# DATA PREPROCESSING (PART II)

## Things to consider

- Data types & data exploration
- Data quality issues
- Data transformation – preprocessing

## Application Related Issues

- Timeliness:
  - Some data age quickly after it is collected
  - Example: Purchasing behavior of online shoppers
  - All patterns/models based on expired data are out of date
  - Quick response time: fraudulent transactions
- Relevance:
  - Data must contain relevant information to the application
- Knowledge about the data
  - Correlated attributes
  - Special values (Example: 9999 for unavailable)
  - Attribute types (Example: Categorical {1, 2, 3})

## Data Preprocessing

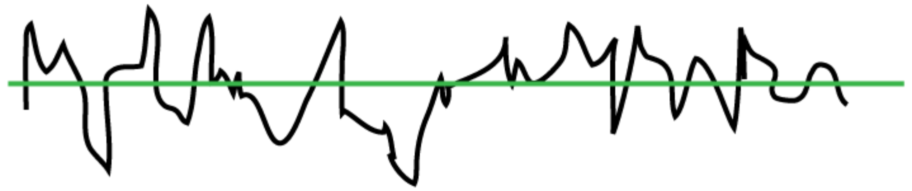
- Aggregation
- Sampling
- Discretization
- Dimensionality reduction
- Variable transformation

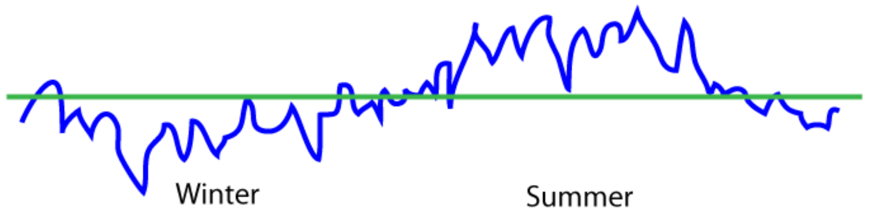
## Aggregation

- Aggregation: combine multiple objects into one
  - Reduce number of objects: less memory, less processing time
  - Quantitative attributes are combined using sum or average
  - Qualitative attributes are omitted or listed as a set
  - Reduces variability

- Example: Daily sales aggregated to seasonal sales

## Aggregation – Weather vs. Climate

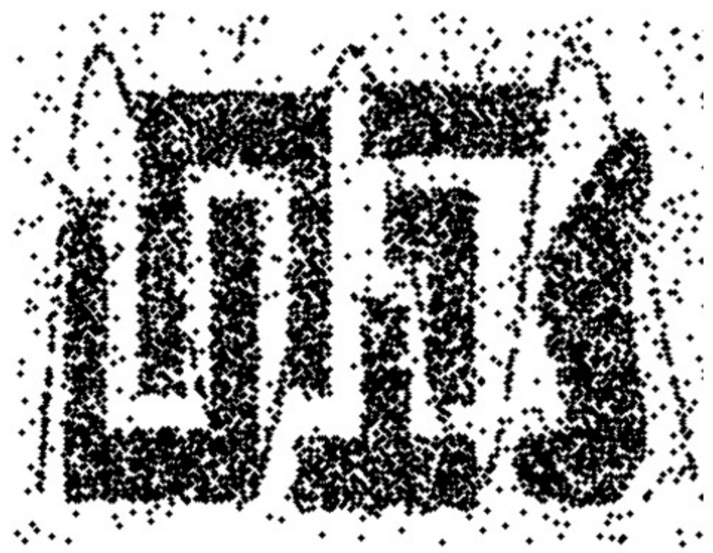


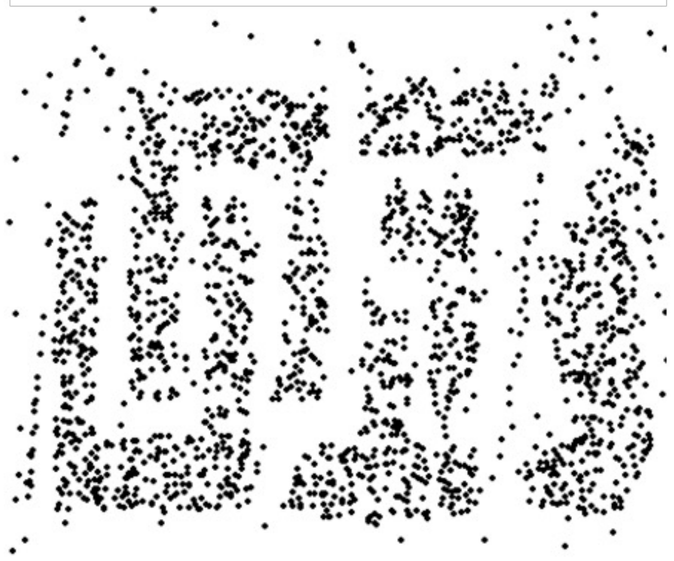


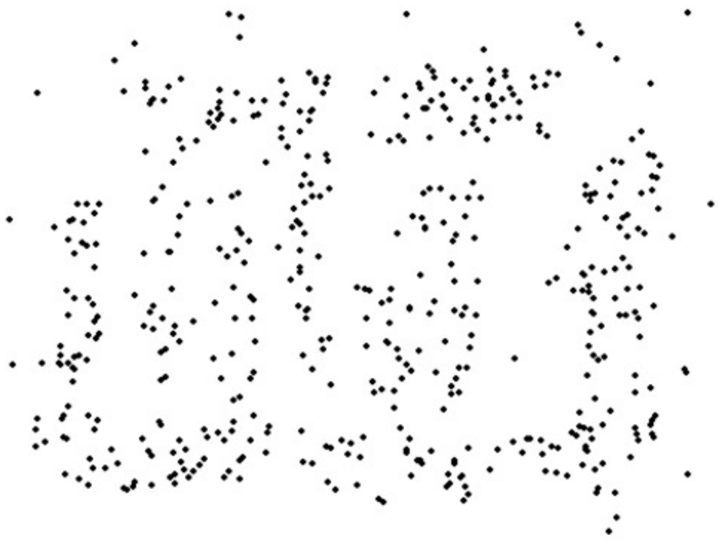
- **Weather**: condition of atmosphere (temperature, pressure, humidity) at particular time and place
- **Climate**: Average weather of an area over long period of time

# Sampling

- Select a subset of objects to be analyzed
- Representative sample: same properties as the original set of data


8000 points


2000 Points


500 Points

## Why use sampling

- Data does not fit in memory

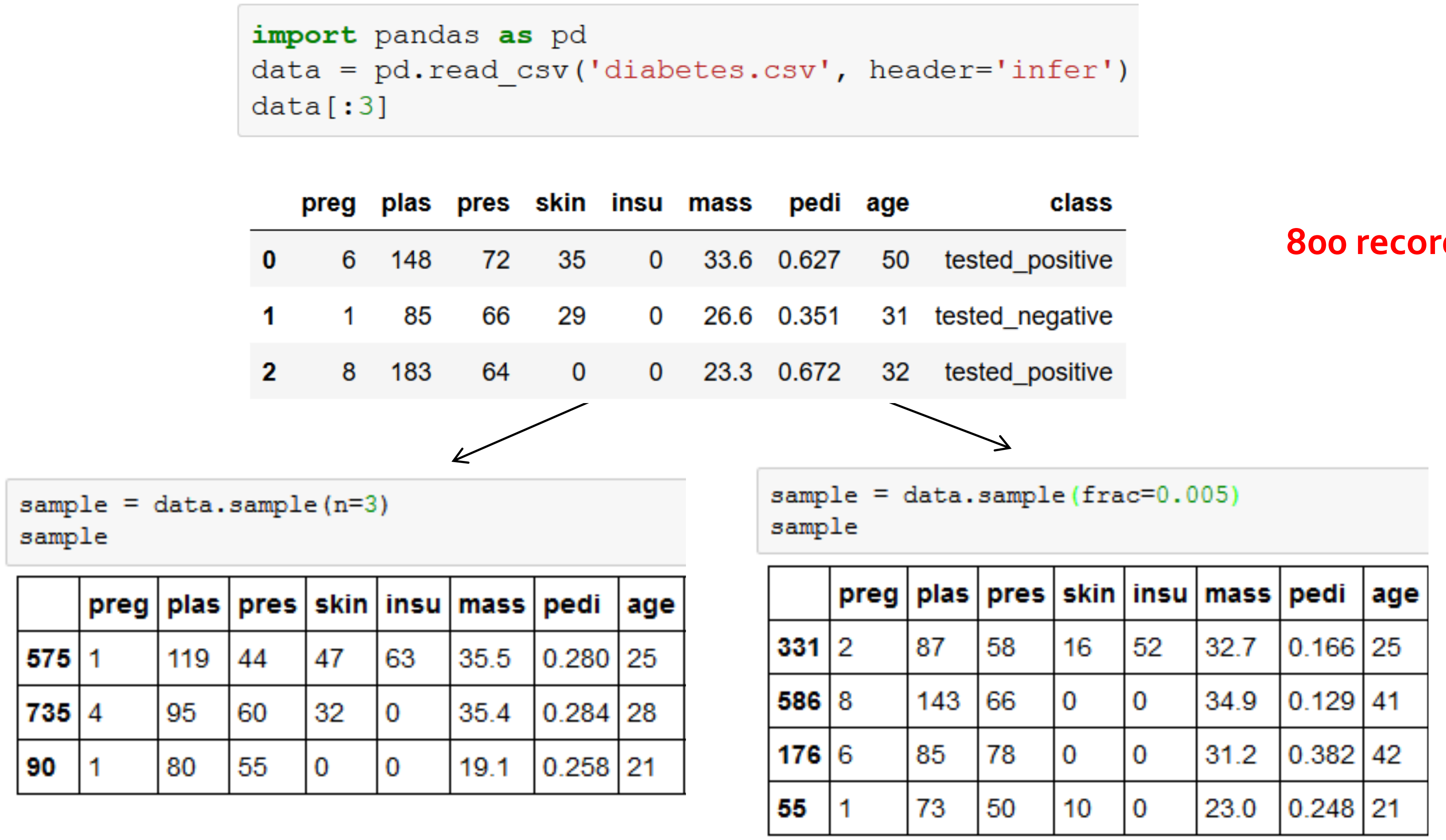
- Speed up model training time

- Class imbalance – Interest in rare events
  - Oversampling
  - Reduce costs of misclassification
    - Fraudulent transactions flagged as normal
    - Selling defective equipment

## Sampling Approaches

- Simple random sampling: equal probability of selecting object
  - Sampling without replacement
  - Sampling with replacement
- Stratified sampling: sample objects by group
  - Sample equally from each group
  - Sample proportional to size of groups
- Progressive Sampling: start with small sample, increase sample size
  - Stop when predictive model performance does not change significantly

