Unsupervised Learning

Peerapon Vateekul, Ph.D.
peerapon.v@4amconsult.com

Outliers

- Clustering
  - K-means
  - DBSCAN
- Market Basket Analysis
- Special tasks

Unsupervised learning (descriptive task)

1. Clustering
2. Association Rule Mining

Training Data

inputs

target

Age	Income	Gender	Province	Purchase
25	25,000	Female	Bangkok	Yes
35	50,000	Female	Nontaburi	Yes
32	35,000	Male	Bangkok	

Testing Data

Age	Income	Gender	Province	Purchase
25	25,000	Female	Bangkok	?

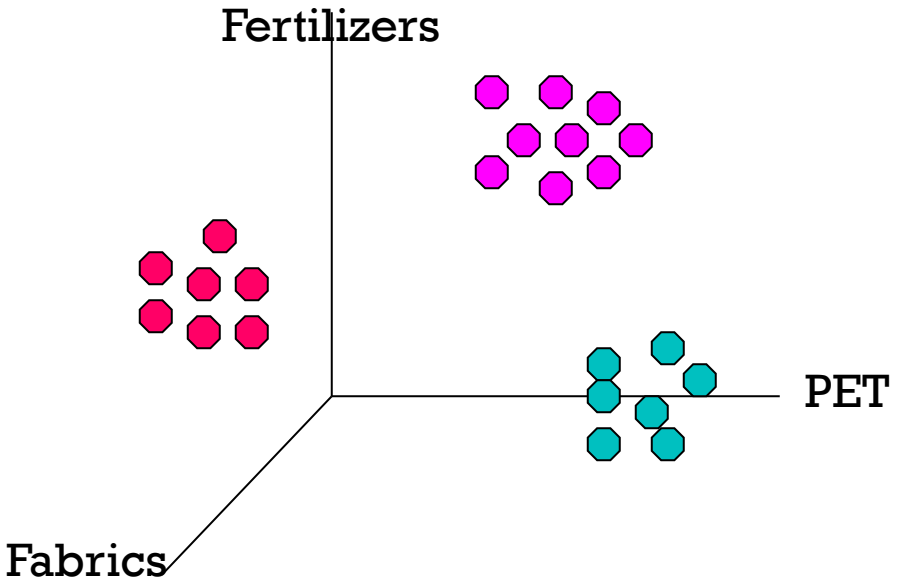
Clustering

- In our class, there are many participants. Should we teach them using the same method?
- May be not! Since they may have different learning behaviors and backgrounds.
- Inputs
  - Education field
  - Level of English communication
  - Level of computer skills
  - Age range
  - Gender

Clustering

Company	Sedan	Truck	Motorcycles
C1	70M	2M	80M
C2	90M	120M	100M
C3	1M	8M	70M

- **Some techniques:**
  1. K-means
  2. Hierarchical clustering
  3. DB-scan



Example: Customer Segmentation

Clustering
1) K-means

- Iterative clustering-based algorithms
  - Find centroid of clusters on each iterations
- Most common exploratory data analysis technique

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$

- Clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the *inertia* or within-cluster sum-of-squares
- This algorithm requires the number of clusters to be specified
- Sensitive to outliers

