MYONGJI UNIVERSITY

Artificial Intelligence

Lecture 4. Data Acquisition & Preprocessing

III. Data Preprocessing

Spring 2022

Prof. Jonghoon Chun, Ph.D.

E-mail : jchun@mju.ac.kr
Lecture Note : http://lms.mju.ac.kr

Data Acquisition & Preprocessing

- Data (Know your data)
- Data Acquisition
- Data Preprocessing

Data Quality

- Poor data quality negatively affects many data processing efforts

“The most important point is that poor data quality is an unfolding disaster.”

  – Poor data quality costs the typical company at least ten percent (10%) of revenue; twenty percent (20%) is probably a better estimate.”

- A classification model for detecting people who are loan risks is built using poor data
  – Some credit-worthy candidates are denied loans
  – More loans are given to individuals that default

Measures for data quality

- A multidimensional view
  - Accuracy: correct or wrong, accurate or not
  - Completeness: not recorded, unavailable, …
  - Consistency: some modified but some not, dangling, …
  - Timeliness: timely update?
  - Believability: how trustable the data are correct?
  - Interpretability: how easily the data can be understood?

Data Quality Problems

- What kinds of data quality problems?

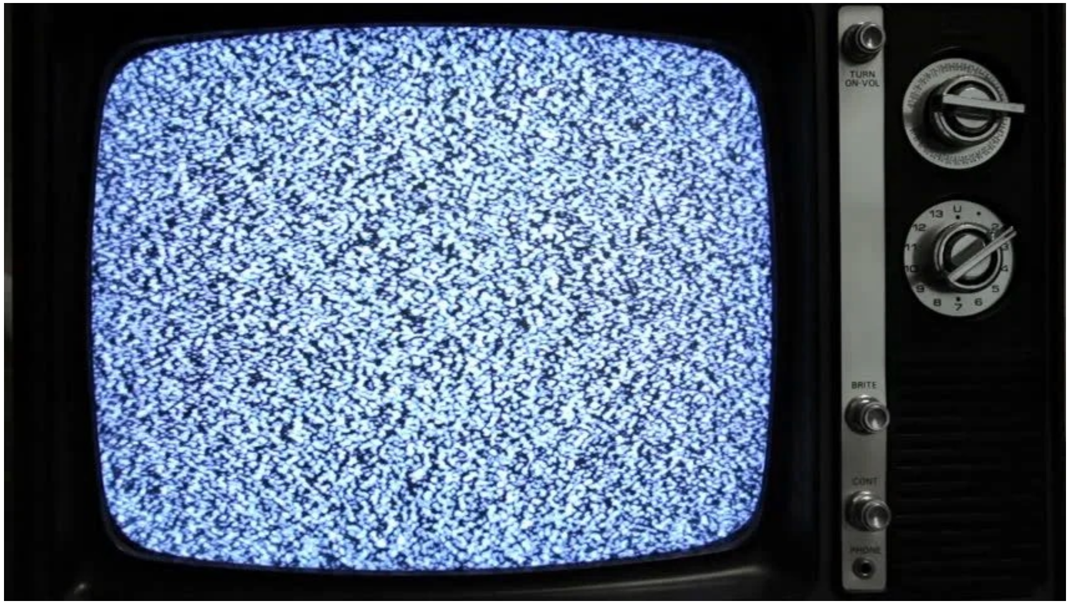
- How can we detect problems with the data?

- What can we do about these problems?

- Examples of data quality problems:
  – Noise and outliers
  – Missing values
  – Duplicate data
  – Wrong data

Noise

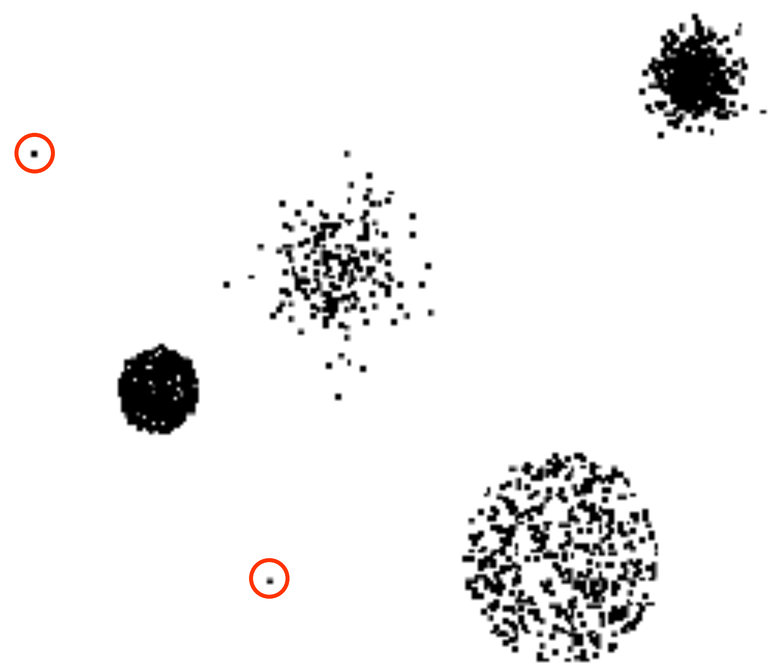
- For objects, noise is an extraneous object
- For attributes, noise refers to modification of original values
  - Examples: distortion of a person's voice when talking on a poor phone and "snow" on television screen



Snowy Picture of TV Screen

Outliers

- ***Outliers*** are data objects with characteristics that are considerably different than most of the other data objects in the data set
  - **Case 1:** Outliers are noise that interferes with data analysis
  - **Case 2:** Outliers are the goal of our analysis
    - ✓ Credit card fraud
    - ✓ Intrusion detection



Missing Values

- Reasons for missing values
  - Information is not collected
    (e.g., people decline to give their age and weight)
  - Attributes may not be applicable to all cases
    (e.g., annual income is not applicable to children)

- Handling missing values
  - Eliminate data objects or variables
  - Estimate missing values
    - Example: time series of temperature
    - Example: census results
  - Ignore the missing value during analysis

Duplicate Data

- Data set may include data objects that are duplicates, or almost duplicates of one another
  - Major issue when merging data from heterogeneous sources

- Examples:
  - Same person with multiple email addresses

- Data Cleansing
  - Process of dealing with duplicate data issues

Major Tasks in Data Preprocessing

- **Data cleansing**
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- **Data integration**
  - Integration of multiple databases, data cubes, or files
- **Data reduction**
  - Dimensionality reduction
  - Numerosity reduction
- **Data transformation and data discretization**
  - Normalization
  - Concept hierarchy generation

Data Clean(s)ing

- Data in the Real World Is Dirty!
  - incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
    - e.g., *Occupation*=" " (missing data)
  - noisy: containing noise, errors, or outliers
    - e.g., *Salary*="−10" (an error)
  - inconsistent: containing discrepancies in codes or names, e.g.,
    - *Age*="42", *Birthday*="03/07/2010"
    - Was rating "1, 2, 3", now rating "A, B, C"
    - discrepancy between duplicate records
  - Intentional (e.g., *disguised missing* data)
    - Jan. 1 as everyone's birthday?

How to Handle Missing Data?

- Ignore the tuple: usually done when class label is missing (when doing classification)—not effective when the % of missing values per attribute varies considerably
- Fill in the missing value manually: tedious + infeasible?
- Fill in it automatically with
  - a global constant : e.g., "unknown", a new class?
  - the attribute mean
  - the attribute mean for all samples belonging to the same class: smarter
  - the most probable value: inference-based such as Bayesian formula or decision tree

How to Handle Noisy Data?