## Python Example



```
import pandas as pd
data = pd.read_csv('diabetes.csv', header='infer')
data[:3]
```

| | preg | plas | pres | skin | insu | mass | pedi | age | class |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | tested_positive |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | tested_negative |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | tested_positive |

**800 records**

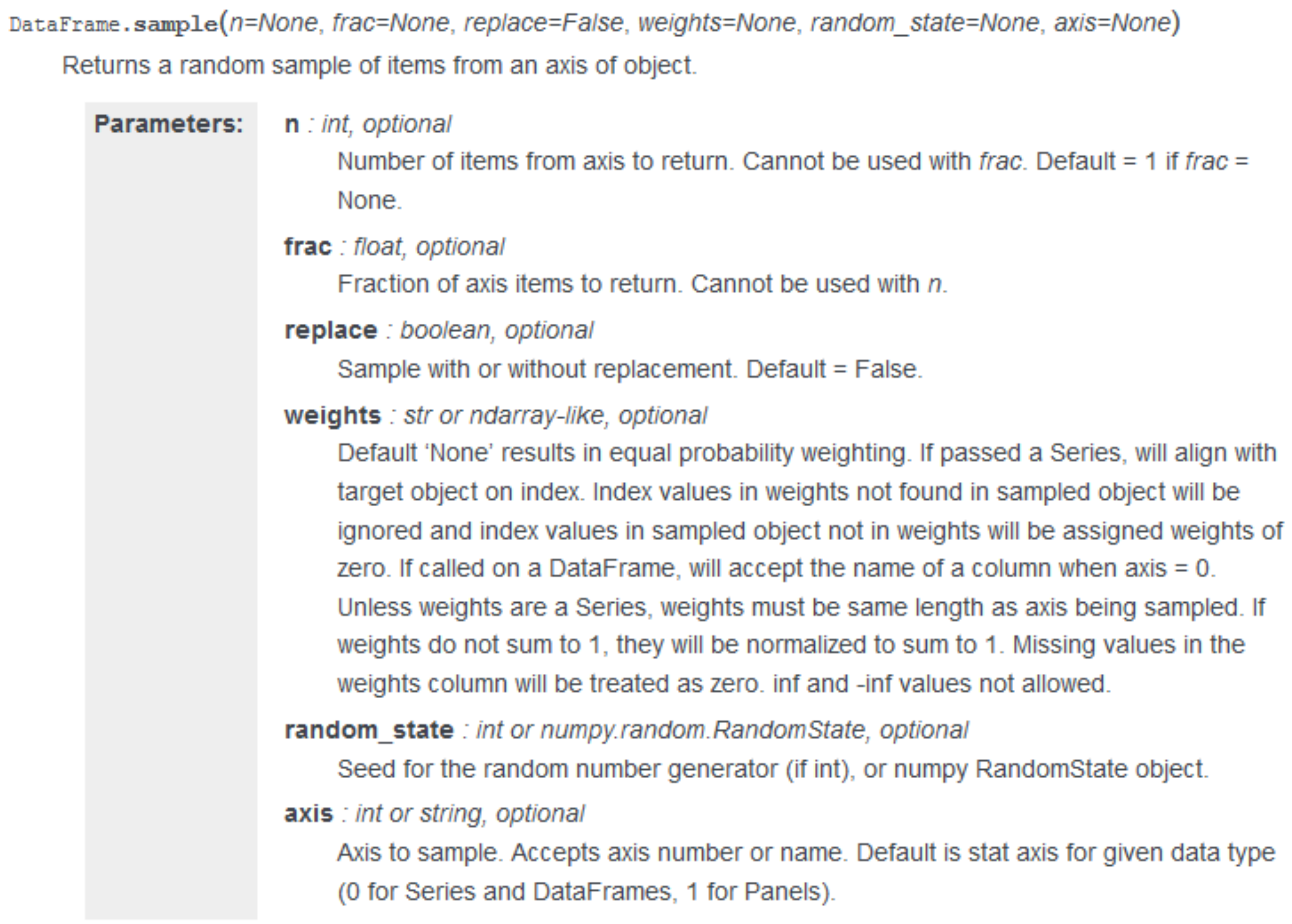
```
sample = data.sample(n=3)
sample
```

| | preg | plas | pres | skin | insu | mass | pedi | age |
|---|---|---|---|---|---|---|---|---|
| **575** | 1 | 119 | 44 | 47 | 63 | 35.5 | 0.280 | 25 |
| **735** | 4 | 95 | 60 | 32 | 0 | 35.4 | 0.284 | 28 |
| **90** | 1 | 80 | 55 | 0 | 0 | 19.1 | 0.258 | 21 |

```
sample = data.sample(frac=0.005)
sample
```

| | preg | plas | pres | skin | insu | mass | pedi | age |
|---|---|---|---|---|---|---|---|---|
| **331** | 2 | 87 | 58 | 16 | 52 | 32.7 | 0.166 | 25 |
| **586** | 8 | 143 | 66 | 0 | 0 | 34.9 | 0.129 | 41 |
| **176** | 6 | 85 | 78 | 0 | 0 | 31.2 | 0.382 | 42 |
| **55** | 1 | 73 | 50 | 10 | 0 | 23.0 | 0.248 | 21 |

# DataFrame .sample()



`DataFrame.sample`(*n=None, frac=None, replace=False, weights=None, random_state=None, axis=None*)

Returns a random sample of items from an axis of object.

| **Parameters:** | |
|---|---|
| | **n** : *int, optional*<br>Number of items from axis to return. Cannot be used with *frac*. Default = 1 if *frac* = None. |
| | **frac** : *float, optional*<br>Fraction of axis items to return. Cannot be used with *n*. |
| | **replace** : *boolean, optional*<br>Sample with or without replacement. Default = False. |
| | **weights** : *str or ndarray-like, optional*<br>Default 'None' results in equal probability weighting. If passed a Series, will align with target object on index. Index values in weights not found in sampled object will be ignored and index values in sampled object not in weights will be assigned weights of zero. If called on a DataFrame, will accept the name of a column when axis = 0. Unless weights are a Series, weights must be same length as axis being sampled. If weights do not sum to 1, they will be normalized to sum to 1. Missing values in the weights column will be treated as zero. inf and -inf values not allowed. |
| | **random_state** : *int or numpy.random.RandomState, optional*<br>Seed for the random number generator (if int), or numpy RandomState object. |
| | **axis** : *int or string, optional*<br>Axis to sample. Accepts axis number or name. Default is stat axis for given data type (0 for Series and DataFrames, 1 for Panels). |

# Discretization

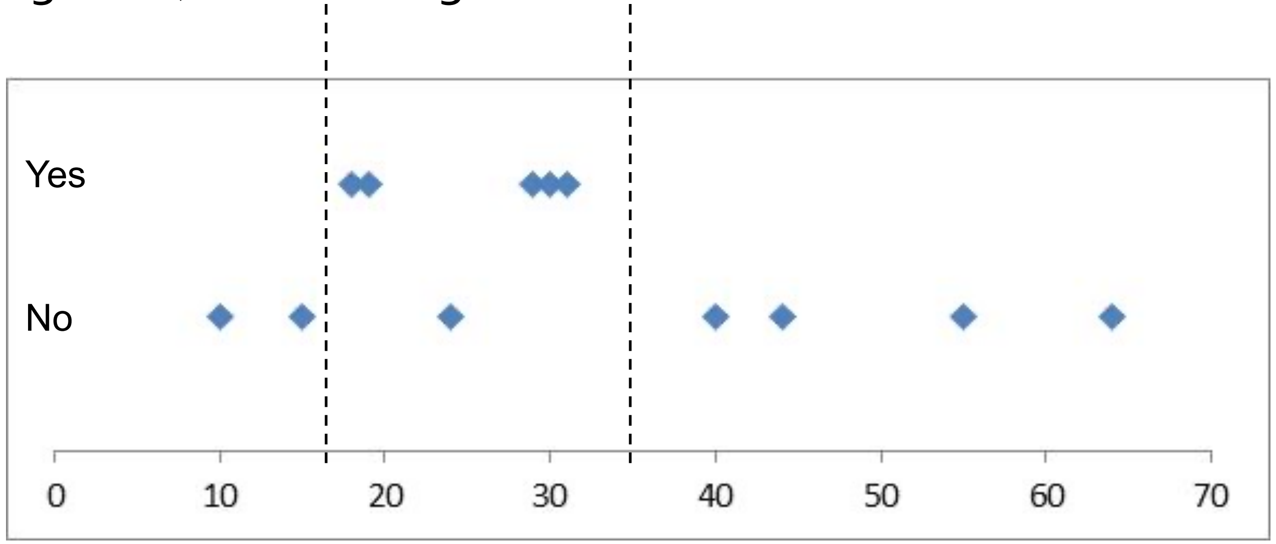
- Discretization is used to split the range of a numeric attribute into discrete number of intervals

- Why do we need discretization?
  - Makes the data more interpretable
    - Instead of using raw age values, discretize it into [child, teen, adult, senior] makes it easier for humans to interpret
  - Makes the results of analysis become more apparent
    - Example: association analysis

# Example

Suppose you have a dataset about online visitors who buy a specific product (e.g., computer games) and their age

| Age | Buy |
|---|---|
| 10 | No |
| 15 | No |
| 18 | Yes |
| 19 | Yes |
| 24 | No |
| 29 | Yes |
| 30 | Yes |
| 31 | Yes |
| 40 | No |
| 44 | No |
| 55 | No |
| 64 | No |



Association rule: (Age $\in$ [17, 35]) => (Buy = Yes)

Rule may not be apparent if you use the raw age values

## Unsupervised Discretization

- Equal interval width
  - Split the range of the numeric attribute into equal length intervals (bins)
  - Pros: cheap and easy to implement
  - Cons: susceptible to outliers

- Equal frequency
  - Split the range of the numeric attribute in such a way that each interval (bin) has the same number of points
  - Pros: robust to outliers
  - Cons: more expensive (must sort data), may not be consistent with inherent structure of the data