Reference : https://scikit-learn.org/stable/modules/clustering.html

Clustering
- K-means

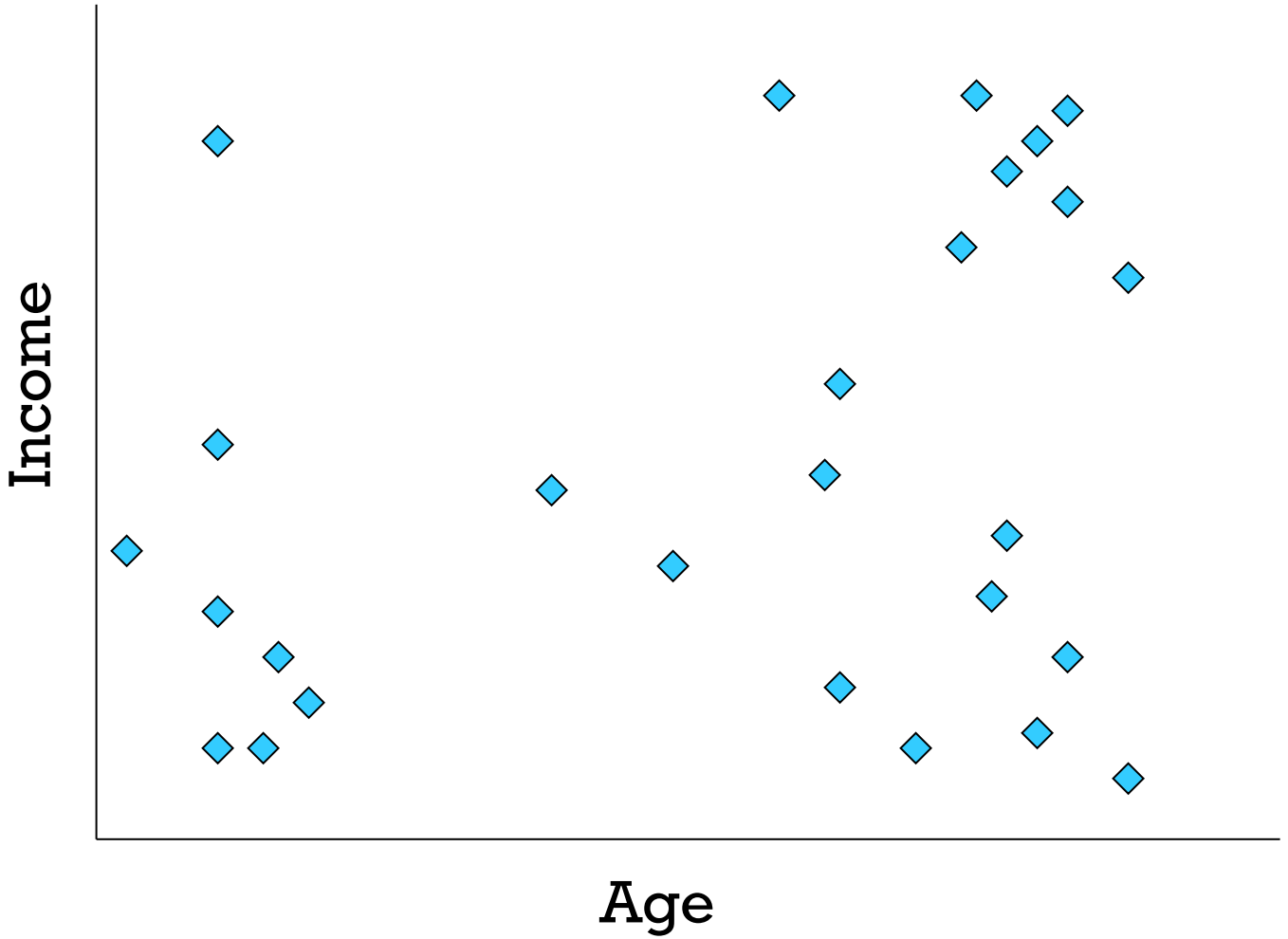
- **Step**

1. Initial random k cluster centers
2. Assign data to closest center
3. Update cluster center using average
4. Repeat step 2 and 3
5. Stop when
    - until convergence
    - reach max iteration

Clustering
- K-means

- **Step**

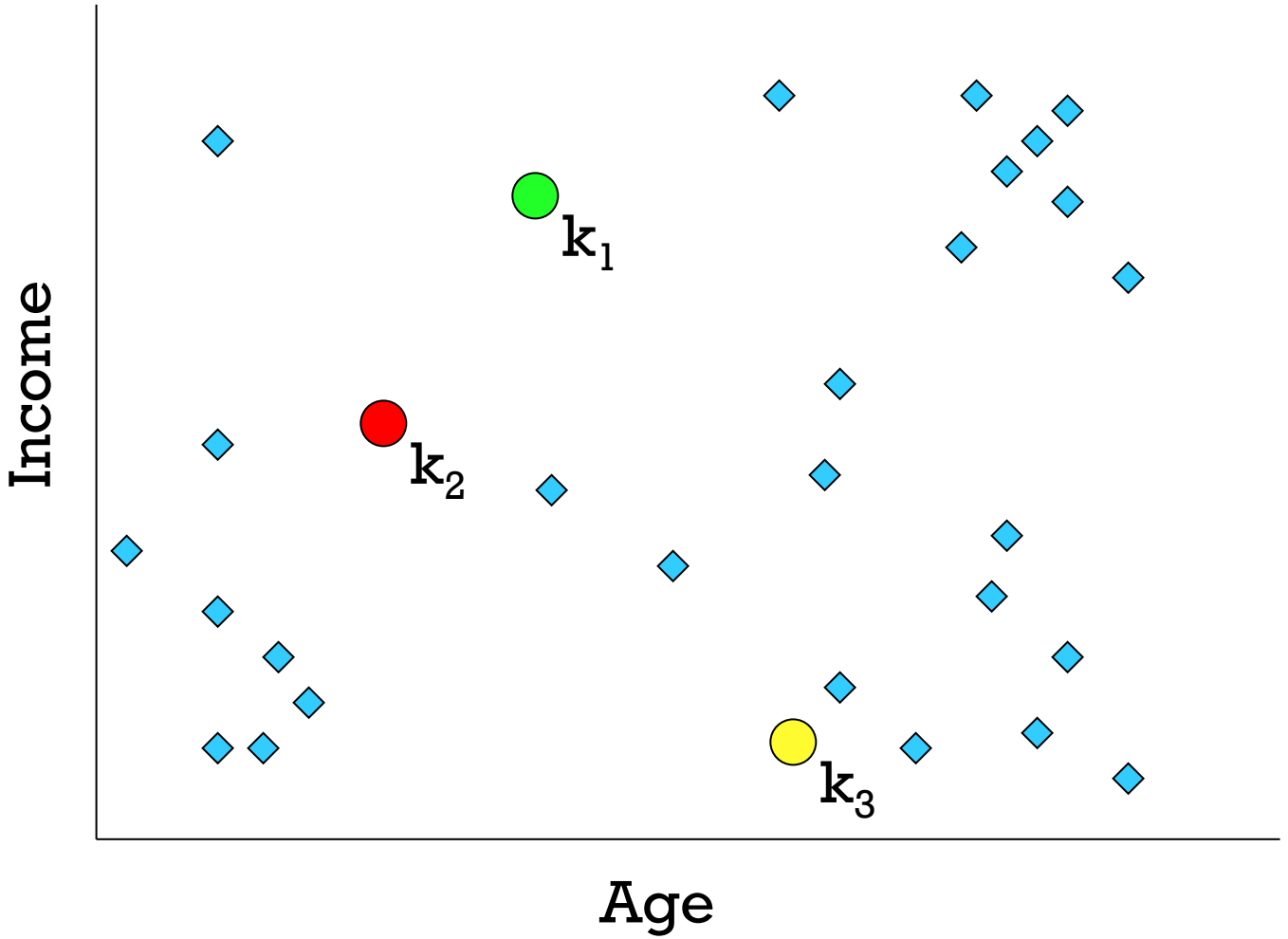
1. Initial random k cluster centers
2. Assign data to closest center
3. Update cluster center using average
4. Repeat step 2 and 3
5. Stop when
   - until convergence
   - reach max iteration



