X_{\text{robust}} = (18 - 29) / 16 \approx -0.6875$

- For 21 years old:

- $X_{\text{robust}} = (21 - 29) / 16 \approx -0.5$

- Continue this process for all data points, including the outlier (170 years old).

Robust Scaling (Scaling by Median and IQR):

- Step 4: Analyze the Scaled Data
- After performing the robust scaling, you have a new dataset of scaled values. These values are less affected by the outlier, and the entire dataset is now on a similar scale, making it suitable for various statistical analyses and machine learning algorithms.
- Robust scaling is valuable when you want to normalize your data while minimizing the impact of extreme values like outliers, as it relies on the robust statistics of the median and IQR.

- Takes the logarithm of the data values.
- Useful when dealing with data that follows an exponential distribution or has a large range of values, making it right-skewed

- Logarithmic normalization is a data transformation technique commonly used to reduce the skewness in data that follows a highly right-skewed distribution. This technique is particularly useful when dealing with data that spans several orders of magnitude, such as income or population data.
- By applying a logarithmic transformation, you can make the data more symmetric and amenable to analysis.
- Here's a step-by-step example of logarithmic normalization using a simple dataset:

- Step 1: Collect Data
- Let's assume you have a dataset representing the income of individuals in a small town:
- Income (in dollars):
- \$10,000
- \$20,000
- \$30,000
- \$50,000
- \$80,000
- \$120,000
- \$200,000
- \$300,000

- Step 2: Calculate the Logarithm
- To perform logarithmic normalization, you'll take the natural logarithm (base e) of each data point. You can use a calculator or a programming language like Python to do this. The formula is:
- $\text{Logarithm}(x) = \ln(x)$
- Applying this to our dataset:
- $\text{Logarithm}(10,000) \approx 9.210$
- $\text{Logarithm}(20,000) \approx 9.903$
- $\text{Logarithm}(30,000) \approx 10.308$
- $\text{Logarithm}(50,000) \approx 10.819$
- $\text{Logarithm}(80,000) \approx 11.289$
- $\text{Logarithm}(120,000) \approx 11.695$
- $\text{Logarithm}(200,000) \approx 12.206$
- $\text{Logarithm}(300,000) \approx 12.611$

- Step 3: Analyze the Transformed Data
- Now that you've applied the logarithmic transformation to your data, you can analyze the transformed values. Notice that the transformed data is much less skewed and spans a smaller range compared to the original income values. This makes it easier to work with in various statistical analyses.
- Step 4: Interpretation
- When interpreting the results, keep in mind that the transformed values represent the logarithm of income rather than income itself. For example, an income of \$30,000 corresponds to a logarithmic value of approximately 10.308. To get back to the original income scale, you can take the antilogarithm, which is the inverse operation:
- $\text{Income} = e^{(\text{Logarithm})}$
- Logarithmic normalization is useful when you want to make data with a wide range of values more suitable for analysis, particularly when dealing with skewed datasets.

Box-Cox Transformation:

- A family of power transformations that can handle data with different shapes.
- It includes the logarithm transformation as a special case.
- Useful when you want to stabilize variance or make data more normally distributed.
- To perform the transformation, use the formula:
- $\text{Transformed Value} = (X^\lambda - 1) / \lambda, \text{ if } \lambda \neq 0$
- $\ln(X), \text{ if } \lambda = 0$

Box-Cox Transformation:

- to compute Q1, Q2, and Q3, use the following method:
 - Order the given data set in ascending order.
 - Use the median to divide the ordered data set into two halves. This median is second quartile (Q2). Exclude this median (if it is one of the data items) from any further computation.
 - The first quartile (Q1) value is the median of the lower half of the data.
 - The third quartile (Q3) value is the median of the upper half of the data.