## Python Example

|  | preg | plas | pres | skin | insu | mass | pedi | age | class |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | tested_positive |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | tested_negative |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | tested_positive |

diabetes.csv

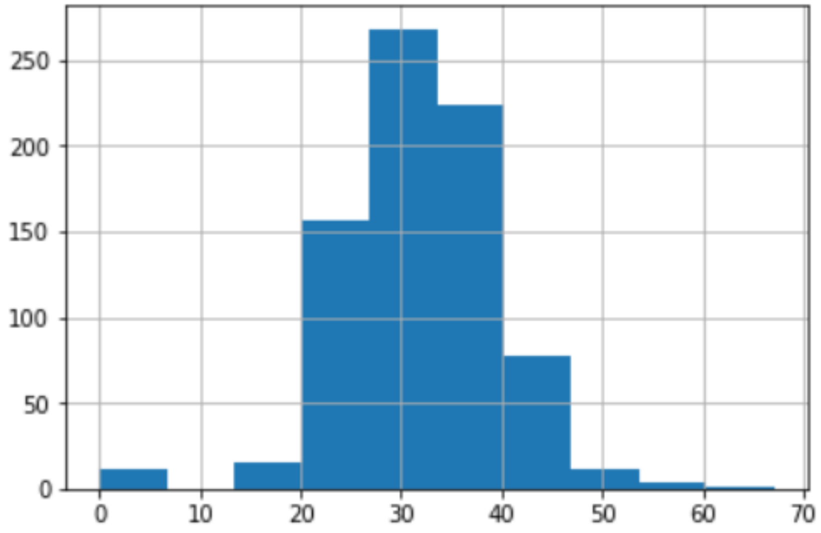
```
data.mass.describe()
```

```
count    768.000000
mean      31.992578
std        7.884160
min        0.000000
25%       27.300000
50%       32.000000
75%       36.600000
max       67.100000
Name: mass, dtype: float64
```

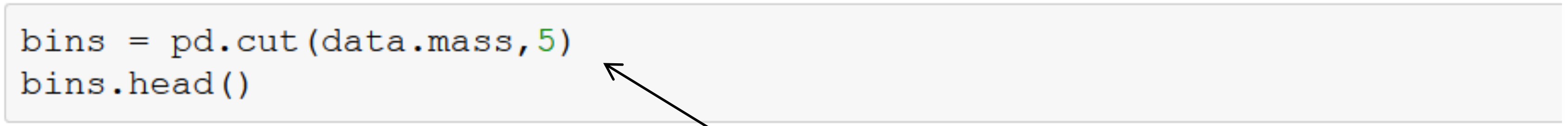
```
%matplotlib inline

data.mass.sort_values().hist(bins=10)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fe8ae59b70>
```

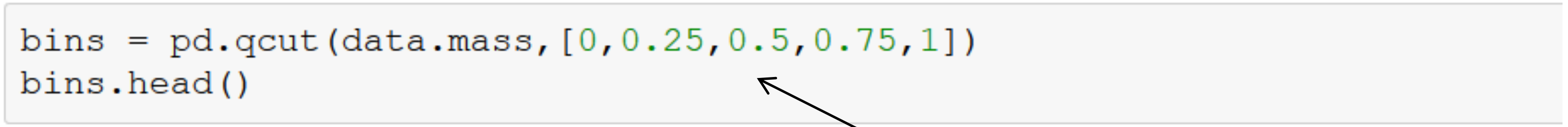


# Python Example



```
bins = pd.cut(data.mass,5)
bins.head()
```

Discretize into 5 equal-width bins



```
bins = pd.qcut(data.mass,[0,0.25,0.5,0.75,1])
bins.head()
```

Discretize into 4 equal-frequency bins (values are the quantiles)

## Unsupervised Discretization

Suppose we're interested to discretize the Age attribute

| Age | Buy |
|---|---|
| 10 | No |
| 15 | No |
| 18 | Yes |
| 19 | Yes |
| 24 | No |
| 29 | Yes |
| 30 | Yes |
| 31 | Yes |
| 40 | No |
| 44 | No |
| 55 | No |
| 64 | No |

Equal width: interval = (64-10)/3 = 54/3 = 18

| | Yes | No |
|---|---|---|
| < 28 | 2 | 3 |
| (29,46) | 3 | 2 |
| > 46 | 0 | 2 |

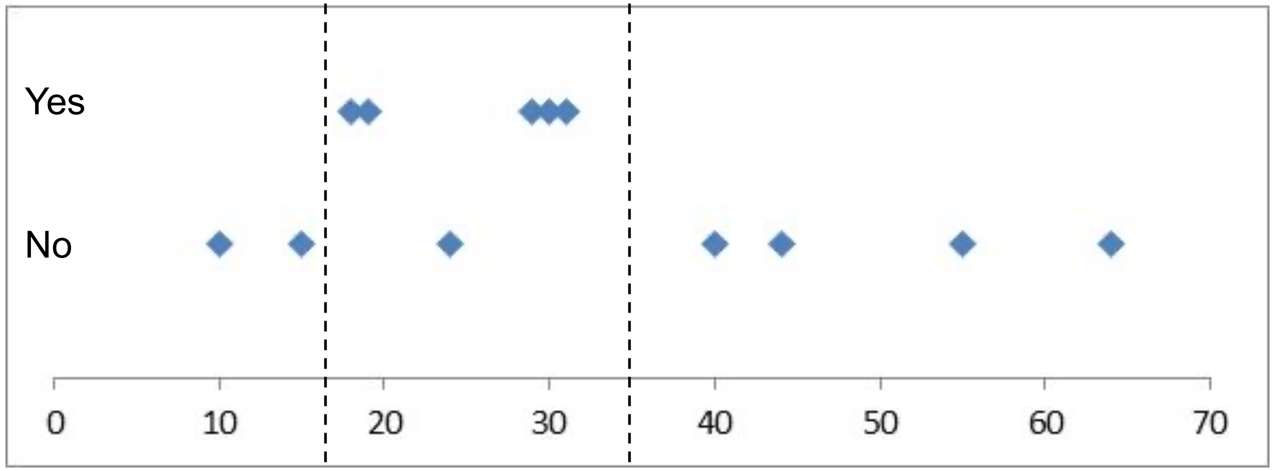
Equal frequency:

| | Yes | No |
|---|---|---|
| < 21.5 | 2 | 2 |
| (21.5,35.5) | 3 | 1 |
| > 35.5 | 0 | 4 |

Both approaches can produce bins that are not homogeneous

## Supervised Discretization

| Age | Buy |
|---|---|
| 10 | No |
| 15 | No |
| 18 | Yes |
| 19 | Yes |
| 24 | No |
| 29 | Yes |
| 30 | Yes |
| 31 | Yes |
| 40 | No |
| 44 | No |
| 55 | No |
| 64 | No |



Supervised discretization:

| | Yes | No |
|---|---|---|
| < 16.5 | 0 | 2 |
| (16.5,35.5) | 5 | 1 |
| > 35.5 | 0 | 4 |

In supervised discretization, our goal is to ensure that each bin contains data points from one class.

# Entropy-based Discretization

- A widely-used supervised discretization method

- Entropy is a measure of impurity
  - Higher entropy implies data points are from a large number of classes (heterogeneous)
  - Lower entropy implies most of the data points are from the same class (homogeneous)

$$Entropy_i = -\sum_{j=1}^{c} p_{ij} \log_2 p_{ij}$$

$$Entropy = \sum_{i=1}^{n} w_i Entropy_i$$

Where $p_{ij}$ is the proportion of data points belonging to class j in interval i and c is the number of classes

Where $w_i$ is the proportion of values in interval i and n is the number of intervals

## Entropy

$$\boxed{Entropy = -\sum_j p_j \log p_j}$$

Where $p_j$ is the fraction of data objects belonging to class j

| | |
|---|---|
| Yes | **0** |
| No | **6** |

P(Yes) = 0/6 = 0     P(No) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

| | |
|---|---|
| Yes | **1** |
| No | **5** |

P(Yes) = 1/6     P(No) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (5/6) = 0.65

| | |
|---|---|
| Yes | **2** |
| No | **4** |

P(Yes) = 2/6     P(No) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

As the bin becomes less homogeneous, entropy increases

# Entropy-based Discretization

Recursively find the best partition that minimizes entropy

| Age | Buy |
|---|---|
| 10 | No |
| 15 | No |
| 18 | Yes |
| 19 | Yes |
| 24 | No |
| 29 | Yes |
| 30 | Yes |
| 31 | Yes |
| 40 | No |
| 44 | No |
| 55 | No |
| 64 | No |

Split point = 35.5

| | Yes | No |
|---|---|---|
| < 35.5 | 5 | 3 |
| >= 35.5 | 0 | 4 |

$$E_1 = -\frac{5}{8}\log_2\frac{5}{8} - \frac{3}{8}\log_2\frac{3}{8} = 0.9544$$

$$E_2 = -\frac{0}{4}\log_2\frac{0}{4} - \frac{4}{4}\log_2\frac{4}{4} = 0$$

$$E_T = \frac{8}{12}E_1 + \frac{4}{12}E_2 = 0.6363$$

# Entropy-based Discretization

| Age | Buy |
|---|---|
| 10 | No |
| 15 | No |
| 18 | Yes |
| 19 | Yes |
| 24 | No |
| 29 | Yes |
| 30 | Yes |
| 31 | Yes |
| 40 | No |
| 44 | No |
| 55 | No |
| 64 | No |

Split point = 16.5

Split point = 35.5

Find the next best partition that minimizes entropy

| | Yes | No |
|---|---|---|
| < 16.5 | 0 | 2 |
| (16.5,35.5) | 5 | 1 |
| > 35.5 | 0 | 4 |

$$E_1 = -\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2} = 0$$

$$E_2 = -\frac{5}{6}\log_2\frac{5}{6} - \frac{1}{6}\log_2\frac{1}{6} = 0.65$$

$$E_3 = -\frac{0}{4}\log_2\frac{0}{4} - \frac{4}{4}\log_2\frac{4}{4} = 0$$

$$E_T = \frac{2}{12}E_1 + \frac{6}{12}E_2 + \frac{4}{12}E_3 = 0.325$$

## Dimension Reduction

- Dimensionality = number of attributes in the data

- Many datasets are inherently high-dimensional
  - Retail data, text data, survey data, gene expression data, etc..