Clustering
- K-means

- **Step**

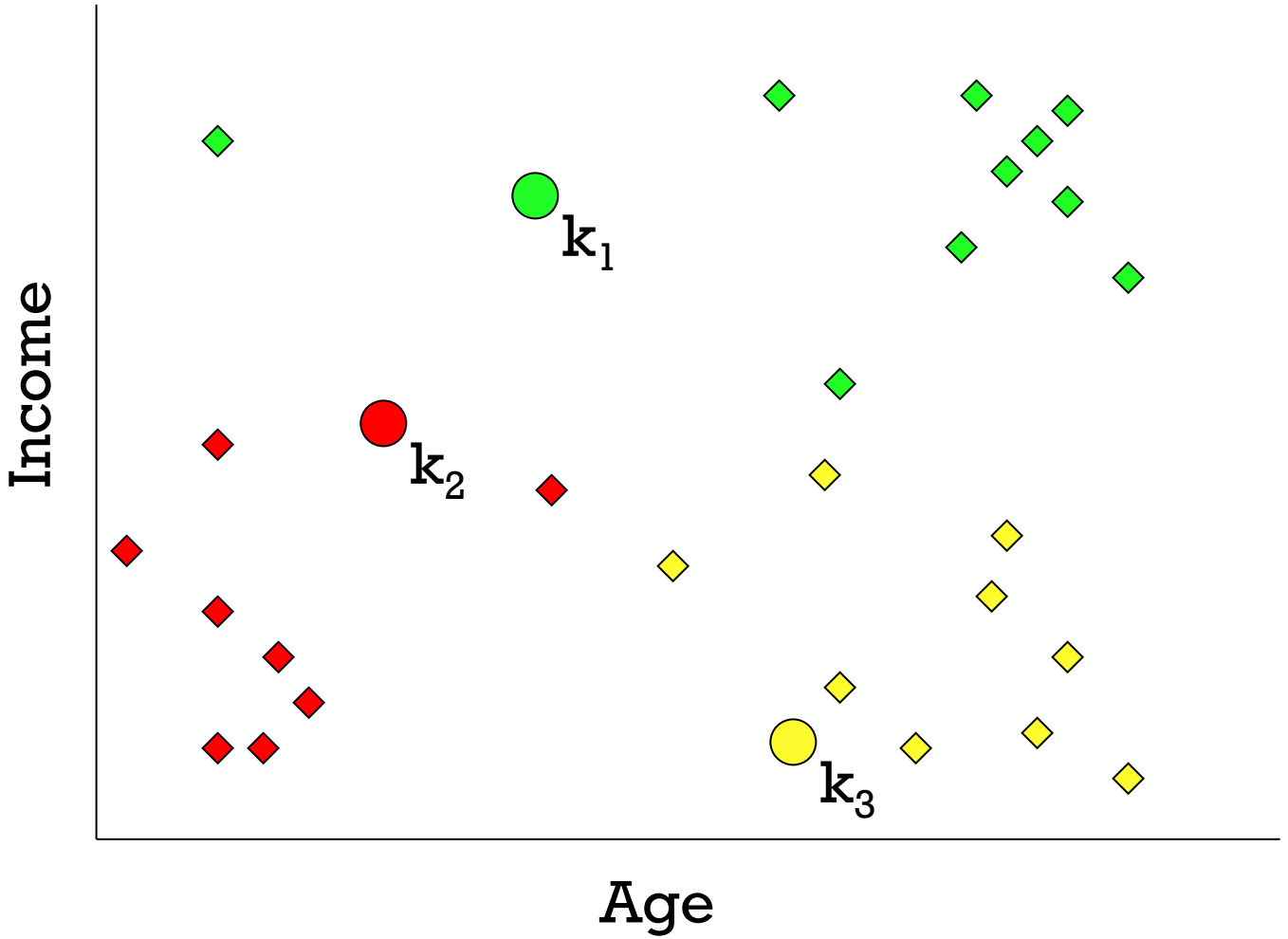
1. Initial random k cluster centers
2. Assign data to closest center
3. Update cluster center using average
4. Repeat step 2 and 3
5. Stop when
    - until convergence
    - reach max iteration



Clustering
- K-means

- **Step**

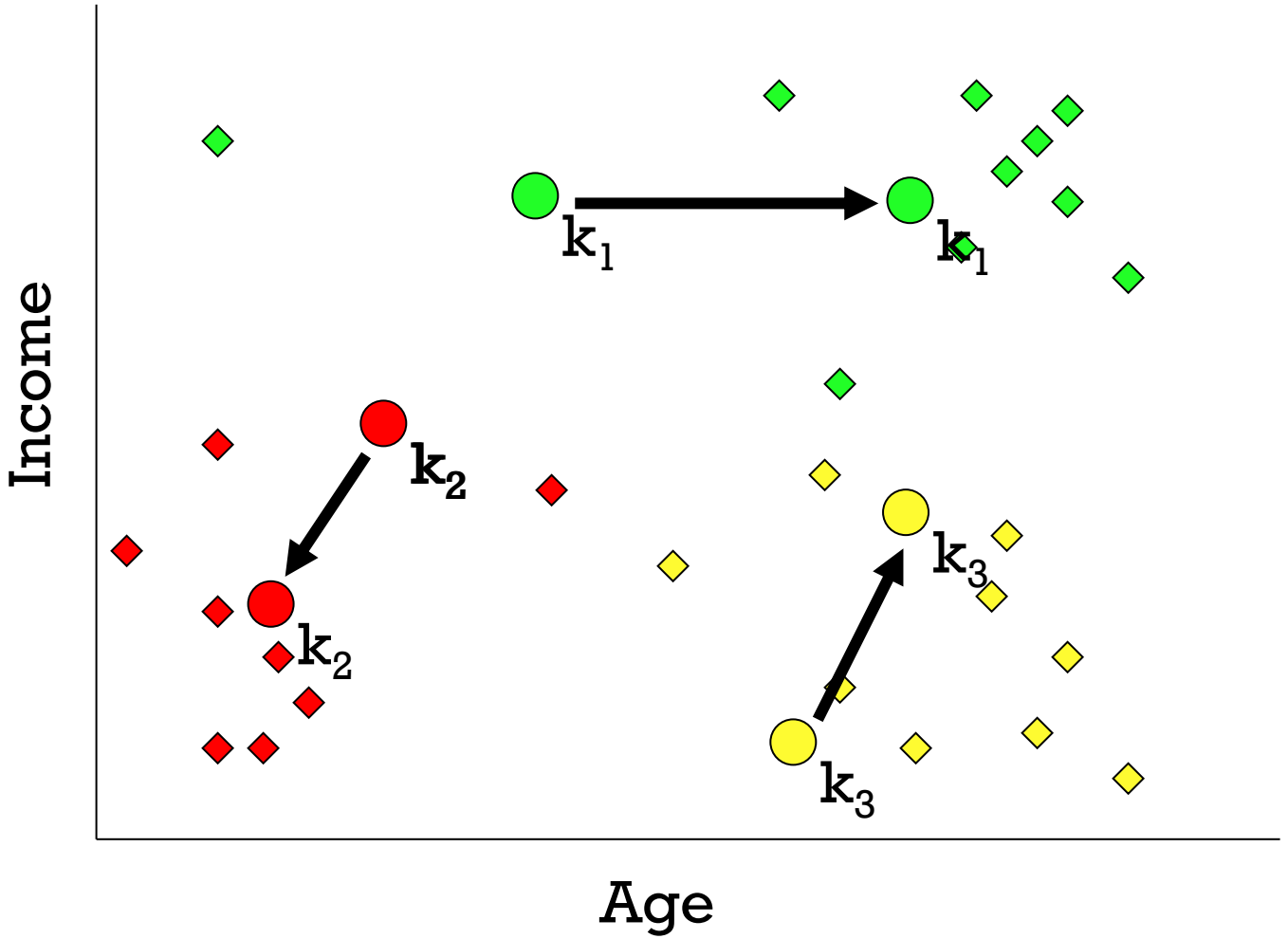
1. Initial random k cluster centers
2. Assign data to closest center
3. Update cluster center using average
4. Repeat step 2 and 3
5. Stop when
   - until convergence
   - reach max iteration