Box-Cox Transformation:

- to compute Q1, Q2, and Q3, use the following method:
 - Order the given data set in ascending order.
 - Use the median to divide the ordered data set into two halves. This median is second quartile (Q2). Exclude this median (if it is one of the data items) from any further computation.
 - The first quartile (Q1) value is the median of the lower half of the data.
 - The third quartile (Q3) value is the median of the upper half of the data.

Box-Cox Transformation:

- Step 2: Determine the Lambda (λ) Parameter
- The Box-Cox transformation involves raising each data point to the power of a lambda (λ) parameter, which you need to determine. To do this, you can use statistical software or libraries in Python or R to find the lambda that maximizes the normality of the transformed data. We'll assume that the lambda for this dataset is approximately 0.25.
- Step 3: Apply the Box-Cox Transformation
- Now that you have the lambda parameter ($\lambda = 0.25$), you can apply the Box-Cox transformation to each data point:

Box-Cox Transformation:

- To perform the transformation, use the formula:
- Transformed Value = $(X^\lambda - 1) / \lambda$, if $\lambda \neq 0$
- $\ln(X)$, if $\lambda = 0$
- In our case, λ is approximately 0.25, so we use the first part of the formula.
 - For \$150,000:
 - Transformed Value = $(150,000^{0.25} - 1) / 0.25 \approx 121.67$
 - For \$200,000:
 - Transformed Value = $(200,000^{0.25} - 1) / 0.25 \approx 141.42$
- Continue this process for all data points.

Box-Cox Transformation:

■ Transformed House Prices:

- \$121.67
- \$141.42
- \$149.70
- \$162.28
- \$174.92
- \$193.53
- \$206.14
- \$218.18
- \$226.16
- \$243.34

Box-Cox Transformation:

- Step 4: Analyze the Transformed Data
 - After applying the Box-Cox transformation, you have a new dataset of transformed values. This transformed data is expected to have a reduced degree of heteroscedasticity and be closer to a normal distribution. You can use this transformed data for statistical analyses that assume normality or when you want to improve the performance of certain machine learning models.
 - Keep in mind that the lambda parameter is essential, and it should be chosen carefully to maximize the normality of the transformed data. Different datasets may require different lambda values to achieve the desired transformation.

Quantile Transformation:

- Maps the data to a specified quantile distribution, often resulting in a uniform distribution.
- Useful when you want to ensure that the data follows a specific distribution or remove the effect of outliers
- Quantile transformation is a data normalization technique that maps the data to a specified quantile distribution, often resulting in a uniform distribution. This transformation is useful when you want to ensure that the data follows a specific distribution or remove the effects of outliers.

Here's a solved example of quantile transformation:

Quantile Transformation:

- Step 1: Collect Data
- Suppose you have a dataset representing the scores of students on a difficult exam:
- Exam Scores:
 - 45
 - 60
 - 62
 - 70
 - 80
 - 90
 - 95
 - 100

- Step 2: Specify the Desired Quantile Distribution
- In this example, let's assume we want to transform the data into a uniform distribution between 0 and 1. This means we'll map the data to the percentiles of a uniform distribution.
- Step 3: Calculate the Percentiles of the Original Data
- To perform quantile transformation, you need to calculate the percentile ranks of each data point. You can use a formula like:
 - $\text{Percentile Rank} = (\text{Number of values below } X + 0.5) / \text{Total number of values}$

Quantile Transformation:

- Apply this formula to each data point:
- For 45:
 - $\text{Percentile Rank} = (0 + 0.5) / 8 \approx 0.0625$
- For 60:
 - $\text{Percentile Rank} = (1 + 0.5) / 8 \approx 0.1875$
- Continue this process for all data points.

Quantile Transformation:

- Step 4: Map the Percentiles to the Desired Distribution
- Now that you have calculated the percentile ranks for each data point, you can map these percentiles to the desired uniform distribution (0 to 1). The mapped values become the transformed data.
- For a percentile rank of 0.0625, the transformed value would be:
 - $\text{Transformed Value} = 0 + 0.0625 * (1 - 0) = 0.0625$
 - For a percentile rank of 0.1875, the transformed value would be:
 - $\text{Transformed Value} = 0 + 0.1875 * (1 - 0) = 0.1875$
 - Continue this process for all percentile ranks.