- Binning
  - first sort data and partition into (equal-frequency) bins
  - then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.
- Regression
  - smooth by fitting the data into regression functions
- Clustering
  - Detect and remove outliers
- Combination of human inspection with computer
  - Detect suspicious values and check by human (e.g., deal with possible outliers)

Data Cleansing as a Process

- Data discrepancy detection
  - Use metadata (e.g., domain, range, dependency, distribution)
  - Check field overloading
  - Check uniqueness rule, consecutive rule and null rule
  - Use commercial tools
    - Data scrubbing: use simple domain knowledge (e.g., postal code, spell-check) to detect errors and make corrections
    - Data auditing: by analyzing data to discover rules and relationship to detect violators (e.g., correlation and clustering to find outliers)
- Data migration and integration
  - Data migration tools: allow transformations to be specified
  - ETL (Extraction/Transformation/Loading) tools: allow users to specify transformations through a graphical user interface

Data Integration

- Data integration
  - Combines data from multiple sources into a coherent store
  - Schema integration: e.g., A.cust-id ≡ B.cust-#
  - Integrate metadata from different sources

- Entity identification problem
  - Identify real world entities from multiple data sources, e.g., Bill Clinton = William Clinton
  - Detecting and resolving data value conflicts

Handling Redundancy in Data Integration

- Redundant data occur often when integration of multiple databases
  - Object identification: The same attribute or object may have different names in different databases
  - Derivable data: One attribute may be a "derived" attribute in another table, e.g., annual revenue
- Careful integration may help reduce/avoid redundancies and inconsistencies and improve mining speed and quality
- Redundant attributes may be able to be detected by correlation analysis and covariance analysis

Correlation Analysis (Numeric Data)

- Correlation coefficient (also called Pearson's product moment coefficient)

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

where n is the number of tuples, $\bar{A}$ and $\bar{B}$ are the respective means of A and B, σ_A and σ_B are the respective standard deviation of A and B, and $\Sigma(a_i b_i)$ is the sum of the AB cross-product.

- If $r_{A,B} > 0$, A and B are positively correlated (A's values increase as B's). The higher, the stronger correlation.
- $r_{A,B} = 0$: independent; $r_{AB} < 0$: negatively correlated

Covariance (Numeric Data)

- Covariance is similar to correlation

$$Cov(A,B) = E((A-\bar{A})(B-\bar{B})) = \frac{\sum_{i=1}^{n}(a_i-\bar{A})(b_i-\bar{B})}{n}$$

- Correlation coefficient: $r_{A,B} = \dfrac{Cov(A,B)}{\sigma_A \sigma_B}$

  - **Positive covariance**: If $Cov_{A,B} > 0$, then A and B both tend to be larger than their expected values.
  - **Negative covariance**: If $Cov_{A,B} < 0$ then if A is larger than its expected value, B is likely to be smaller than its expected value.
  - **Independence**: $Cov_{A,B} = 0$ but the converse is not true!
    - Some pairs of random variables may have a covariance of 0 but are not independent.

Covariance: An Example

- It can be simplified:

$$r_{A,B} = \frac{Cov(A,B)}{\sigma_A \sigma_B} \Longrightarrow Cov(A,B) = E(A \cdot B) - \bar{A}\bar{B}$$

- Suppose two stocks A and B have the following values in one week: (2, 5), (3, 8), (5, 10), (4, 11), (6, 14).
- Question: If the stocks are affected by the same industry trends, will their prices rise or fall together?
  - E(A) = (2 + 3 + 5 + 4 + 6)/ 5 = 20/5 = 4
  - E(B) = (5 + 8 + 10 + 11 + 14) /5 = 48/5 = 9.6
  - Cov(A,B) = (2×5+3×8+5×10+4×11+6×14)/5 − 4 × 9.6 = 4
- Thus, A and B rise together since Cov(A, B) > 0.

Categorical Data Preprocessing

- **Nominal Attributes**
  - Categories, states, or "names of things"
  - *Hair_color = {auburn, black, blond, brown, grey, red, white}*
  - marital status, occupation, ID numbers, zip codes
  - Categorical data

- Categorical data handling
  - One-hot encoding: 카테고리 값을 attribute로 변환
  - Attribute의 갯수는 변환되는 categorical data의 distinct value의 갯수와 같아짐

Sample	Category	Numerical
1	Human	1
2	Human	1
3	Penguin	2
4	Octopus	3
5	Alien	4
6	Octopus	3
7	Alien	4

ONE-HOT encoding

Sample	Human	Penguin	Octopus	Alien
1	1	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	0	0	1	0
7	0	0	0	1

Data Reduction

- Obtain a reduced representation of the data set that is much smaller in volume but yet produces the same (or almost the same) analytical results
- Why? — A database/data warehouse may store terabytes of data. Complex data analysis may take a very long time to run on the complete data set.

- Data Reduction Strategies
  - Dimensionality reduction, e.g., remove unimportant attributes
    - Wavelet transforms
    - Principal Components Analysis (PCA)
    - Feature subset selection, feature creation
  - Numerosity reduction (some simply call it: Data Reduction)
    - Regression and Log-Linear Models
    - Histograms, clustering, sampling
    - Data cube aggregation
  - Data compression

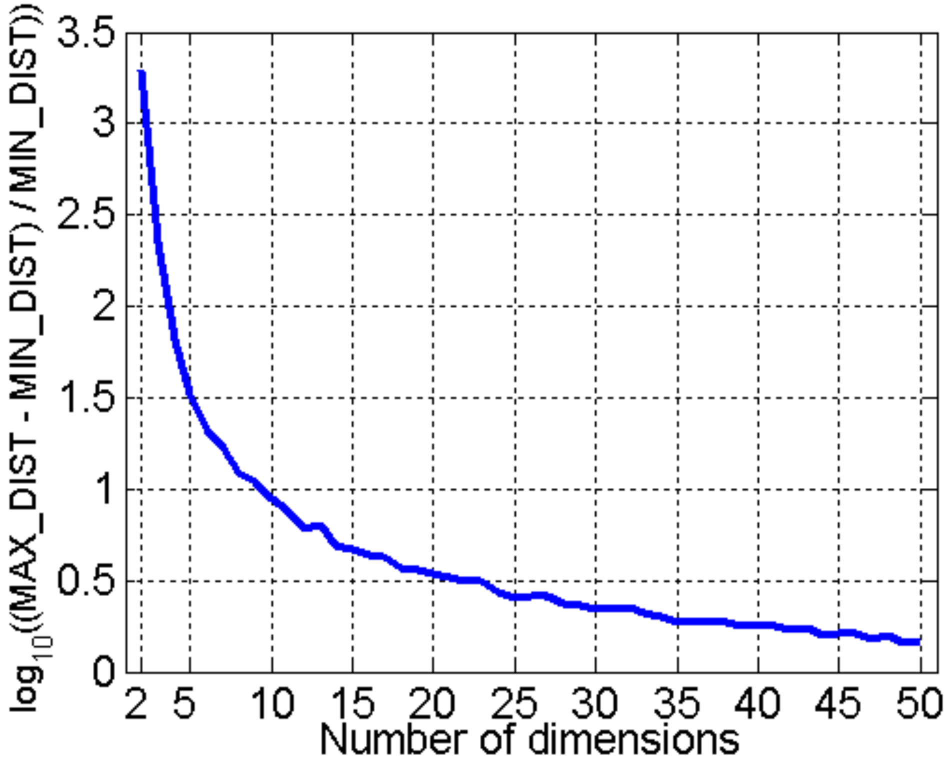
Dimensionality Reduction

- **Curse of dimensionality**
  - When dimensionality increases, data becomes increasingly sparse
  - Density and distance between points, which is critical to clustering, outlier analysis, becomes less meaningful
  - The possible combinations of subspaces will grow exponentially
- **Dimensionality reduction**
  - Avoid the curse of dimensionality
  - Help eliminate irrelevant features and reduce noise
  - Reduce time and space required in data mining
  - Allow easier visualization
- **Dimensionality reduction techniques**
  - Principal Component Analysis(PCA)
  - Singular Value Decomposition
  - Supervised and nonlinear techniques (e.g., feature selection)
  - Wavelet transforms

Curse of Dimensionality

- When dimensionality increases, data becomes increasingly sparse in the space that it occupies.