- Why high dimensionality is bad?
  - Some attributes are irrelevant to the prediction task
  - Some attributes are correlated with each other
    - Can lead to model overfitting
  - Can lead to curse of dimensionality problem

# Effect of Irrelevant Attributes

| Name | Age | State | Avg Time Spent | … | Buy |
|---|---|---|---|---|---|
| John | 28 | MI | 40 | | No |
| Mary | 21 | MI | 105 | | No |
| Bob | 39 | FL | 100 | | Yes |
| … | … | … | … | … | … |
| Jim | 45 | TX | 74 | | No |

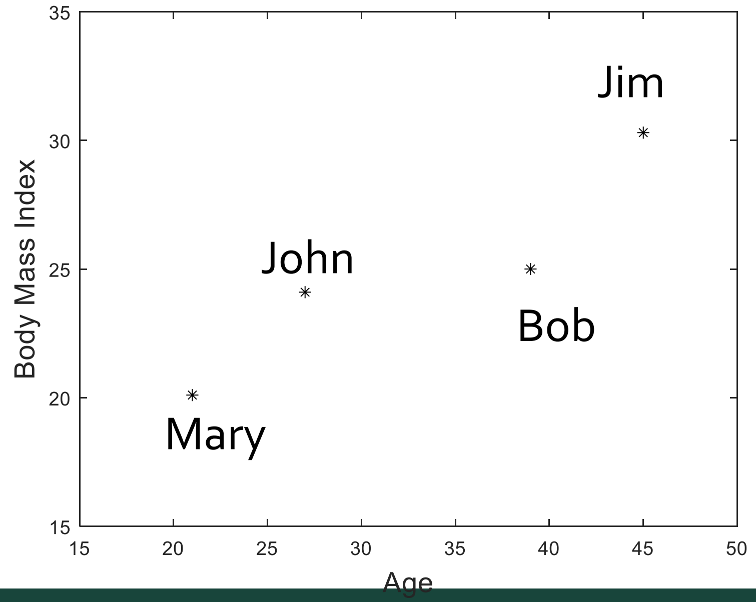
- Suppose we're interested to predict whether a visitor to an online site will buy anything
  - Name is an irrelevant attribute (it does not determine whether a visitor will buy or not)
  - If you have too many irrelevant attributes, your predictive modeling algorithm may inadvertently select them to build its model

# Effect of Correlated Attributes

| Name | Age | Body Mass Index |
|---|---|---|
| John | 23 | 24.1 |
| Mary | 21 | 20.1 |
| Bob | 39 | 25.0 |
| Jim | 45 | 32.2 |

Euclidean distance(X, Y)

$$= \sqrt{(Age_X - Age_Y)^2 + (BMI_X - BMI_Y)^2}$$



| | John | Mary | Bob | Jim |
|---|---|---|---|---|
| John | 0 | 7.211103 | 12.0337 | 19.03786 |
| Mary | 7.211103 | 0 | 18.65503 | 26.07758 |
| Bob | 12.0337 | 18.65503 | 0 | 8.005623 |
| Jim | 19.03786 | 26.07758 | 8.005623 | 0 |

John is more similar to Mary;

Bob is more similar to Jim

# Effect of Correlated Attributes

| Name | Age | Body Mass Index | Weight | Height | Daily Calorie Intake |
|---|---|---|---|---|---|
| John | 23 | 24.1 | 190 | 6.2 | 2700 |
| Mary | 21 | 20.1 | 110 | 5.2 | 2200 |
| Bob | 39 | 25.0 | 155 | 5.5 | 2400 |
| Jim | 45 | 32.2 | 215 | 5.9 | 3000 |

- BMI is correlated with weight, height, and daily calorie

| | John | Mary | Bob | Jim |
|---|---|---|---|---|
| John | 0 | 506.3803 | 302.4604 | 301.9515 |
| Mary | 506.3803 | 0 | 205.8473 | 807.309 |
| Bob | 302.4604 | 205.8473 | 0 | 603.0655 |
| Jim | 301.9515 | 807.309 | 603.0655 | 0 |

John is more similar to Jim;
Mary is more similar to Bob

Presence of correlated attributes can alter patterns observed in the data

## The Curse of Dimensionality

- Data analysis becomes harder as the dimensionality of data increases
- As dimensionality increases, data becomes increasingly sparse
- Classification: not enough data to create a reliable model
- Clustering: distances between points less meaningful

## Examples with Curse of Dimensionality

- Data sets with large number of features
  - Document matrix

| | w1 | w2 | w3 | w4 | … | … | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Doc1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | … |
| Doc2 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | … |
| … | | | | | | | | | | | | |

## Dimensionality Reduction

- Identifies strong patterns in the data
- Expresses data in a way that highlights differences and similarities
- Reduces the number of dimensions without much loss of information
- Techniques:
  - Feature Reduction
  - Principal Component Analysis
  - Singular Value Decomposition
  - Others: supervised and non-linear techniques

## Why Feature Reduction?

- Less data so the data mining algorithm learns faster
- Higher accuracy so the model can generalize better
- Simple results so they are easier to understand
- Fewer features for improving the data collection process
- Highly correlated features may affect accuracy/stability of algorithm
  - Naïve Bayes, Linear regression

## Techniques

- Feature selection:
  - Select a subset of the attributes in the original data set
  - Eliminate the irrelevant and highly correlated ones

- Feature composition/creation:
  - Creating new features based on (linear or nonlinear) combination of original ones

## Examples for Feature Selection

- Redundant/Correlated features:
  - contain duplicate information: *price*, *tax amount*
- Irrelevant features:
  - No useful information: Phone number, employee id
  - Zero/Close to zero variance feature: not distinctive enough
  - Text mining: stop words: *a*, *an*, *of*, *the*, *your*, ….

## Feature Selection

- Exhaustive enumeration is not practically feasible
- Approaches:
  - **Filter**: before the data mining algorithm is run, evaluate features independent of data mining task
  - **Embedded**: attributes are selected in the process of the data mining algorithm execution
  - **Wrapper approaches**: every subset that is proposed by the subset selection measure is evaluated in the context of the learning algorithm

# Feature Selection

- General Approach:
  - Generate a subset of features
  - Evaluate
    - Using an evaluation function for filter approaches
    - Using the data mining algorithm for wrapper approaches
  - If good, stop
  - Otherwise, update subset and repeat
- Need:
  - Subset generation strategy
  - Measure for evaluation and a stopping criterion

## Filter approach: Pairwise correlation

- Heuristic: Remove predictors to ensure correlation is below a certain threshold
  1. Calculate the correlation matrix of the predictors.
  2. Determine the two predictors associated with the largest absolute pairwise correlation (call them predictors *A* and *B*).
  3. Determine the average correlation between *A* and the other variables.
  4. Do the same for predictor *B*.
  5. If *A* has a larger average correlation, remove it; otherwise, remove predictor *B*.
  6. Repeat Steps 2–4 until no absolute correlations are above the threshold

## Feature Creation

- Feature Extraction
- Dimensionality Reduction
  - Principal Component Analysis (PCA)
  - Singular Value Decomposition (SVD)
  - Factor Analysis
- Feature transformation:
  - Categorical to binary
  - Binning of continuous data

# Feature Extraction

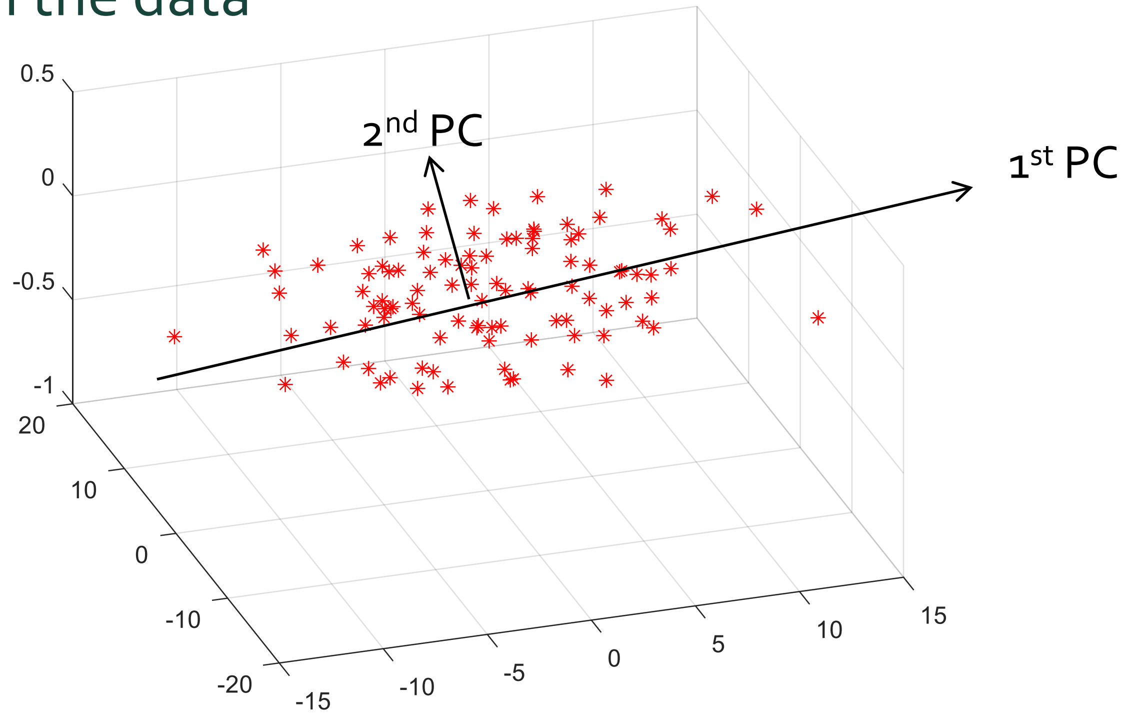
- Domain specific

- Image classification: goal is face recognition
  - Raw data: pixel values
  - Extracted features: edges, areas highly correlated with human faces