Quantile Transformation:

- Step 5: Analyze the Transformed Data After applying the quantile transformation, you have a new dataset of transformed values that follow a uniform distribution between 0 and 1. This transformation can be useful when you want to work with data that conforms to a specific distribution, such as uniformity, and it can be especially beneficial in statistical analyses that assume uniformity.
- Keep in mind that the choice of the desired distribution and the mapping of percentiles may vary based on your specific goals and the characteristics of your data.

- Assigns ranks to data points, converting them into a uniform distribution.
- Useful when you want to make the data more robust to outliers and non-normality.
- Step 1: Collect Data
- Suppose you have a dataset representing the test scores of students in a class:
- Test Scores:
 - 78
 - 85
 - 92
 - 70
 - 65
 - 88
 - 95
 - 60

Rank Transformation Example

- Step 2: Rank the Data
- To perform rank transformation, you need to assign a rank to each data point. Ranks are assigned based on the order of the data values, with the lowest value receiving a rank of 1, the second lowest a rank of 2, and so on. Ties are typically handled by giving tied values the average rank.
- Let's rank the data based on ascending order:
 - Rank of 60: 1
 - Rank of 65: 2
 - Rank of 70: 3
 - Rank of 78: 4
 - Rank of 85: 5
 - Rank of 88: 6
 - Rank of 92: 7
 - Rank of 95: 8

Rank Transformation Example

- Step 3: Replace Data with Ranks
- Replace the original data with their corresponding ranks:
- Ranked Test Scores:
 - 4
 - 5
 - 3
 - 1
 - 2
 - 6
 - 7
 - 8

Rank Transformation Example

- Step 4: Analyze the Transformed Data
- After performing the rank transformation, you have a new dataset of ranked values. This transformation can be beneficial when you want to make the data more robust to outliers and make it suitable for non-parametric statistical tests or analyses that assume uniformity.
- Keep in mind that rank transformation does not change the order of values relative to each other. It simply replaces the original values with their positions in the sorted order. This can be useful when you want to focus on the order or rank of data points rather than their absolute values.

Unit Vector Transformation:

- Scales each data point to have a length of 1, turning it into a unit vector.
- Often used in text mining and natural language processing to normalize word frequency data
- Unit vector transformation, also known as vector normalization, is a data preprocessing technique used to scale data vectors to have a length or magnitude of 1. This normalization is often used in machine learning, particularly in algorithms like cosine similarity or K-nearest neighbors (K-NN). Let's go through an example of unit vector transformation:

Unit Vector Transformation Example:

- Step 1: Collect Data
- Suppose you have a dataset of three-dimensional vectors representing the positions of points in a 3D space:
 - Vector 1: $[2, 3, 4]$
 - Vector 2: $[1, 0, 1]$
 - Vector 3: $[4, 1, 2]$

Unit Vector Transformation Example:

- Step 2: Calculate the Magnitude of Each Vector
- To perform unit vector transformation, you first need to calculate the magnitude (length) of each vector using the Euclidean distance formula:
- Magnitude ($||V||$) = $\sqrt{x^2 + y^2 + z^2}$
- For each vector, calculate its magnitude:
- Magnitude of Vector 1:
 $||[2, 3, 4]|| = \sqrt{2^2 + 3^2 + 4^2} = \sqrt{4 + 9 + 16} = \sqrt{29}$
- Magnitude of Vector 2:
 $||[1, 0, 1]|| = \sqrt{1^2 + 0^2 + 1^2} = \sqrt{1 + 0 + 1} = \sqrt{2}$
- Magnitude of Vector 3:
 $||[4, 1, 2]|| = \sqrt{4^2 + 1^2 + 2^2} = \sqrt{16 + 1 + 4} = \sqrt{21}$