- Definitions of density and distance between points, which are critical for clustering and outlier detection, become less meaningful.



- Randomly generate 500 points
- Compute difference between max and min distance between any pair of points

Principal Component Analysis (PCA)

- The original data are projected onto a much smaller space, resulting in dimensionality reduction.
- Find the eigenvectors of the covariance matrix, and these eigenvectors define the new space.

- PCA usage:
  - Works for ordered and unordered attributes.
  - Can handle sparse data and skewed data.
  - n-D (n>2) data can be handled by reducing the problem to 2-D.
  - Works for numeric data only.

PCA Procedure

- Given N data vectors from n-dimensions, find $k \leq n$ orthogonal vectors (*principal components*) that can be best used to represent data
  - Normalize input data: Each attribute falls within the same range
  - Compute k orthonormal (unit) vectors, i.e., *principal components*
  - Each input data (vector) is a linear combination of the k principal component vectors
  - The principal components are sorted in order of decreasing "significance" or strength
  - Since the components are sorted, the size of the data can be reduced by eliminating the *weak components*, i.e., those with low variance (i.e., using the strongest principal components, it is possible to reconstruct a good approximation of the original data)

Attribute Subset Selection

- Another way to reduce dimensionality of data
- Redundant attributes
  - Duplicate much or all of the information contained in one or more other attributes
  - E.g., purchase price of a product and the amount of sales tax paid
- Irrelevant attributes
  - Contain no information that is useful for the data mining task at hand
  - E.g., students' ID is often irrelevant to the task of predicting students' GPA

Attribute Creation (Feature Generation)

- Create new attributes (features) that can capture the important information in a data set more effectively than the original ones

- General methodologies:
  - Attribute extraction
    - Example: extracting edges from images
  - Attribute construction
    - Example: dividing mass by volume to get density
  - Mapping data to new space
    - Example: Fourier and wavelet analysis

Numerosity Reduction

- Reduce data volume by choosing alternative, *smaller forms* of data representation
- **Parametric methods** (e.g., regression)
  - Assume the data fits some model, estimate model parameters, store only the parameters, and discard the data (except possible outliers)
- **Non-parametric** methods
  - Do not assume models
  - Major families: histograms, clustering, sampling, …

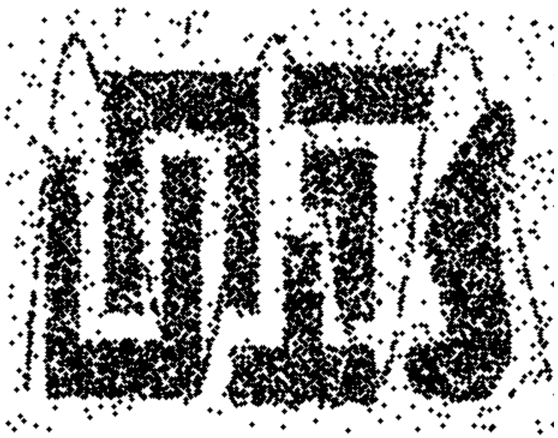
Clustering

- Partition data set into clusters based on similarity, and store cluster representation (e.g., centroid and diameter) only
- Can be very effective if data is clustered but not if data is "smeared"
- Can have hierarchical clustering and be stored in multi-dimensional index tree structures
- There are many choices of clustering definitions and clustering algorithms (will be studied later!)

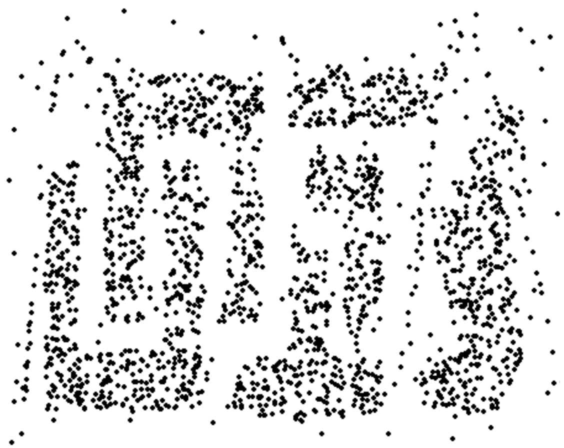
Sampling

- Sampling: obtaining a small sample *s* to represent the whole data set *N*
- Key principle: Choose a representative subset of the data
  - Simple random sampling may have very poor performance in the presence of skew
  - Develop adaptive sampling methods, e.g., stratified sampling
- Using a sample will work almost as well as using the entire data set, if the sample is representative
- A sample is representative if it has approximately the same properties (of interest) as the original set of data