Clustering
- K-means : Step 1

- **Step**

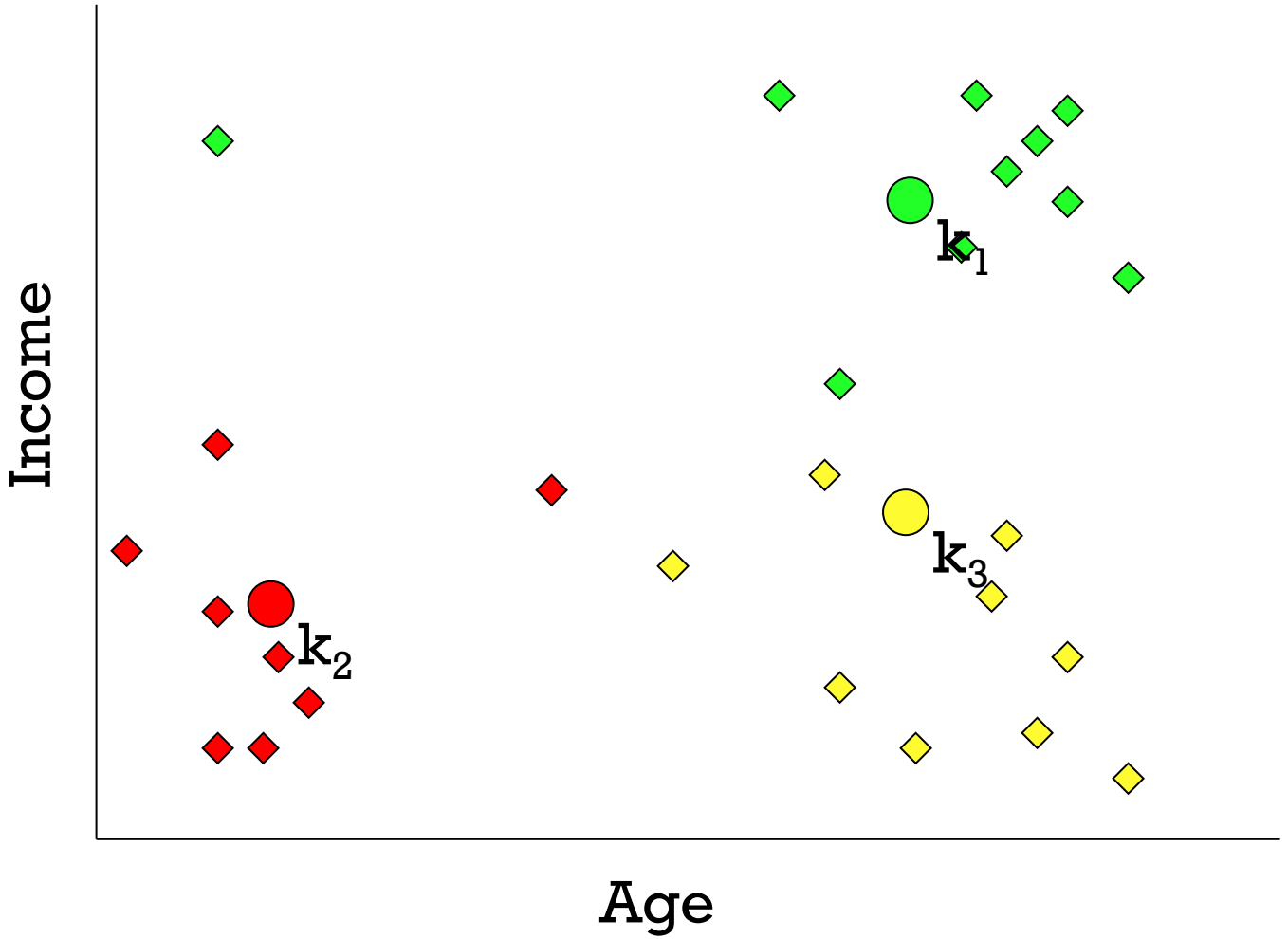
1. Initial random k cluster centers
2. Assign data to closest center
3. Update cluster center using average
4. Repeat step 2 and 3
5. Stop when
   - until convergence
   - reach max iteration