Unit Vector Transformation Example:

- Step 3: Normalize the Vectors to Unit Vectors
- To normalize each vector to a unit vector, divide each component of the vector by its magnitude:
- Unit Vector 1:
[$2/\sqrt{29}$, $3/\sqrt{29}$, $4/\sqrt{29}$]
- Unit Vector 2:
[$1/\sqrt{2}$, $0/\sqrt{2}$, $1/\sqrt{2}$]
- Unit Vector 3:
[$4/\sqrt{21}$, $1/\sqrt{21}$, $2/\sqrt{21}$]
- These are the unit vectors corresponding to the original vectors. They all have a magnitude of 1, and they point in the same direction as the original vectors but have been scaled down to have unit length.

Unit Vector Transformation Example:

- Step 4: Analyze the Transformed Data
- After applying the unit vector transformation, you have a new dataset of unit vectors. These unit vectors are useful when you want to compare the directions or similarities between vectors without being influenced by their magnitudes.
- Unit vector transformation is often used in various applications, such as text analysis (for representing documents as unit vectors) and recommendation systems (for finding similar items based on user preferences).

Sparse Data Normalization (TF-IDF):

- Specialized normalization techniques for sparse data, such as TF-IDF (Term Frequency-Inverse Document Frequency) for text data.
- The goal is to normalize non-zero values without affecting the zeros. Here's an example of sparse data normalization using a simple document-term matrix:

- Step 1: Collect Data
- Suppose you have a document-term matrix that represents the frequency of words in three documents:
- Term Frequency Matrix:
- | | term1 | term2 | term3 | term4 |
|------|-------|-------|-------|-------|
| Doc1 | 2 | 0 | 1 | 0 |
| Doc2 | 1 | 3 | 0 | 2 |
| Doc3 | 0 | 0 | 1 | 1 |
- In this matrix, most of the entries are zeros, indicating the absence of a term in a document.

Sparse Data Normalization (TF-IDF):

- Step 2: Normalize the Non-Zero Values
- To normalize the non-zero values in a sparse matrix, you typically use one of the standard normalization techniques like Min-Max scaling, z-score standardization, or TF-IDF (Term Frequency-Inverse Document Frequency). Let's use TF-IDF as an example:
- Term Frequency (TF): Calculate the term frequency for each term in each document. This is the count of how many times each term appears in the document divided by the total number of terms in the document.
- Inverse Document Frequency (IDF): Calculate the inverse document frequency for each term. This is a measure of how important a term is across all documents. It's calculated as the logarithm of the total number of documents divided by the number of documents containing the term.

- Here's a simplified example of TF-IDF calculation:
- TF-IDF Matrix:
- | | term1 | term2 | term3 | term4 |
|------|--------|--------|--------|--------|
| Doc1 | 0.7071 | 0.0000 | 0.3010 | 0.0000 |
| Doc2 | 0.3010 | 0.7071 | 0.0000 | 0.3010 |
| Doc3 | 0.0000 | 0.0000 | 0.3010 | 0.3010 |
- In this TF-IDF matrix, the non-zero values have been normalized based on their term frequency and inverse document frequency, while the zeros remain unchanged.

Sparse Data Normalization (TF-IDF):

- Step 3: Analyze the Transformed Data
- After applying sparse data normalization, you have a new matrix where the non-zero values have been scaled and normalized. This is particularly useful in NLP tasks like text classification or information retrieval, where you want to focus on the importance of terms in documents without being affected by the varying document lengths or the presence of many zeros in the data.
- Sparse data normalization helps capture the relative importance of terms and documents in a sparse matrix, making it a crucial step in many text analysis applications.

Image Normalization:

- Specific techniques for normalizing pixel values in images, such as mean subtraction and standard deviation scaling.



مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY



Thank You