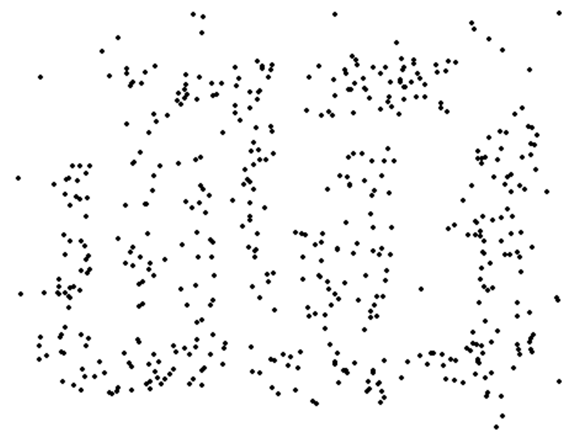
Sample Size



8000 points



2000 points

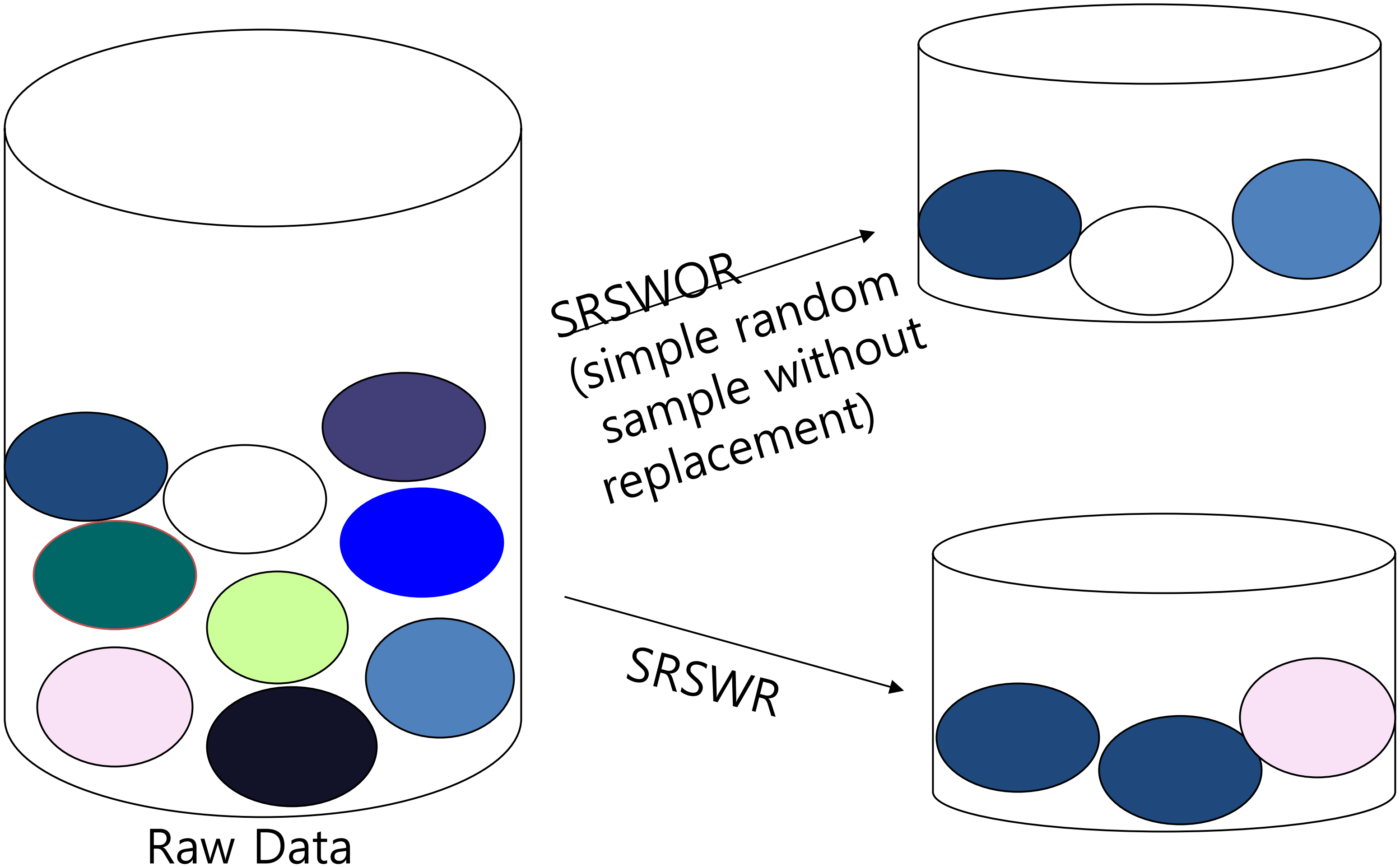


500 points

Types of Sampling

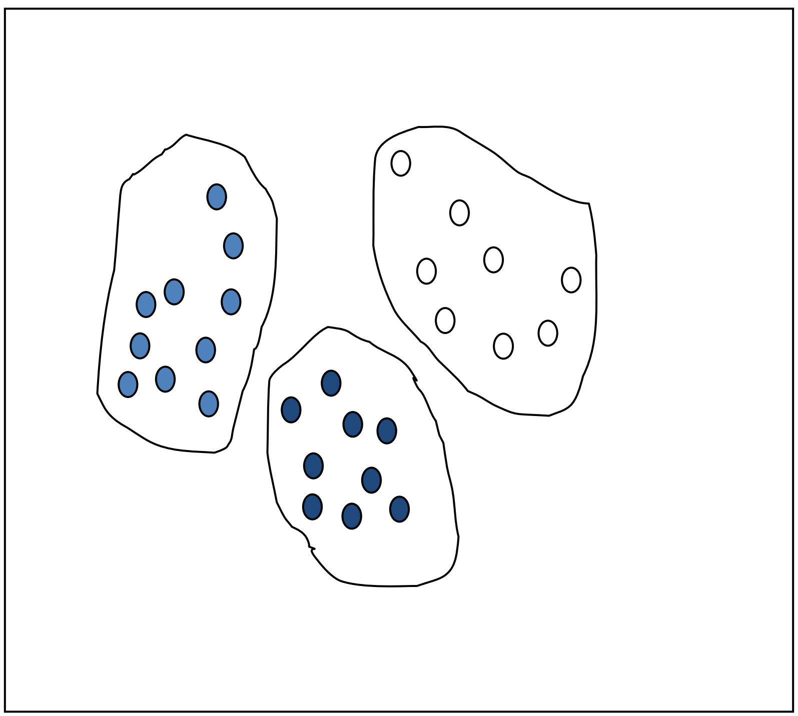
- **Simple random sampling**
  - There is an equal probability of selecting any particular item
- **Sampling without replacement**
  - Once an object is selected, it is removed from the population
- **Sampling with replacement**
  - A selected object is not removed from the population
- **Stratified sampling:**
  - Partition the data set, and draw samples from each partition (proportionally, i.e., approximately the same percentage of the data)
  - Used in conjunction with skewed data