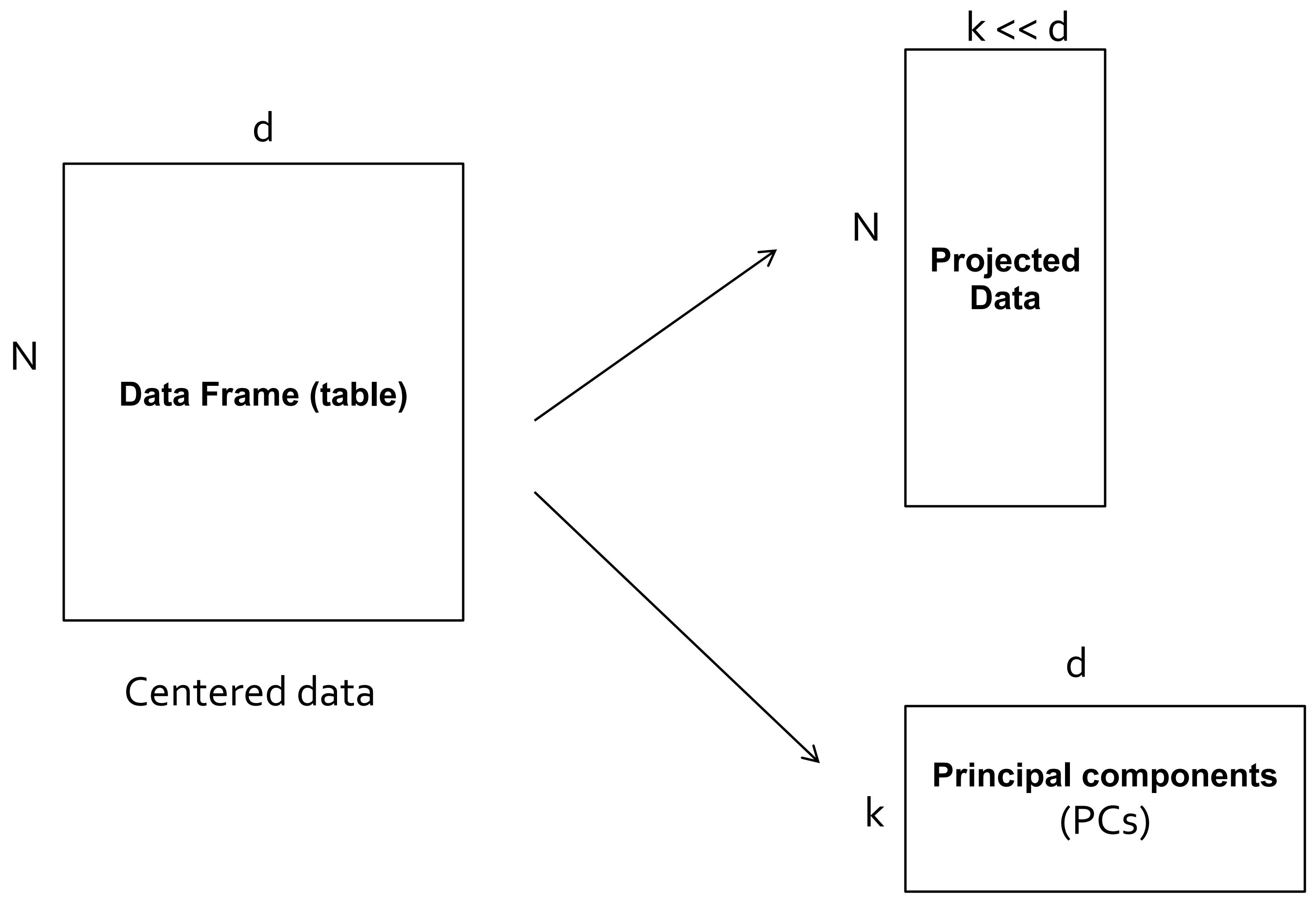
- Historical artifacts: goal is material detection
  - Raw data: volume and mass
  - Extracted feature: density = mass/volume

# Principal Component Analysis (PCA)

- Create new set of attributes from original data
- Goal of PCA is to construct a new set of attributes that capture most of the variability in the data



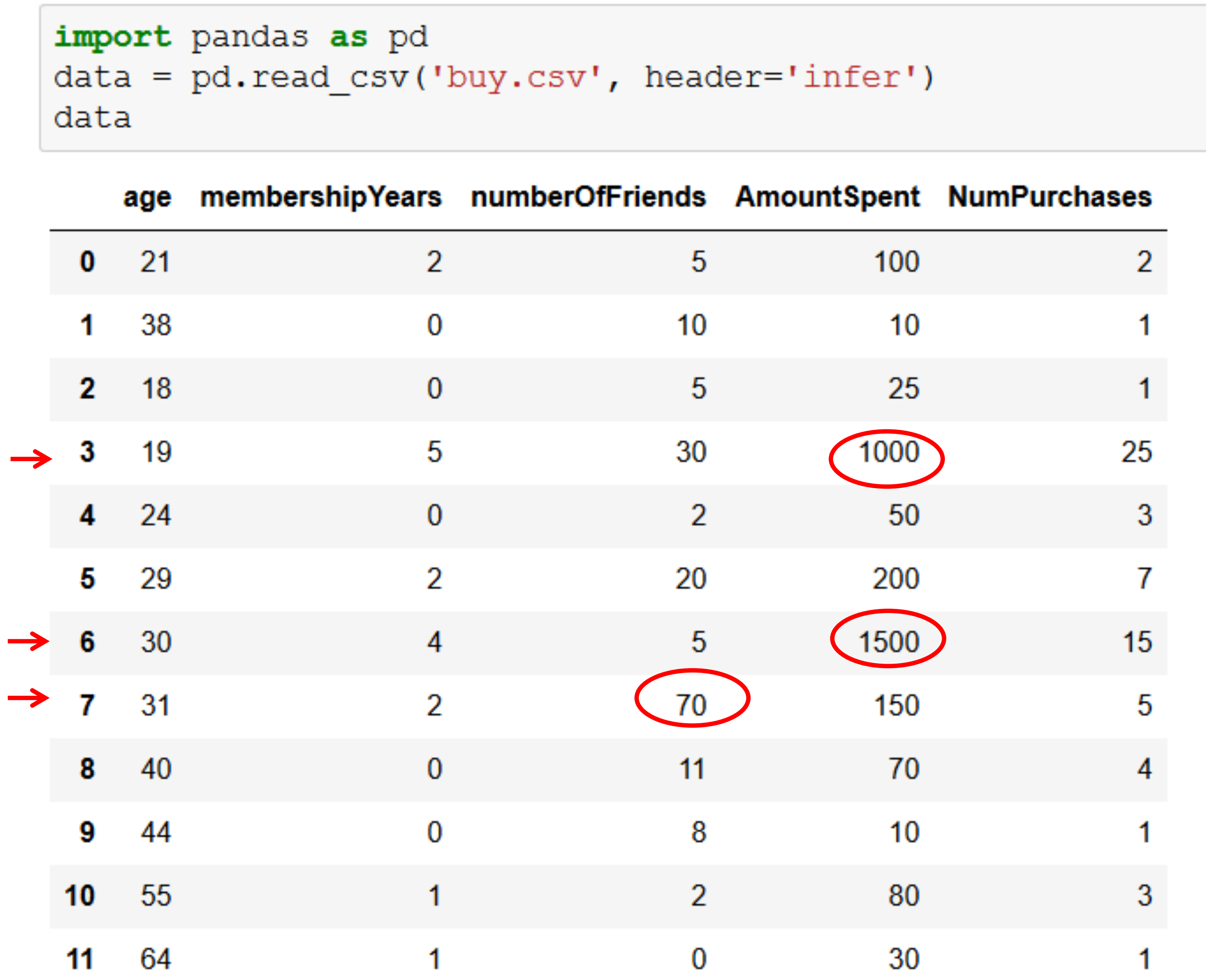
# Principal Component Analysis



## Principal Component Analysis

- First dimension is chosen to capture the variability of the data
- Second dimension is chosen orthogonal to the first one so it captures as much of the remaining variability, and so on …
- Each pair of new attributes has covariance 0
- Attributes are ordered in decreasing order of the variance in data that they capture

# Example

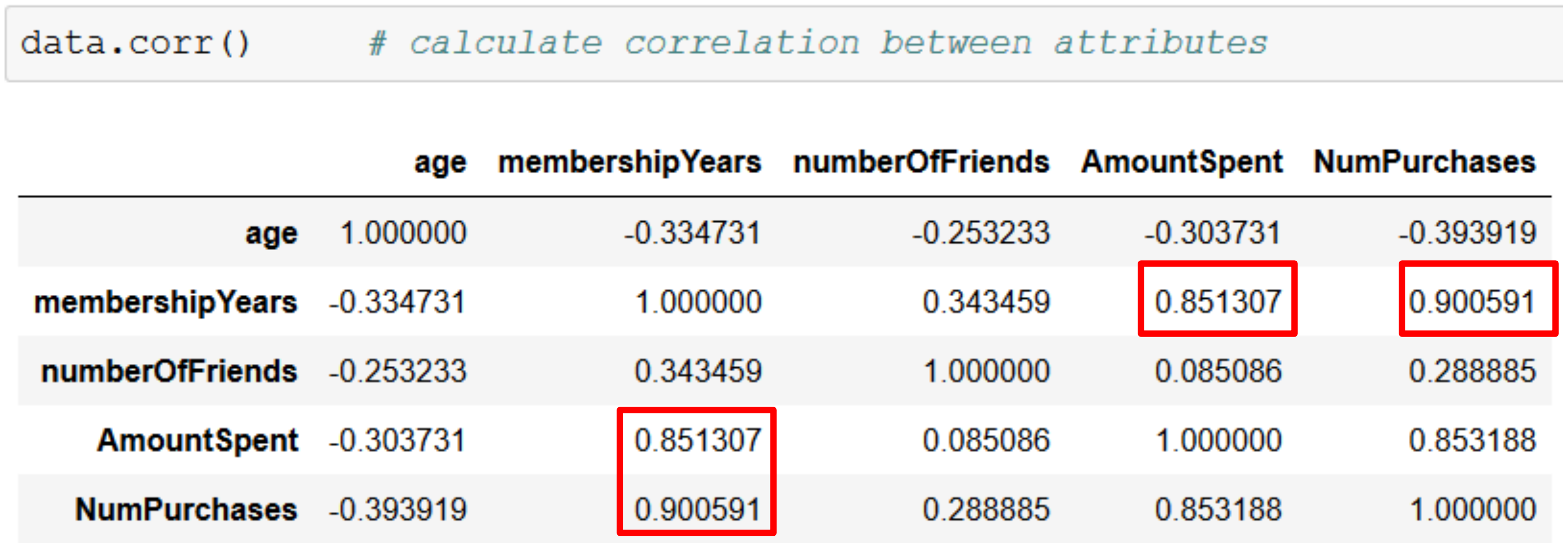


```python
import pandas as pd
data = pd.read_csv('buy.csv', header='infer')
data
```

| | age | membershipYears | numberOfFriends | AmountSpent | NumPurchases |
|---|---|---|---|---|---|
| **0** | 21 | 2 | 5 | 100 | 2 |
| **1** | 38 | 0 | 10 | 10 | 1 |
| **2** | 18 | 0 | 5 | 25 | 1 |
| **3** | 19 | 5 | 30 | 1000 | 25 |
| **4** | 24 | 0 | 2 | 50 | 3 |
| **5** | 29 | 2 | 20 | 200 | 7 |
| **6** | 30 | 4 | 5 | 1500 | 15 |
| **7** | 31 | 2 | 70 | 150 | 5 |
| **8** | 40 | 0 | 11 | 70 | 4 |
| **9** | 44 | 0 | 8 | 10 | 1 |
| **10** | 55 | 1 | 2 | 80 | 3 |
| **11** | 64 | 1 | 0 | 30 | 1 |