Clustering
- K-means

- **Step**

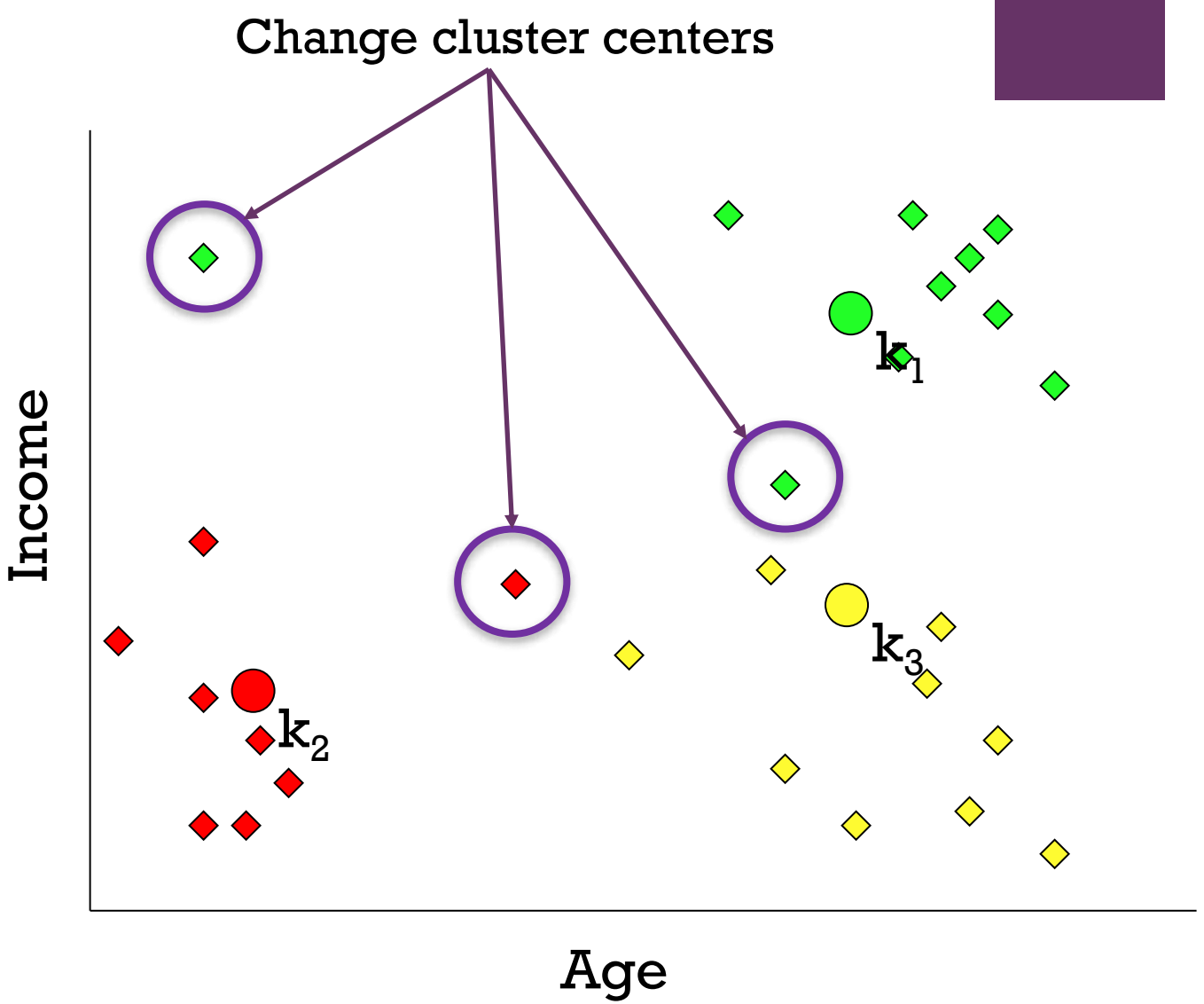
1. Initial random k cluster centers
2. Assign data to closest center
3. Update cluster center using average
4. Repeat step 2 and 3
5. Stop when
   - until convergence
   - reach max iteration



Clustering
- K-means

- **Step**

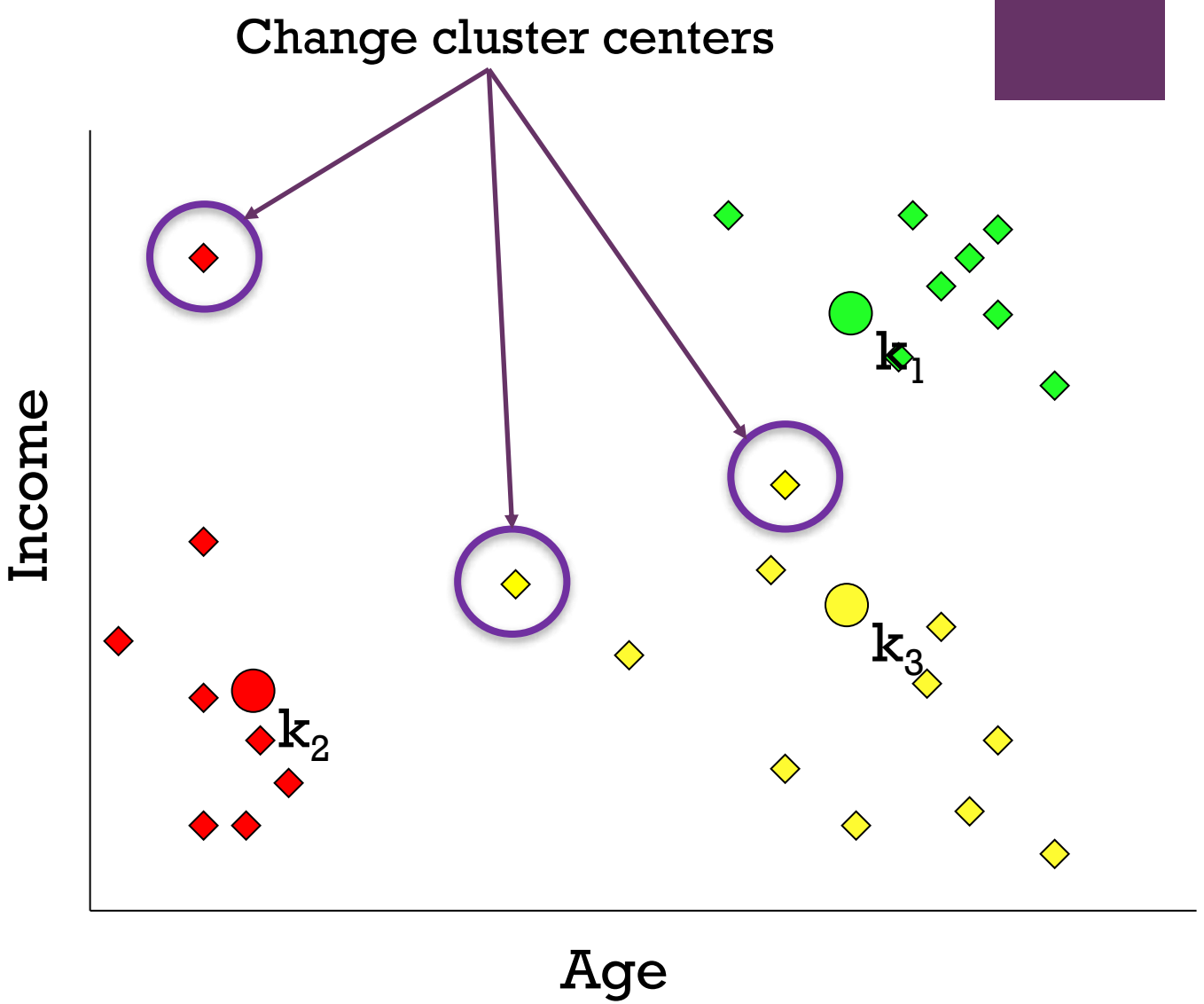
1. Initial random k cluster centers
2. Assign data to closest center
3. Update cluster center using average
4. Repeat step 2 and 3
5. Stop when
   - until convergence
   - reach max iteration