Sampling: With or without Replacement

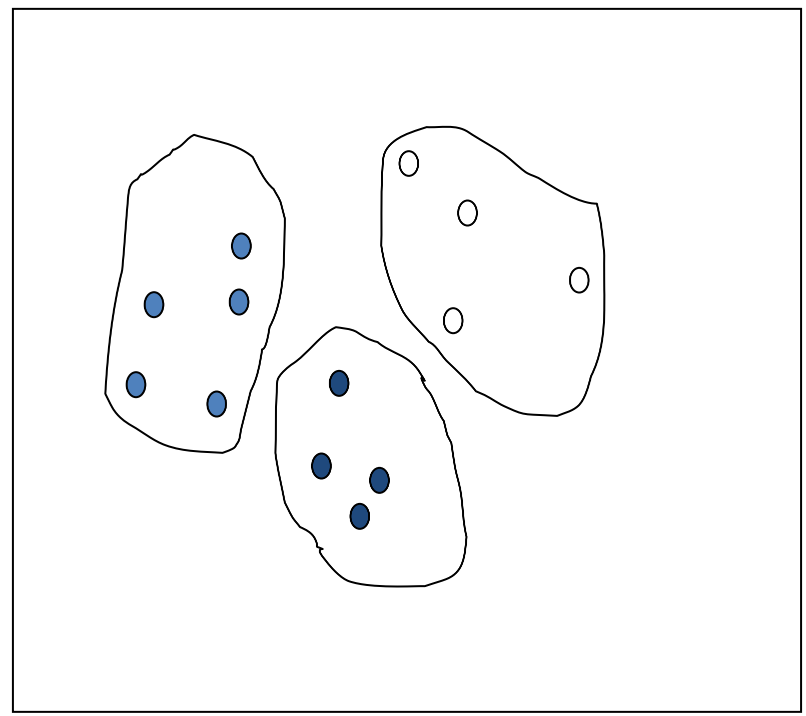


Sampling: Cluster or Stratified Sampling

Raw Data

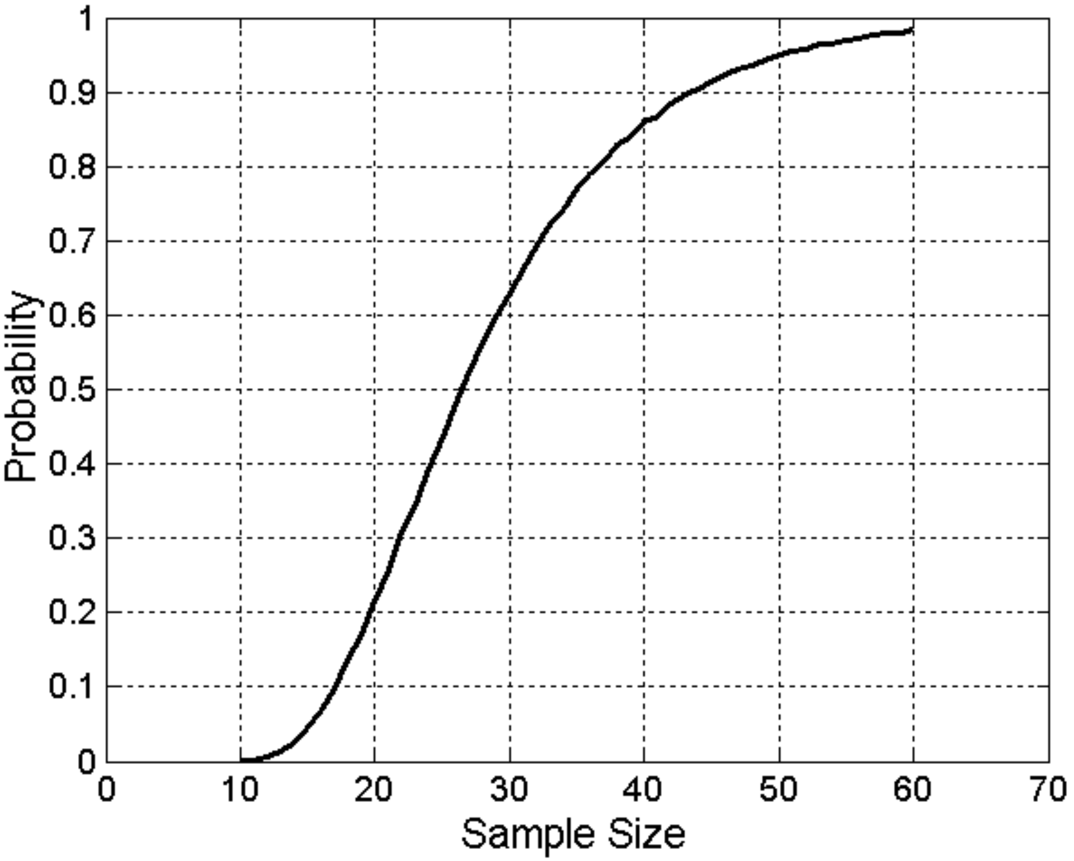


Cluster/Stratified Sample



Sample Size

- What sample size is necessary to get at least one object from each of 10 equal-sized groups



Data Transformation

- Data transformation methods
  - Smoothing: Remove noise from data
  - Attribute/feature construction
    - New attributes constructed from the given ones
  - Normalization: Scaled to fall within a smaller, specified range
    - min-max normalization
    - z-score normalization
    - normalization by decimal scaling
  - Discretization
  - Aggregation: Summarization, data cube construction

Normalization

- **Min-max normalization**: to [new_min$_A$, new_max$_A$]

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

  - Ex. Let income range \$12,000 to \$98,000 normalized to [0.0, 1.0]. Then \$73,000 is mapped to $\frac{73{,}600 - 12{,}000}{98{,}000 - 12{,}000}(1.0 - 0) + 0 = 0.716$

- **Z-score normalization** (μ: mean, σ: standard deviation):

$$v' = \frac{v - \mu_A}{\sigma_A}$$

  - Ex. Let μ = 54,000, σ = 16,000. Then $\frac{73{,}600 - 54{,}000}{16{,}000} = 1.225$

Discretization

- Discretization is the process of converting a continuous attribute into an ordinal attribute
  - A potentially infinite number of values are mapped into a small number of categories
  - Discretization is commonly used in classification
  - Many classification algorithms work best if both the independent and dependent variables have only a few values

- (Remember!) 3 types of attributes
  - Nominal—values from an unordered set, e.g., color, profession
  - Ordinal—values from an ordered set, e.g., military or academic rank
  - Numeric—real numbers, e.g., integer or real numbers

Discretization

- Divide the range of a continuous attribute into intervals
  – Interval labels can then be used to replace actual data values
  – Reduce data size by discretization
  – Supervised vs. unsupervised
  – Split (top-down) vs. merge (bottom-up)
  – Discretization can be performed recursively on an attribute
  – Prepare for further analysis, e.g., classification

Data Discretization Methods

- All the methods can be applied recursively
  - Binning
    - Top-down split, unsupervised
  - Histogram analysis
    - Top-down split, unsupervised
  - Clustering analysis
    - unsupervised, top-down split or bottom-up merge
  - Decision-tree analysis
    - supervised, top-down split
  - Correlation (e.g., χ^2) analysis
    - unsupervised, bottom-up merge

Binning

- Equal-width (distance) partitioning
  - Divides the range into N intervals of equal size: uniform grid
  - if A and B are the lowest and highest values of the attribute, the width of intervals will be: $W = (B - A)/N.$
  - The most straightforward, but outliers may dominate presentation
  - Skewed data is not handled well
- Equal-depth (frequency) partitioning
  - Divides the range into N intervals, each containing approximately same number of samples
  - Good data scaling
  - Managing categorical attributes can be tricky

Binning for Data Smoothing

- Sorted data for price (in dollars)
  4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34

* Partition into equal-frequency (**equi-depth**) bins:
  - Bin 1: 4, 8, 9, 15
  - Bin 2: 21, 21, 24, 25
  - Bin 3: 26, 28, 29, 34

* Smoothing by **bin means**:
  - Bin 1: 9, 9, 9, 9
  - Bin 2: 23, 23, 23, 23
  - Bin 3: 29, 29, 29, 29

* Smoothing by **bin boundaries**:
  - Bin 1: 4, 4, 4, 15
  - Bin 2: 21, 21, 25, 25
  - Bin 3: 26, 26, 26, 34

Discretization by Classification & Correlation Analysis

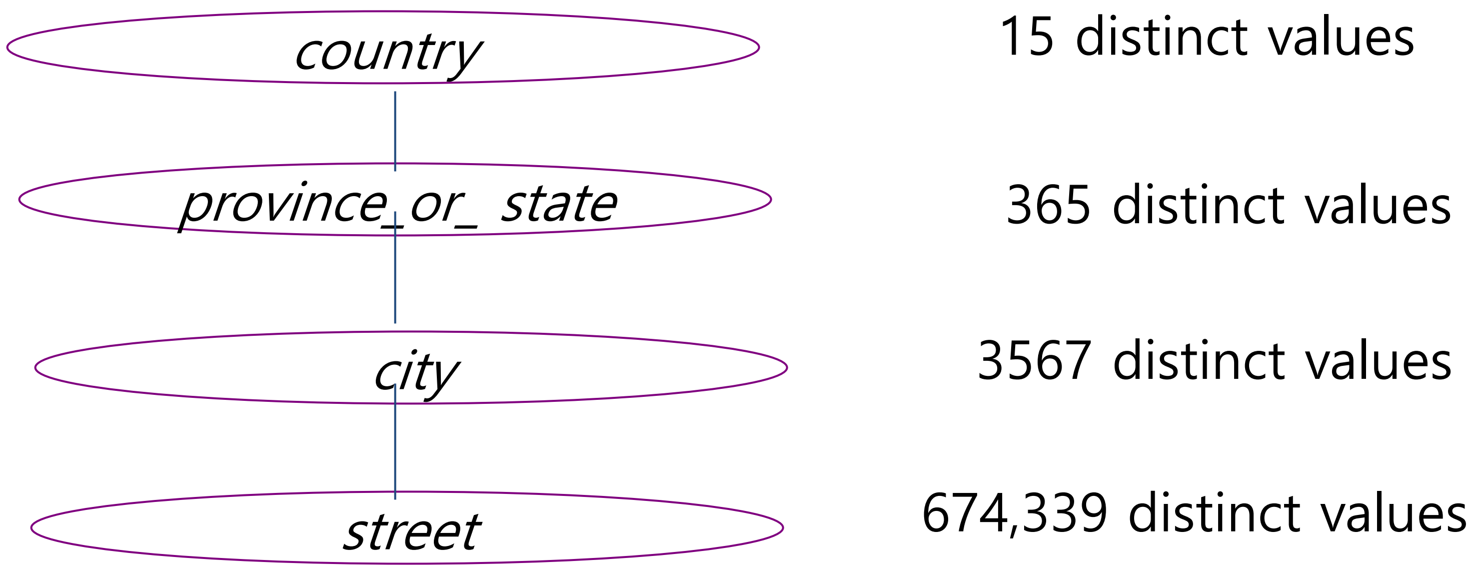
- Classification (e.g., decision tree analysis)
  - Supervised: Given class labels, e.g., cancerous vs. benign
  - Using *entropy* to determine split point (discretization point)
  - Top-down, recursive split (Details to be covered later!)
- Correlation analysis (e.g., Chi-merge: χ^2-based discretization)
  - Supervised: use class information
  - Bottom-up merge: find the best neighboring intervals (those having similar distributions of classes, i.e., low χ^2 values) to merge
  - Merge performed recursively, until a predefined stopping condition