# Example

- Note that membership years, amount spent, and number of purchases are quite correlated



```
data.corr()        # calculate correlation between attributes
```

| | age | membershipYears | numberOfFriends | AmountSpent | NumPurchases |
|---|---|---|---|---|---|
| **age** | 1.000000 | -0.334731 | -0.253233 | -0.303731 | -0.393919 |
| **membershipYears** | -0.334731 | 1.000000 | 0.343459 | 0.851307 | 0.900591 |
| **numberOfFriends** | -0.253233 | 0.343459 | 1.000000 | 0.085086 | 0.288885 |
| **AmountSpent** | -0.303731 | 0.851307 | 0.085086 | 1.000000 | 0.853188 |
| **NumPurchases** | -0.393919 | 0.900591 | 0.288885 | 0.853188 | 1.000000 |

- Suppose we want to project the data from d = 5 features to k = 2 features (principal components)

# Computing Principal Components

- Given a data set D
  - Calculate the covariance matrix C
  - PCs are the eigenvectors of the covariance matrix
  - To calculate the projected data:
    - Center each column of the data by subtracting its mean: D'
    - Calculate the projections: (T: transpose operation)

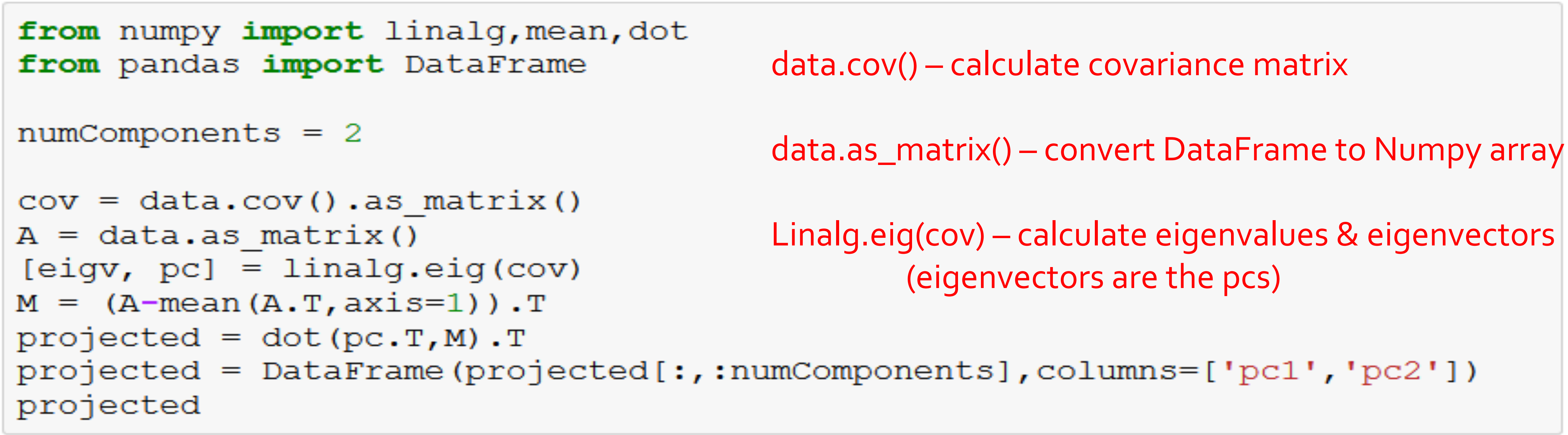
$$\underbrace{\text{projected}^T}_{k \times N} = \underbrace{(PC)^T}_{k \times d} \times \underbrace{(D')^T}_{d \times N}$$

Non-transposed form:

$$\underset{N \times k}{\text{projected}} = \underset{N \times d}{D'} \times \underset{d \times k}{PC}$$

## Example

We can use numpy linear algebra functions to calculate eigenvectors and perform matrix multiplication



```
from numpy import linalg,mean,dot
from pandas import DataFrame

numComponents = 2

cov = data.cov().as_matrix()
A = data.as_matrix()
[eigv, pc] = linalg.eig(cov)
M = (A-mean(A.T,axis=1)).T
projected = dot(pc.T,M).T
projected = DataFrame(projected[:,:numComponents],columns=['pc1','pc2'])
projected
```

data.cov() – calculate covariance matrix

data.as_matrix() – convert DataFrame to Numpy array

Linalg.eig(cov) – calculate eigenvalues & eigenvectors (eigenvectors are the pcs)

(A – mean(A.T, axis=1)) - center the columns of data matrix

dot(pc.T, M).T - multiply PC with centered data matrix

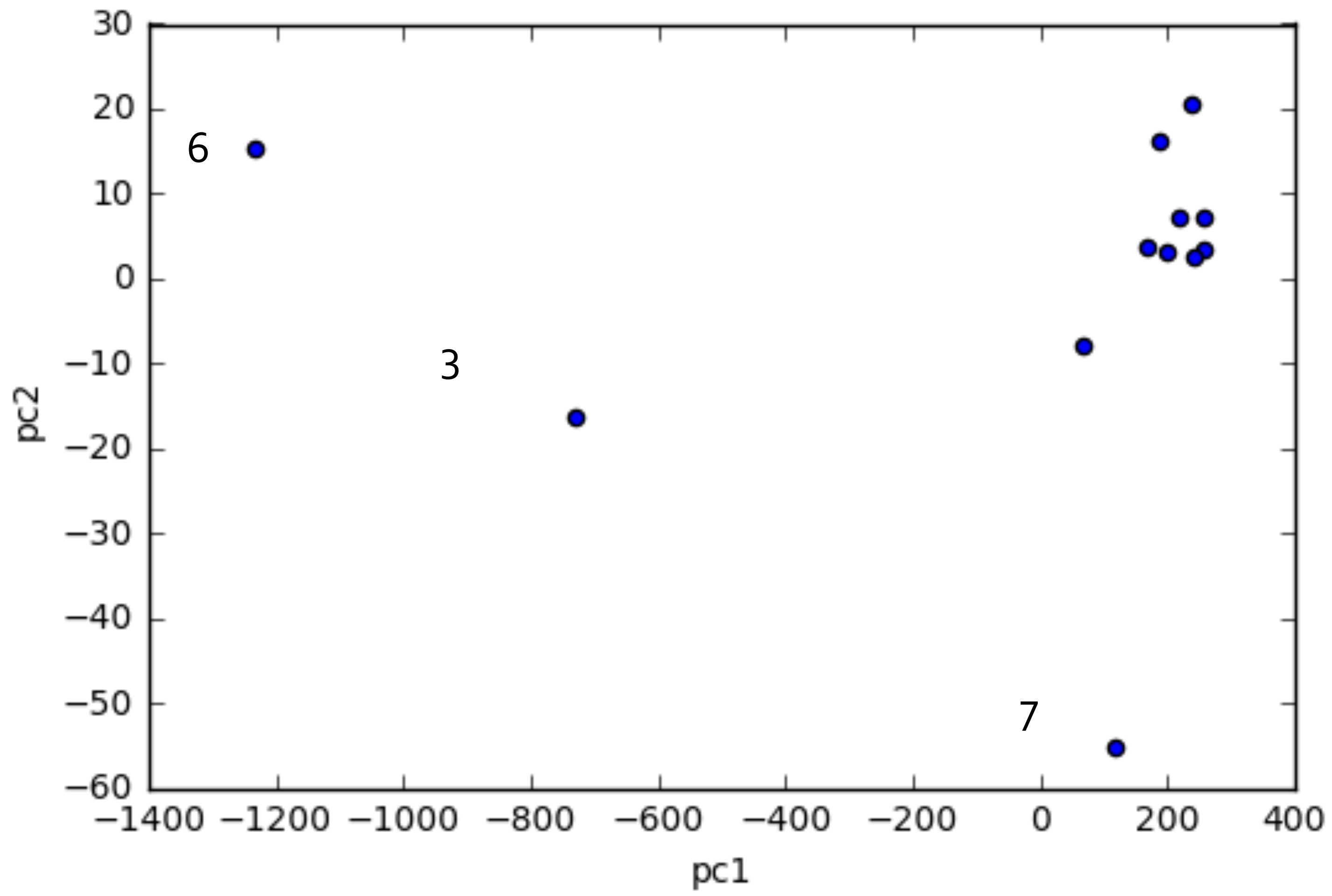
## Example

| | pc1 | pc2 |
|---|---|---|
| **0** | 168.680257 | 3.738699 |
| **1** | 258.826567 | 3.412064 |
| **2** | 243.661156 | 2.452026 |
| **3** | -731.610997 | -16.175841 |
| **4** | 218.704586 | 7.067655 |
| **5** | 68.649959 | -7.919821 |
| **6** | -1231.217724 | 15.348058 |
| **7** | 118.511454 | -55.146559 |
| **8** | 198.810667 | 3.219743 |
| **9** | 258.889152 | 7.066049 |
| **10** | 188.992684 | 16.299912 |
| **11** | 239.102238 | 20.638014 |

```
%matplotlib inline

projected.plot(kind='scatter',x='pc1',y='pc2')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xd0d4940>
```



## Example

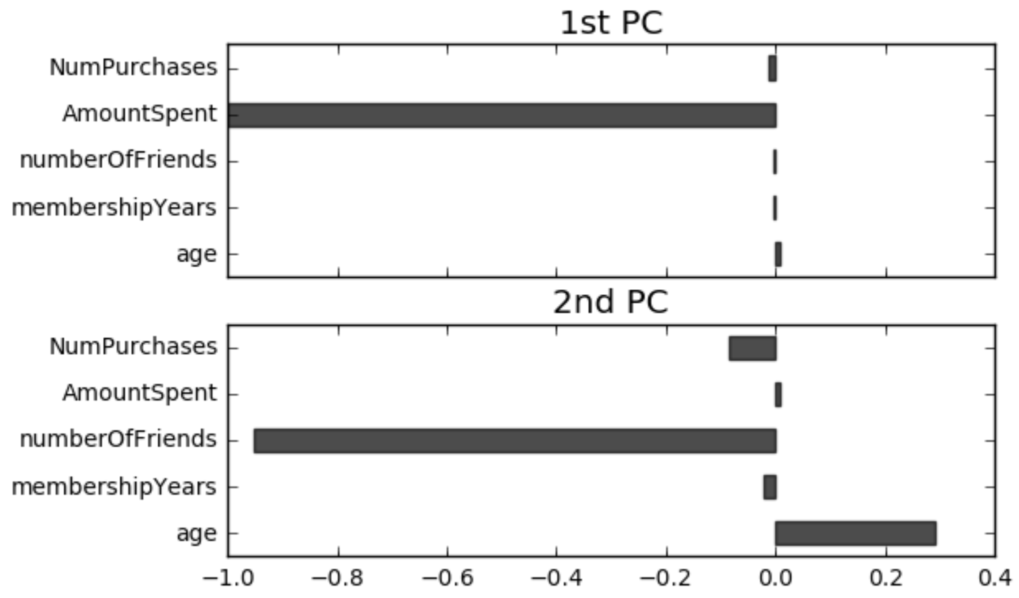
```
pc[:,:numComponents]
```

```
array([[ 0.00925599,  0.29153545],
       [-0.00301099, -0.02282163],
       [-0.00352437, -0.95238638],
       [-0.99986089,  0.0072493 ],
       [-0.01307821, -0.08598769]])
```