Clustering
- K-means

- **Step**

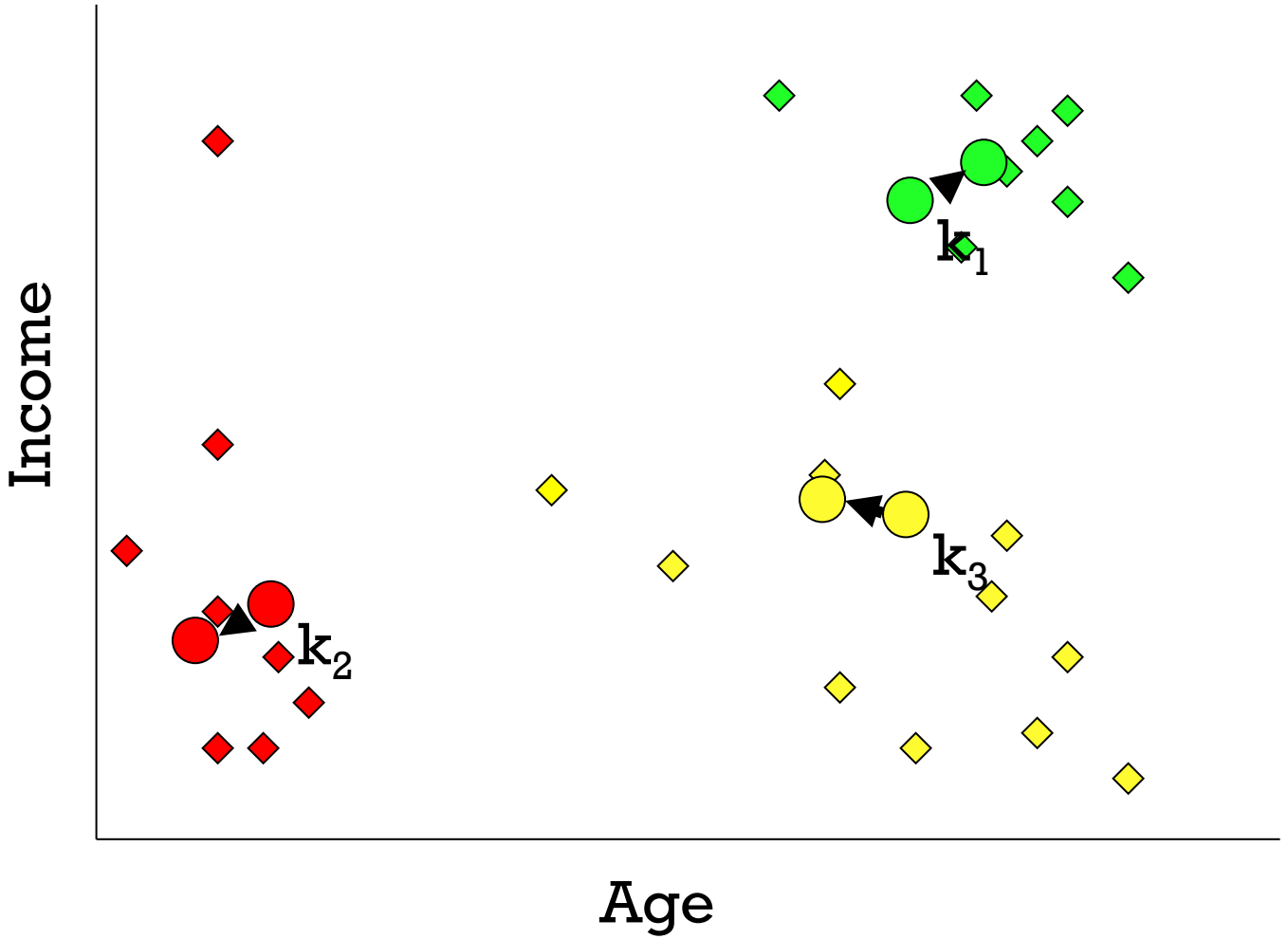
1. Initial random k cluster centers
2. Assign data to closest center
3. Update cluster center using average
4. Repeat step 2 and 3
5. Stop when
   - until convergence
   - reach max iteration