Concept Hierarchy Generation for Nominal Data

- Specification of a partial/total ordering of attributes explicitly at the schema level by users or experts
  - *street* < *city* < *state* < *country*
- Specification of a hierarchy for a set of values by explicit data grouping
  - *{Urbana, Champaign, Chicago}* < *Illinois*
- Automatic generation of hierarchies (or attribute levels) by the analysis of the number of distinct values
  - E.g., for a set of attributes: {*street, city, state, country*}

Automatic Concept Hierarchy Generation

- Hierarchies can be automatically generated based on the analysis of the number of distinct values per attribute
  - The attribute with the most distinct values is placed at the lowest level of the hierarchy
  - Exceptions, e.g., weekday, month, quarter, year



Summary

- **Data quality**: accuracy, completeness, consistency, timeliness, believability, interpretability
- **Data cleansing**: e.g. missing/noisy values, outliers
- **Data integration** from multiple sources:
  - Entity identification problem
  - Remove redundancies
- **Data reduction**
  - Dimensionality reduction
  - Numerosity reduction
- **Data transformation and data discretization**
  - Normalization
  - Concept hierarchy generation

SCIKIT-LEARN

Scikit-learn

- 파이썬 머신러닝 라이브러리 중 가장 많이 사용됨
- 다양한 알고리즘 및 샘플 데이터 제공
- Pandas나 Numpy를 이용하면 편리하게 활용이 가능

Scikit-learn 설치

- pip install scipy

- 에러가 발생할 경우 다음 사이트에서 whl파일을 받아 설치
  - http://www.lfd.uci.edu/~gohlke/pythonlibs/#scipy
  - (python 3.5와 64bit에 맞추어야 함)
  - pip install *.whl

SciPy is software for mathematics, science, and engineering.

Install numpy+mkl before installing scipy.

scipy-0.19.1-cp27-cp27m-win32.whl
scipy-0.19.1-cp27-cp27m-win_amd64.whl
scipy-0.19.1-cp34-cp34m-win32.whl
scipy-0.19.1-cp34-cp34m-win_amd64.whl
scipy-0.19.1-cp35-cp35m-win32.whl
scipy-0.19.1-cp35-cp35m-win_amd64.whl
scipy-0.19.1-cp36-cp36m-win32.whl
scipy-0.19.1-cp36-cp36m-win_amd64.whl
scipy-1.0.0rc2-cp27-cp27m-win32.whl

Data Preprocessing

- Normalization
  - Min-max normalization (일반적으로 0~1 사이의 실수로 변환)

- Data transformation
  - E.g., 문자열로 구성된 attribute(feature)의 변환
  - Classification algorithm의 경우, 문자열을 허용하지 않는 경우가 존재하기 때문에, numeric 이나 one-hot encoding으로의 변환이 필요

- Missing data handling

Scaling

```
In [2]: import pandas as pd
        from sklearn.preprocessing import scale, minmax_scale

        x = pd.DataFrame({'col':[-3, -1, 1, 3, 5, 7, 9]})

        # 평균 0, 분산을 이용해 정규화
        # astype(float)는 scale의 입력이 float이므로 warning 방지를 위해 변환
        x["scale"] = scale(x.col.astype(float))

        # 0~1 사이의 값으로 정규화
        x["minmax_scale"] = minmax_scale(x.col.astype(float))

        print(x)
```

```
   col  scale  minmax_scale
0   -3   -1.5      0.000000
1   -1   -1.0      0.166667
2    1   -0.5      0.333333
3    3    0.0      0.500000
4    5    0.5      0.666667
5    7    1.0      0.833333
6    9    1.5      1.000000
```

Scaling

```
import pandas as pd
from sklearn.preprocessing import scale, minmax_scale

x = pd.DataFrame({'col':[-3, -1, 1, 3, 5, 7, 9]})

x["scale"] = scale(x.col.astype(float))

x.describe()
```

	col	scale
count	7.000000	7.000000
mean	3.000000	0.000000
std	4.320494	1.080123
min	-3.000000	-1.500000
25%	0.000000	-0.750000
50%	3.000000	0.000000
75%	6.000000	0.750000
max	9.000000	1.500000

```
import pandas as pd
from sklearn.preprocessing import scale, minmax_scale

x = pd.DataFrame({'col':[-3, -1, 1, 3, 5, 7, 9]})

x["minmax_scale"] = minmax_scale(x.col.astype(float))

x.describe()
```

	col	minmax_scale
count	7.000000	7.000000
mean	3.000000	0.500000
std	4.320494	0.360041
min	-3.000000	0.000000
25%	0.000000	0.250000
50%	3.000000	0.500000
75%	6.000000	0.750000
max	9.000000	1.000000

MinMaxScaler 객체 이용

- MinMaxScaler 객체를 활용하여 0~1사이의 값으로 스케일링

```
In [3]: import pandas as pd
        from sklearn.preprocessing import MinMaxScaler

        scaler = MinMaxScaler()

        dfTest = pd.DataFrame({'A':[14.00, 90.20, 90.95, 96.27, 91.21],
                               'B':[103.02, 107.26, 110.35, 114.23, 114.68],
                               'C':['big','small','big','small','small']})

        dfTest[['A','B']] = scaler.fit_transform(dfTest[['A','B']])

        print(dfTest)
```

```
          A         B      C
0  0.000000  0.000000    big
1  0.926219  0.363636  small
2  0.935335  0.628645    big
3  1.000000  0.961407  small
4  0.938495  1.000000  small
```

Nominal Attributes

- Nominal Attribute (“names of things”) 처리
  - Scikit-learn에 preprocessing 라이브러리가 존재함
  - E.g., 도시 명 ["paris", "paris", "tokyo", "amsterdam"]

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()

le.fit(["paris", "paris", "tokyo", "amsterdam"])
print(le.classes_)
print(type(le.classes_), "\n")

data = le.transform(["paris", "paris", "tokyo", "amsterdam"])
print(data)
print(type(data), "\n")

original = le.inverse_transform([2, 2, 1])
print(original)
print(type(data))
```

```
['amsterdam' 'paris' 'tokyo']
<class 'numpy.ndarray'>

[1 1 2 0]
<class 'numpy.ndarray'>

['tokyo' 'tokyo' 'paris']
<class 'numpy.ndarray'>
```

Transform 예제

- 영어 소문자 transformation

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()

str = []
for i in range(ord('a'), ord('z') + 1): # ord('a'): 'a'의 ascii code
    str.append(chr(i))                  # chr(i): ascii code i에 해당하는 문자
print(str)

le.fit(str)
data = le.transform(['q', 'a', 'z'])
print(data)
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n',
'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
[16  0 25]
```