↑ 1st PC ↑ 2nd PC

```
%matplotlib inline
import matplotlib.pyplot as plt
from pandas import Series

fig,axes = plt.subplots(2,1,sharex=True)
attrib = list(data)           # get attribute names
pcdata  = Series(pc[:,0], index=attrib)
pcdata.plot(kind='barh',ax=axes[0],color='k',alpha=0.7)
axes[0].set_title('1st PC', size = 'x-large')
pcdata  = Series(pc[:,1], index=attrib)
pcdata.plot(kind='barh',ax=axes[1],color='k',alpha=0.7)
axes[1].set_title('2nd PC', size = 'x-large')
```



# PCA Using Scikit-learn

- An easier way to apply principal component analysis is by using Python's scikit-learn module
- First, you need to install scikit-learn on python

```
conda install scikit-learn
```

1. Create a 2-d numpy array to store the data
2. Create a PCA object and specifies the number of components needed
3. Use fit() function to extract principal components
   - PCs are stored in the components_ field
4. Use transform() function to project the data to its new components

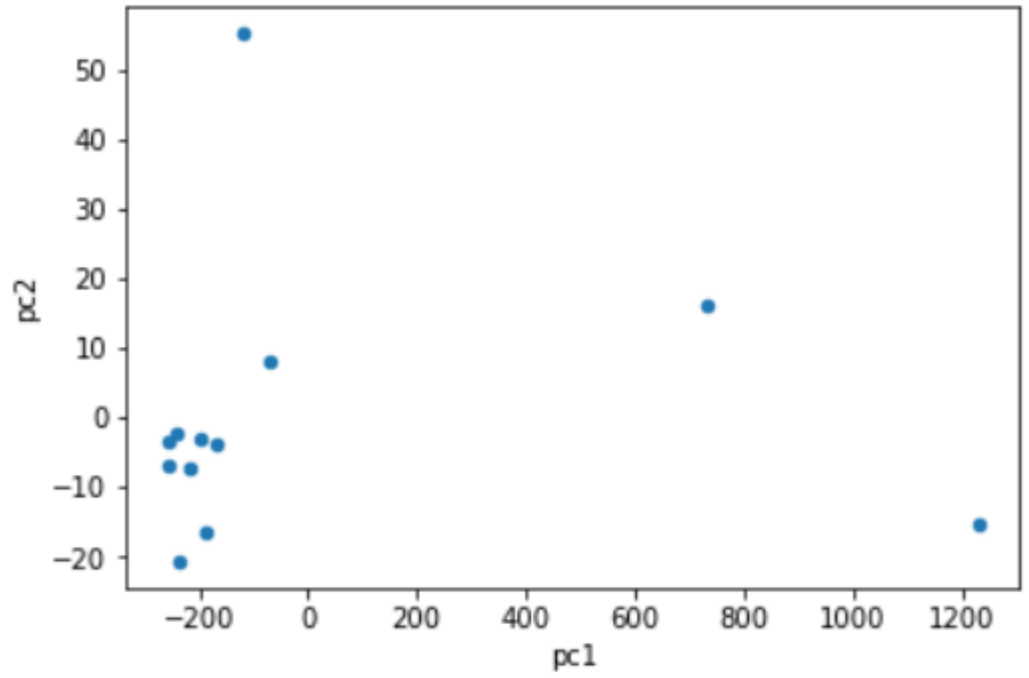
## PCA Using Scikit-Learn

```
from sklearn.decomposition import PCA

pca = PCA(n_components=numComponents)
pca.fit(A)
projected = pca.transform(A)
projected = pd.DataFrame(projected,columns=['pc1','pc2'])
projected.plot(kind='scatter',x='pc1',y='pc2')
```

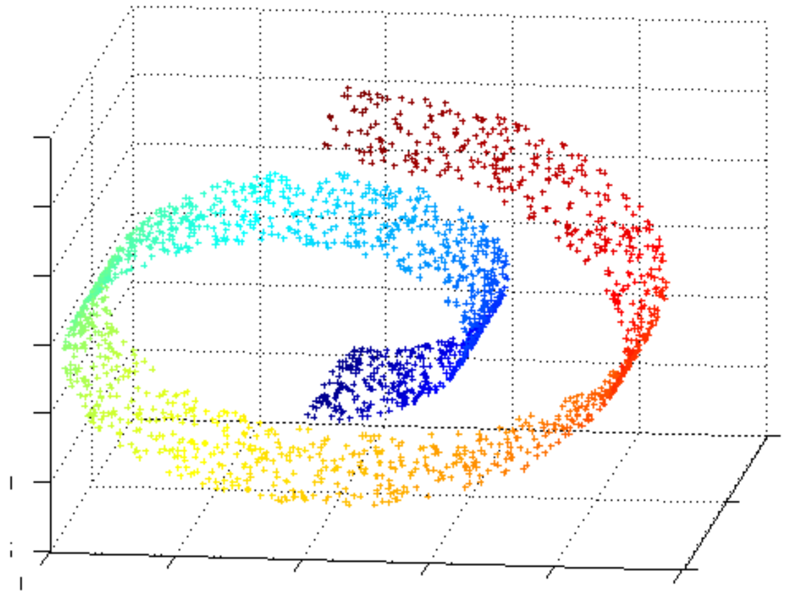
A is the data matrix

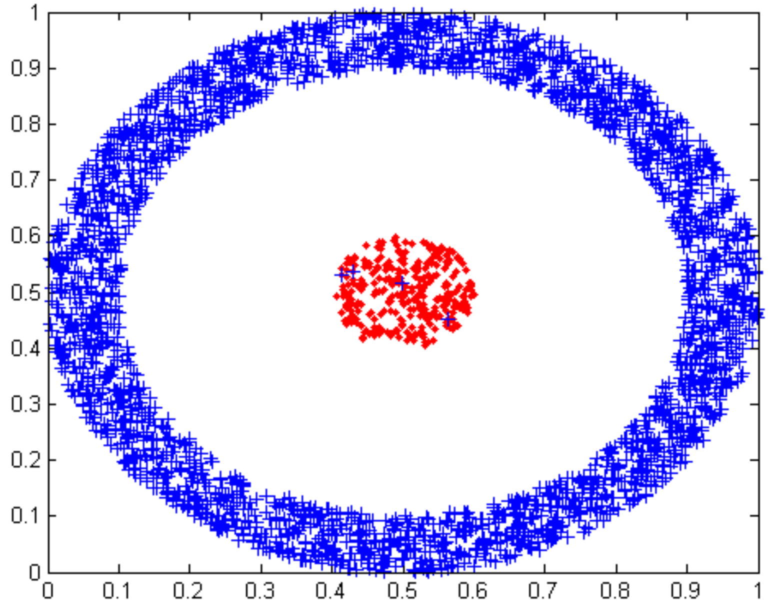
```
<matplotlib.axes._subplots.AxesSubplot at 0x17a46839518>
```



## Extending to Nonlinear PCA

- PCA constructs only linear components; it doesn't work well if the data is embedded in a nonlinear manifold





# Kernel Trick

- Project the data to a high dimensional space so that you can apply linear PCA
  - Original 2-d data: $\boldsymbol{x} = (x_1, x_2)$
  - Project the data to a high dimensional space and find the components in this space

$$\Phi(\boldsymbol{x}) = (x_1, x_2, x_1x_2, x_1^2, x_2^2, x_1^3, x_2^3, x_1^2\mathrm{x}_2, \ldots)$$

- For example:
  - Original 2-d data: $\boldsymbol{x} = (x, y)$
  - Non linear relationship: $4x^2 + 9xy + 7y^2 = 10$
  - New data: x1 = $x^2$ , x2 = $xy$ , x3 = $y^2$
  - New linear relationship: $4x1 + 9x2 + 7x3 = 10$

## Examples of Kernel Matrices

- Radial Basis Function (RBF) kernel

$$K\left(\Phi\left(x_i\right),\Phi\left(x_j\right)\right)=\exp\left(-k\left\|x_i-x_j\right\|^2\right)$$

- Polynomial kernel

$$\boldsymbol{K}\left(\Phi\left(\boldsymbol{x}_i\right),\Phi\left(\boldsymbol{x}_j\right)\right)=\left(\boldsymbol{x}_i\cdot\boldsymbol{x}_j\right)^{\boldsymbol{d}}$$

Trick is that similarity between two high-dimensional vectors can be computed in the original feature space!

## PCA Issues

- Selects predictors based on variability => scale matters
- **Unsupervised**: Does not take into account the outcome (dependent variable)
- If features variability is not connected to outcome, PCA will not provide suitable relationship
- Difficult to interpret the results

## Variable Transformation

- Attributes could have different scales

- When to transform:
  - Different units
  - Different scale

- When not to transform:
  - Units of measurements are common, and
  - Scale reflects importance

- Examples:
  - Tuition
  - Graduation rate

## Feature scaling – Range [0,1]

**Standardization**

- Subtract mean $\mu$
- Divide by standard deviation $\sigma$

$$\hat{x} = \frac{x - \mu}{\sigma}$$

- Original data follows normal distribution

**Min-Max Normalization**

- Subtract minimum
- Divide by range

$$\hat{x} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- Original data does not follow normal distribution