Clustering
- K-means

- **Step**

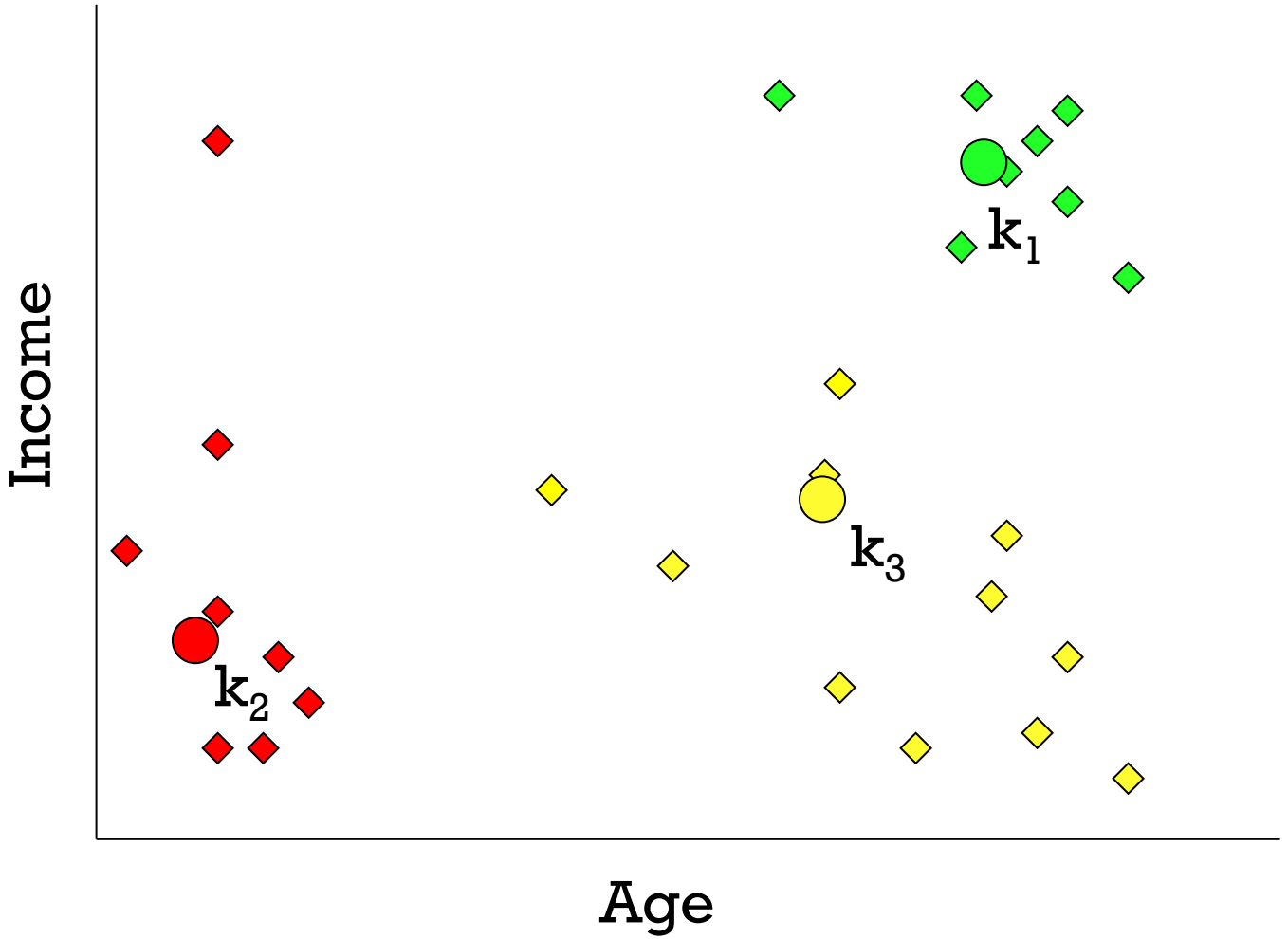
1. Initial random k cluster centers
2. Assign data to closest center
3. Update cluster center using average
4. Repeat step 2 and 3
5. Stop when
   - until convergence
   - reach max iteration