DataFrame 변환

```
from sklearn import preprocessing
import pandas as pd

le = preprocessing.LabelEncoder()
df = pd.DataFrame({'A': ['a', 'b', 'b', 'c', 'a'],
                   'B': ['x', 'y', 'x', 'y', 'x']})

# fit_transform: fit과 transform을 동시에 처리
# df.apply는 dataframe에서 인자로 주어진 함수를 각 Column에 적용하는 함수
data = df.apply(le.fit_transform)
print(data)
print(type(data), "\n")
```

```
   A  B
0  0  0
1  1  1
2  1  0
3  2  1
4  0  0
<class 'pandas.core.frame.DataFrame'>
```

One-hot encoding

- Process by which categorical variables are converted into a form that could be provided to ML algorithms
- 카테고리 값을 컬럼으로 변환
  - 컬럼의 갯수는 카테고리 값의 종류수와 같아짐

Sample	Category	Numerical
1	Human	1
2	Human	1
3	Penguin	2
4	Octopus	3
5	Alien	4
6	Octopus	3
7	Alien	4

ONE-HOT encoding

Sample	Human	Penguin	Octopus	Alien
1	1	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	0	0	1	0
7	0	0	0	1

Pandas를 활용한 One-hot encoding

- Pandas.get_dummies
  - Categorical variable를 dummy/indicator 변수로 변환

```
import pandas as pd

df = pd.DataFrame({'country':['russia', 'germany', 'australia', 'korea', 'germany']})
a = pd.get_dummies(df, prefix = ['country'])
print(a)
```

```
   country_australia  country_germany  country_korea  country_russia
0                  0                0              0               1
1                  0                1              0               0
2                  1                0              0               0
3                  0                0              1               0
4                  0                1              0               0
```

Pandas를 활용한 One-hot encoding

```
import pandas as pd

df = pd.DataFrame({'A': ['a', 'b', 'a'], 'B':['b', 'a', 'c']})

# Get one-hot encoding of columns B
one_hot = pd.get_dummies(df['B'])

# Drop column B as it is now encoded
df = df.drop('B', axis = 1)

# Join the encoded df
df = df.join(one_hot)

print(df)
```

```
   A  a  b  c
0  a  0  1  0
1  b  1  0  0
2  a  0  0  1
```

Multiple column 변환

```
import pandas as pd

df = pd.DataFrame({'A': ['a', 'b', 'b', 'c', 'a'],
                   'B': ['x', 'y', 'x', 'y', 'x']})
a = pd.get_dummies(df, prefix = ['A', 'B'])
print(a)
```

	A_a	A_b	A_c	B_x	B_y
0	1	0	0	1	0
1	0	1	0	0	1
2	0	1	0	1	0
3	0	0	1	0	1
4	1	0	0	1	0

일부만을 자동으로 인식 encoding

- Encoding이 필요한 부분만을 자동으로 인식

```
import pandas as pd

df = pd.DataFrame({'A': ['a', 'b', 'b', 'c', 'a'],
                   'B': [3, 4, 7, 2, 5]})
a = pd.get_dummies(df, prefix = ['A'])
print(a)
```

```
   B  A_a  A_b  A_c
0  3    1    0    0
1  4    0    1    0
2  7    0    1    0
3  2    0    0    1
4  5    1    0    0
```

'B'까지 넣어주면 error 발생
'B'는 encoding이 필요 없음

One-hot encoding with Scikit-learn

- LabelEncoder()와 OneHotEncoder()를 모두 이용해야 함

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder

x1 = pd.DataFrame({'country':['russia', 'germany', 'australia', 'korea', 'germany']})

# DataFrame 전체를 라벨인코딩(숫자로 변환) 한 후, one-hot encoding을 해야 함
le = LabelEncoder()
x2 = x1.apply(le.fit_transform)
print(x2)
print(type(x2))

encoder = OneHotEncoder()
x2 = encoder.fit_transform(x2) # 결과는 sparse matrix로 변환됨
print(x2)
print(type(x2))

x3 = x2.toarray() # numpy array로 변화, 추후에 DataFrame으로 변환
print(x3)
print(type(x3))
```

One-hot encoding with Scikit-learn

```
   country
0        3
1        1
2        0
3        2
4        1
<class 'pandas.core.frame.DataFrame'>
  (0, 3)	1.0
  (1, 1)	1.0
  (2, 0)	1.0
  (3, 2)	1.0
  (4, 1)	1.0
<class 'scipy.sparse.csr.csr_matrix'>
[[0. 0. 0. 1.]
 [0. 1. 0. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 [0. 1. 0. 0.]]
<class 'numpy.ndarray'>
```

3번째 position 값이 1이라는 의미

One-hot encoding with Scikit-learn

- OneHotEncoder(categories = ‘auto’)를 사용
- LabelEncoder 사용이 불필요함

In [21]:
```
import pandas as pd

from sklearn.preprocessing import OneHotEncoder

x1 = pd.DataFrame({'country': ['russia', 'germany', 'australia', 'korea', 'germany']})

encoder = OneHotEncoder(categories = 'auto')
x2 = encoder.fit_transform(x1)
print(x2)
print(type(x2))
```

```
  (0, 3)	1.0
  (1, 1)	1.0
  (2, 0)	1.0
  (3, 2)	1.0
  (4, 1)	1.0
<class 'scipy.sparse.csr.csr_matrix'>
```

One-hot encoding with Scikit-learn

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder

x1 = pd.DataFrame({'A': ['a', 'b', 'b', 'c', 'a'],
                   'B': ['x', 'y', 'x', 'y', 'x']})

# DataFrame 전체를 라벨인코딩(숫자로 변환) 한 후, one-hot encoding을 해야 함
le = LabelEncoder()
x2 = x1.apply(le.fit_transform)
print(x2)
print(type(x2))

encoder = OneHotEncoder(categories='auto')
x2 = encoder.fit_transform(x2) # 결과는 sparse matrix로 변환됨
print(x2)
print(type(x2))

x3 = x2.toarray() # numpy array로 변환, 추후에 DataFrame으로 변환
print(x3)
print(type(x3))

x4 = pd.DataFrame(x3) # 최종적으로 다시 DataFrame으로 변환
print(x4)
print(type(x4))
```

One-hot encoding with Scikit-learn

```
   A  B
0  0  0
1  1  1
2  1  0
3  2  1
4  0  0
<class 'pandas.core.frame.DataFrame'>
  (0, 0)	1.0
  (0, 3)	1.0
  (1, 1)	1.0
  (1, 4)	1.0
  (2, 1)	1.0
  (2, 3)	1.0
  (3, 2)	1.0
  (3, 4)	1.0
  (4, 0)	1.0
  (4, 3)	1.0
<class 'scipy.sparse.csr.csr_matrix'>
[[1. 0. 0. 1. 0.]
 [0. 1. 0. 0. 1.]
 [0. 1. 0. 1. 0.]
 [0. 0. 1. 0. 1.]
 [1. 0. 0. 1. 0.]]
<class 'numpy.ndarray'>
     0    1    2    3    4
0  1.0  0.0  0.0  1.0  0.0
1  0.0  1.0  0.0  0.0  1.0
2  0.0  1.0  0.0  1.0  0.0
3  0.0  0.0  1.0  0.0  1.0
4  1.0  0.0  0.0  1.0  0.0
<class 'pandas.core.frame.DataFrame'>
```

