Week 7: Data Mining

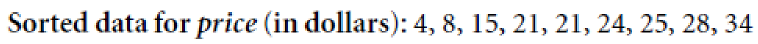
Data Preprocessing

- Today's real-world databases are highly susceptible to noisy, missing, and inconsistent data due to their typically huge size (often several gigabytes or more) and their likely origin from multiple, heterogeneous sources.
- There are several data preprocessing techniques.
    - **Data cleaning** can be applied to remove noise and correct inconsistencies in data.
    - **Data integration** merges data from multiple sources into a coherent data store such as a data warehouse.
    - **Data reduction** can reduce data size by, for instance, aggregating, eliminating redundant features, or clustering.
    - **Data transformations** (e.g., normalization) may be applied, where data are scaled to fall within a smaller range like 0.0 to 1.0. This can improve the accuracy and efficiency of mining algorithms involving distance measurements.
    - These techniques are not mutually exclusive; they may work together. For example, data cleaning can involve transformations to correct wrong data, such as by transforming all entries for a date field to a common format.
- Data processing techniques, when applied before mining, can substantially improve the overall quality of the patterns mined and/or the time required for the actual mining.

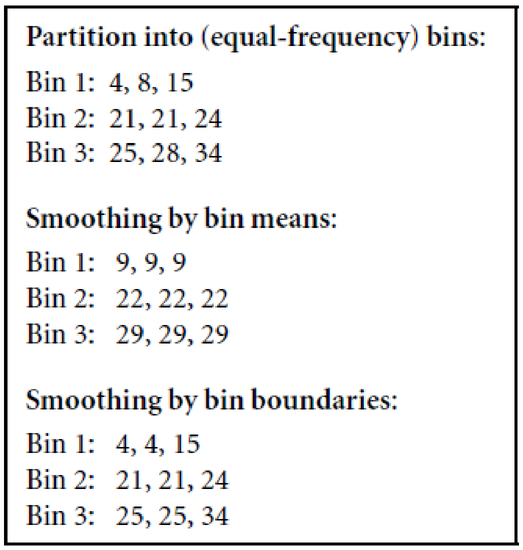
Data Cleaning

- Real-world data tends to be incomplete, noisy, and inconsistent. Data cleaning (or data cleansing) routines attempt to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data.
- **Missing Values**
    - **Ignore the tuple: *drop nan***
        - This is usually done when the class label is missing (assuming the mining task involves classification).
        - This method is not very effective, unless the tuple contains several attributes with missing values. It is especially poor when the percentage of missing values per attribute varies considerably.
        - By ignoring the tuple, we do not make use of the remaining attributes' values in the tuple. Such data could have been useful to the task at hand.
    - **Fill in the missing value manually**
        - This approach is time consuming and may not be feasible given a large data set with many missing values.
    - **Use a global constant to fill in the missing value**
        - Replace all missing attribute values by the same constant such as a label like ***"Unknown".***
    - **Use a measure of central tendency for the attribute (e.g., the mean or median) to fill in the missing value:**
        - Measures of central tendency, which indicate the "middle" value of a data distribution.
        - For normal (symmetric) data distributions, the mean can be used, while skewed data distribution should employ the median.

    - **Use the attribute mean or median for all samples belonging to the same class as the given tuple**
        - For example, if classifying customers according to credit risk, we may replace the missing value with the mean income value for customers in the same credit risk category as that of the given tuple. If the data distribution for a given class is skewed, the median value is a better choice.
- **Noisy Data**
    - Noise is a random error or variance in a measured variable.
    - Given a numeric attribute such as, say, price, how can we “smooth” out the data to remove the noise?
    - **Binning**
        - Binning methods smooth a **sorted data value** by consulting its **“neighborhood,”** that is, the values around it. The sorted values are distributed into a number of **“buckets,” or bins.**
        - Because binning methods consult the neighborhood of values, they perform local smoothing



Sorted data for *price* (in dollars): 4, 8, 15, 21, 21, 24, 25, 28, 34



Partition into (equal-frequency) bins:

Bin 1: 4, 8, 15
Bin 2: 21, 21, 24
Bin 3: 25, 28, 34

Smoothing by bin means:

Bin 1: 9, 9, 9
Bin 2: 22, 22, 22
Bin 3: 29, 29, 29

Smoothing by bin boundaries:

Bin 1: 4, 4, 15
Bin 2: 21, 21, 24
Bin 3: 25, 25, 34

        -

- In this example, the data for price are first sorted and then partitioned into **equal-frequency bins** of size 3 (i.e., each bin contains three values).
- **In smoothing by bin means,** each value in a bin is replaced by the **mean value** of the bin. For example, the mean of the values 4, 8, and 15 in Bin 1 is 9. Therefore, each original value in this bin is replaced by the value 9.
- Similarly, **smoothing by bin medians** can be employed, in which each bin value is replaced by the **bin median.**
- In **smoothing by bin boundaries,** the minimum and maximum values in a given bin are identified as the bin boundaries.
  - Each bin value is then replaced by the closest boundary value. In general, the larger the width, the greater the effect of the smoothing.