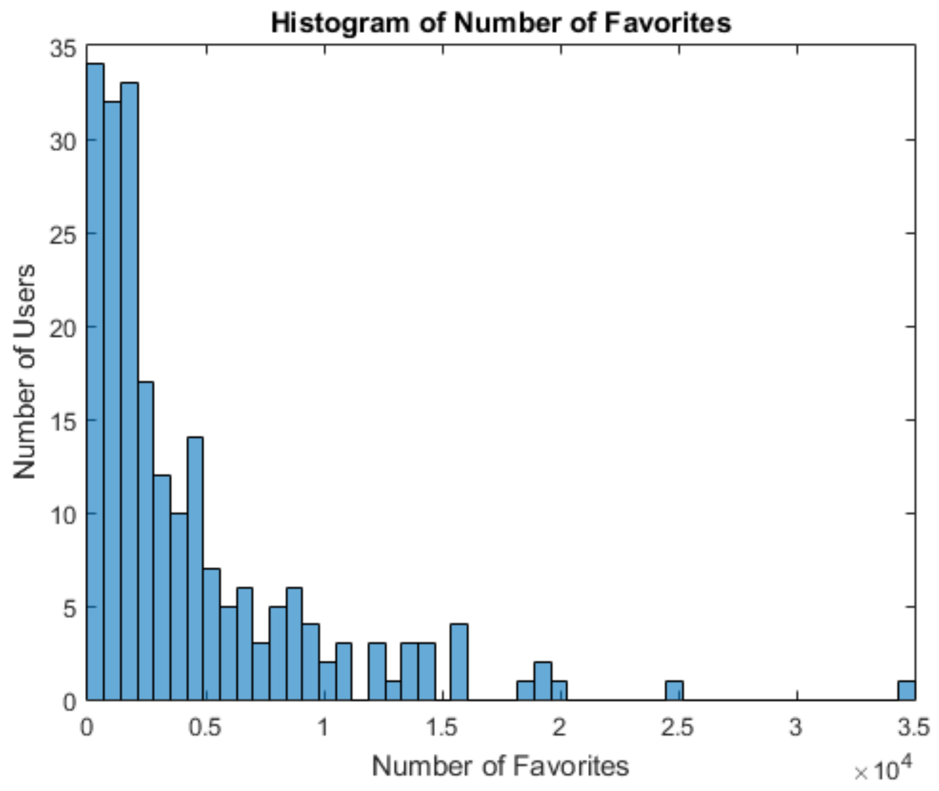
# Nonlinear Variable Transformations

- Determined by plotting a histogram of the data for that attribute

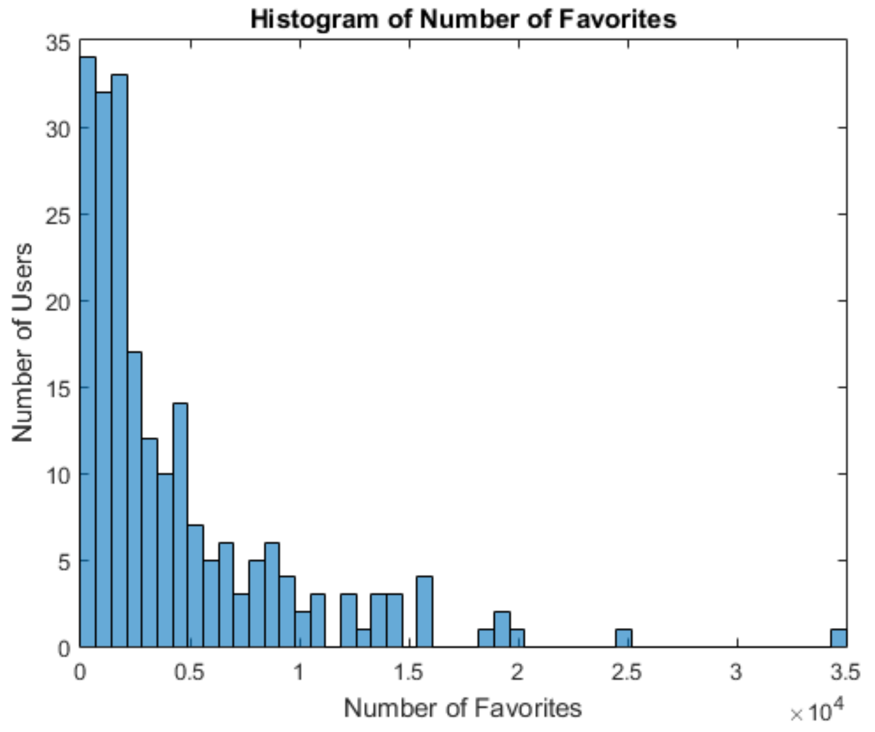
- Square root $\sqrt{x}$

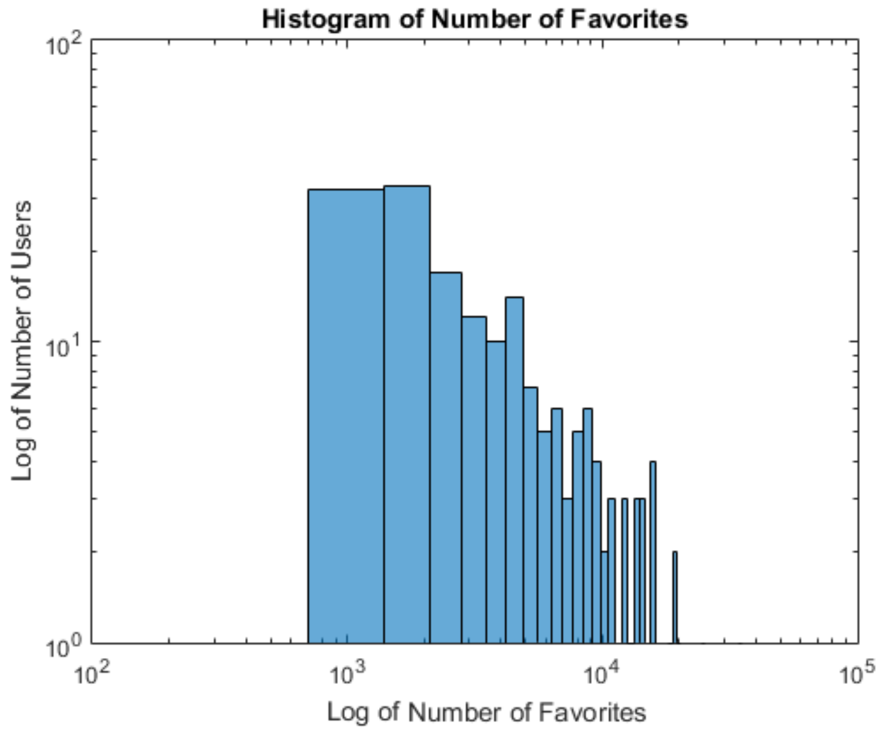
- Logarithmic $\log(x)$

- Reciprocal ${}^{1}/_{x}$



## Number of tweets favorited by each user





## No Free Lunch

**“what an algorithm gains in performance on one class of problems is necessarily offset by its performance on the remaining problem”**

Wolpert and Macready, 1997

- No approach works well on all problems
- Significant amount of trial and error

## Summary

- Data quality affects quality of knowledge discovered
- Preprocessing needed:
  - Improve quality
  - Transform into suitable form
- Approaches:
  - Aggregation
  - Sampling
  - Discretization
  - Dimensionality reduction
  - Variable transformation

# Data Mining Process

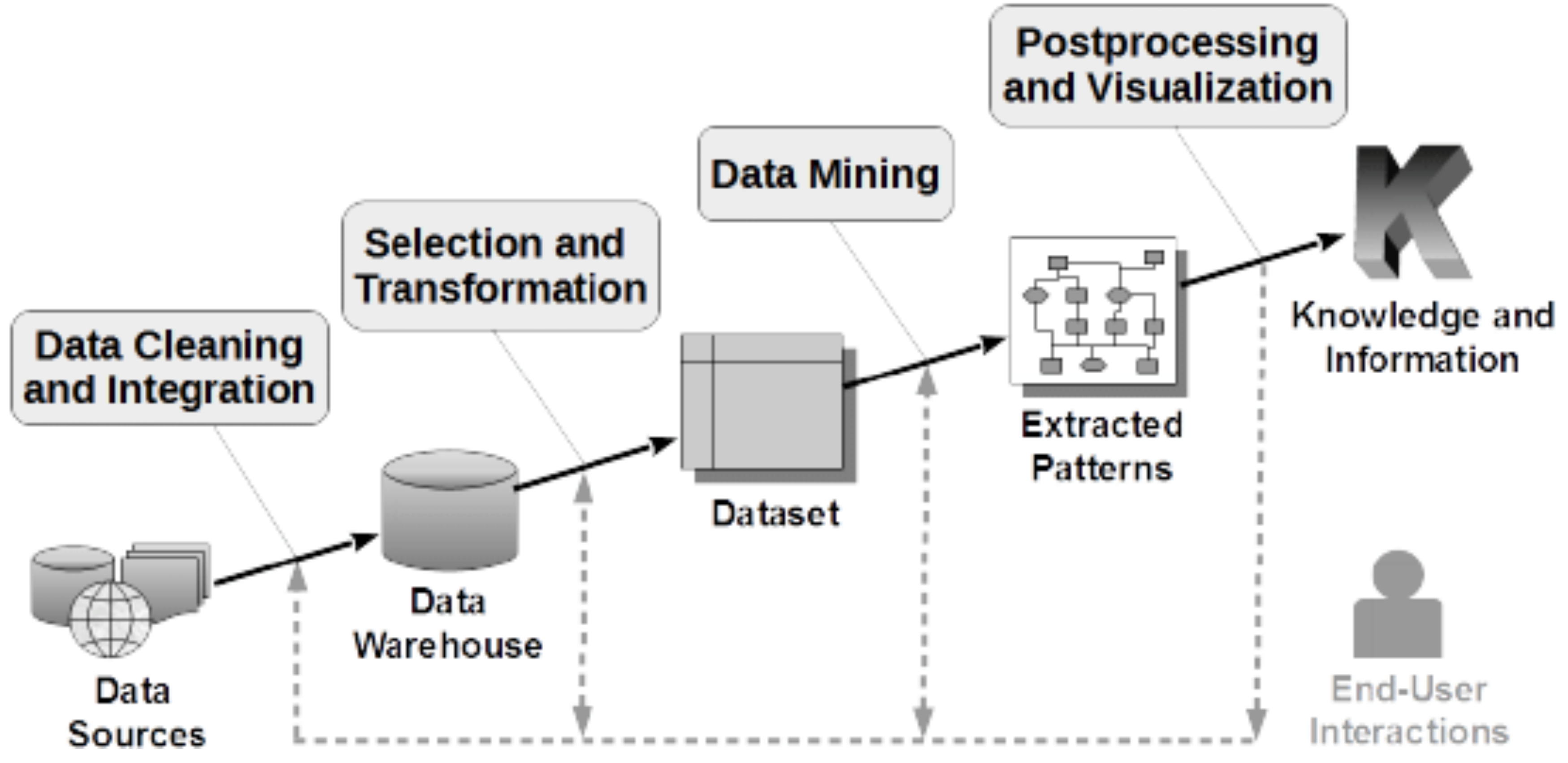

Postprocessing and Visualization
Data Mining
Selection and Transformation
Data Cleaning and Integration
Knowledge and Information
Extracted Patterns
Dataset
Data Warehouse
Data Sources
End-User Interactions

Image from: Inthasone, Somsack & Pasquier, Nicolas & Tettamanzi, Andrea & da Costa Pereira, Celia. (2014). The BioKET Biodiversity Data Warehouse: Data and Knowledge Integration and Extraction. 10.1007/978-3-319-12571-8_12.