일부 컬럼만 One-hot encoding이 필요한 경우

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder

x = pd.DataFrame({'A': ['a', 'b', 'b', 'c', 'a'],
                  'B': [3, 4, 5, 1, 7]})

# DataFrame 전체를 라벨인코딩(숫자로 변환) 한 후, one-hot encoding을 해야 함
le = LabelEncoder()
x.A = le.fit_transform(x.A)
# x.A는 series이므로 apply  함수를 적용하지 않고 fit_transform의 인자로 사용
print(x)

# one-hot encoding이 필요한 column만 지정
encoder = OneHotEncoder(categorical_features = [True, False])

x2 = encoder.fit_transform(x) # 결과는 sparse matrix로 변환됨
print(x2)
print(type(x2))

x3 = x2.toarray() # numpy array로 변환, 추후에 DataFrame으로 변환
print(x3)
print(type(x3))

x4 = pd.DataFrame(x3) # 최종적으로 다시 DataFrame으로 변환
print(x4)
print(type(x4))
```

일부 컬럼만 One-hot encoding이 필요한 경우

```
   A  B
0  0  3
1  1  4
2  1  5
3  2  1
4  0  7
  (0, 0)	1.0
  (1, 1)	1.0
  (2, 1)	1.0
  (3, 2)	1.0
  (4, 0)	1.0
  (0, 3)	3.0
  (1, 3)	4.0
  (2, 3)	5.0
  (3, 3)	1.0
  (4, 3)	7.0
<class 'scipy.sparse.coo.coo_matrix'>
[[1. 0. 0. 3.]
 [0. 1. 0. 4.]
 [0. 1. 0. 5.]
 [0. 0. 1. 1.]
 [1. 0. 0. 7.]]
<class 'numpy.ndarray'>
     0    1    2    3
0  1.0  0.0  0.0  3.0
1  0.0  1.0  0.0  4.0
2  0.0  1.0  0.0  5.0
3  0.0  0.0  1.0  1.0
4  1.0  0.0  0.0  7.0
<class 'pandas.core.frame.DataFrame'>
```

ColumnTransformer*

- ColumnTransformer 활용 one-hot encoding을 수행(권장)

```
In [87]: import pandas as pd

         from sklearn.preprocessing import OneHotEncoder
         from sklearn.compose import ColumnTransformer

         x = pd.DataFrame({'A': ['a', 'b', 'b', 'c', 'a'],
                           'B': [3, 4, 5, 1, 7]})

         ct = ColumnTransformer(
             [('one_hot_encoder', OneHotEncoder(), [0])], remainder = 'passthrough'

         )

         x = ct.fit_transform(x)
         print(x)
         print(type(x))
         print()
```

```
[[1. 0. 0. 3.]
 [0. 1. 0. 4.]
 [0. 1. 0. 5.]
 [0. 0. 1. 1.]
 [1. 0. 0. 7.]]
<class 'numpy.ndarray'>
```

*https://scikit-learn.org/stable/modules/generated/sklearn.compose.ColumnTransformer.html

Handling Missing Values

- Mean 또는 median 값으로 대치
- Missing value가 존재하는 row를 제거하고 분석
- CSV 파일의 경우, missing value를 NaN으로 처리
  - 단, missing value 대신에 0을 사용하는 경우도 있으므로 주의

Handling Missing Values

```
import pandas as pd

dataset = pd.read_csv('pima-indians-diabetes.csv', header = None)
print(dataset[0:10], "\n")
print((dataset == 0).sum())
```

```
    0    1   2   3    4     5      6   7  8
0   6  148  72  35    0  33.6  0.627  50  1
1   1   85  66  29    0  26.6  0.351  31  0
2   8  183  64   0    0  23.3  0.672  32  1
3   1   89  66  23   94  28.1  0.167  21  0
4   0  137  40  35  168  43.1  2.288  33  1
5   5  116  74   0    0  25.6  0.201  30  0
6   3   78  50  32   88  31.0  0.248  26  1
7  10  115   0   0    0  35.3  0.134  29  0
8   2  197  70  45  543  30.5  0.158  53  1
9   8  125  96   0    0   0.0  0.232  54  1

0    111
1      5
2     35
3    227
4    374
5     11
6      0
7      0
8    500
dtype: int64
```

NaN로 변환

```
import pandas as pd
import numpy as np

dataset = pd.read_csv('pima-indians-diabetes.csv', header = None)

# 컬럼 1~5까지의 0값을 NaN으로 치환하고 결과를 dataset에 저장
dataset[[1, 2, 3, 4, 5]] = dataset[[1, 2, 3, 4, 5]].replace(0, np.NaN)

# print first 10 rows of data
print(dataset.head(10))
```

	0	1	2	3	4	5	6	7	8
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50	1
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31	0
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1
5	5	116.0	74.0	NaN	NaN	25.6	0.201	30	0
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26	1
7	10	115.0	NaN	NaN	NaN	35.3	0.134	29	0
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53	1
9	8	125.0	96.0	NaN	NaN	NaN	0.232	54	1

(Missing Value를 포함하는) 행 삭제

- Use dropna() to remove all rows with missing data

```
import pandas as pd
import numpy as np

dataset = pd.read_csv('pima-indians-diabetes.csv', header = None)

# 컬럼 1~5까지의 0값을 NaN으로 치환하고 결과를 dataset에 저장
dataset[[1, 2, 3, 4, 5]] = dataset[[1, 2, 3, 4, 5]].replace(0, np.NaN)
print(dataset.shape)

# drop rows with missing values
dataset.dropna(inplace = True)

# summarize the number of rows and columns in the dataset
print(dataset.shape)
```

```
(768, 9)
(392, 9)
```

다른 값으로 대체

- Pandas provides the fillna() function for replacing missing values with a specific value

```
import pandas as pd
import numpy as np

dataset = pd.read_csv('pima-indians-diabetes.csv', header = None)
dataset[[1, 2, 3, 4, 5]] = dataset[[1, 2, 3, 4, 5]].replace(0, np.NaN)

# column별 mean을 출력하여 값을 확인한 후에 대체
print(dataset.mean())

# fill missing values with mean column values
dataset.fillna(dataset.mean(), inplace = True)

# count the number of NaN values in each column
print(dataset.isnull().sum())

# print first 10 rows of data
print(dataset.head(10))
```

다른 값으로 대체

```
0      3.845052
1    121.686763
2     72.405184
3     29.153420
4    155.548223
5     32.457464
6      0.471876
7     33.240885
8      0.348958
dtype: float64
0    0
1    0
2    0
3    0
4    0
5    0
6    0
7    0
8    0
dtype: int64
```

	0	1	2	3	4	5	6	7	8
0	6	148.0	72.000000	35.00000	155.548223	33.600000	0.627	50	1
1	1	85.0	66.000000	29.00000	155.548223	26.600000	0.351	31	0
2	8	183.0	64.000000	29.15342	155.548223	23.300000	0.672	32	1
3	1	89.0	66.000000	23.00000	94.000000	28.100000	0.167	21	0
4	0	137.0	40.000000	35.00000	168.000000	43.100000	2.288	33	1
5	5	116.0	74.000000	29.15342	155.548223	25.600000	0.201	30	0
6	3	78.0	50.000000	32.00000	88.000000	31.000000	0.248	26	1
7	10	115.0	72.405184	29.15342	155.548223	35.300000	0.134	29	0
8	2	197.0	70.000000	45.00000	543.000000	30.500000	0.158	53	1
9	8	125.0	96.000000	29.15342	155.548223	32.457464	0.232	54	1

END