Clustering
- K-means

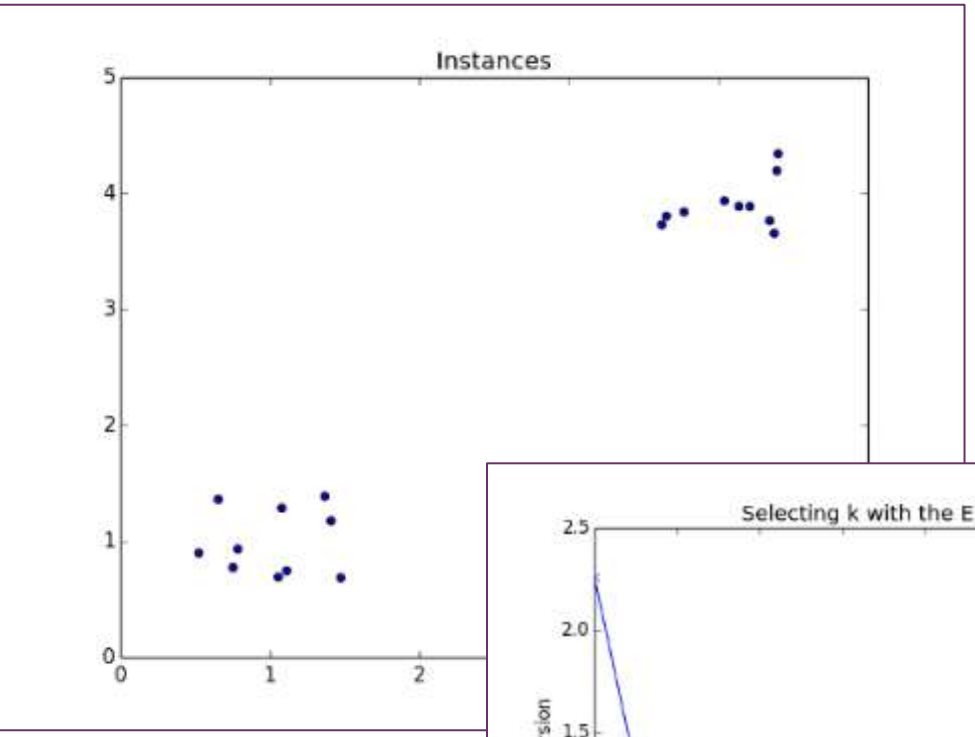
- **Step**

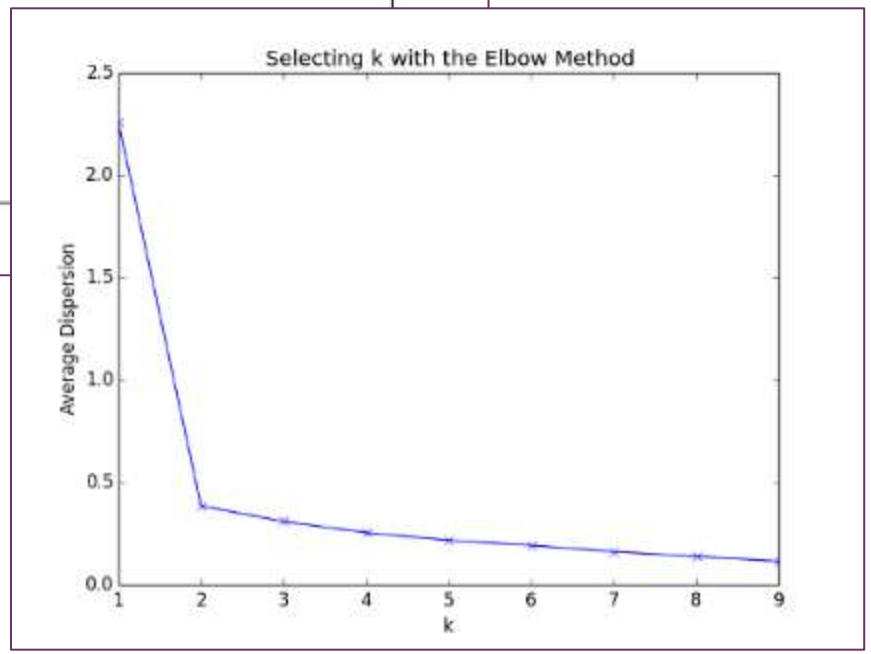
1. Initial random k cluster centers
2. Assign data to closest center
3. Update cluster center using average
4. Repeat step 2 and 3
5. Stop when
   - until convergence
   - reach max iteration

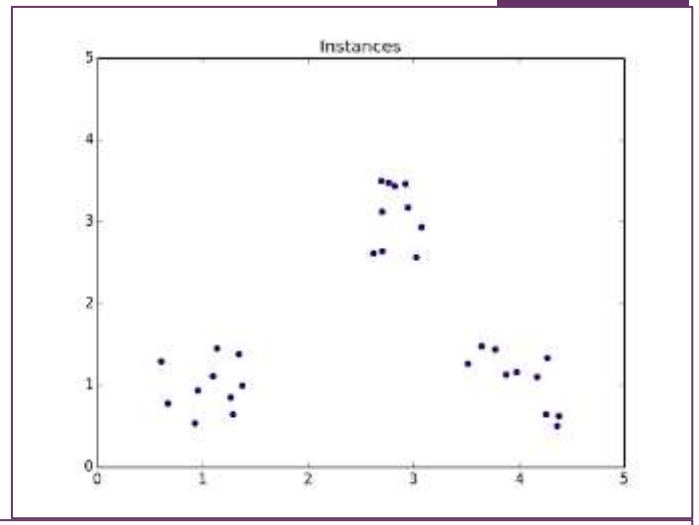


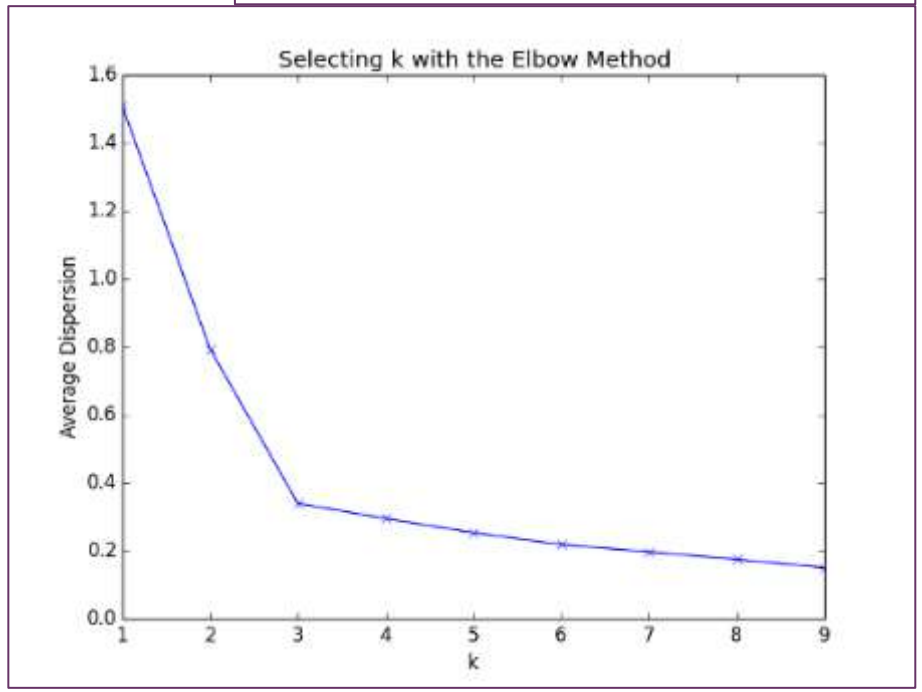
Determine the number of *k*: Elbow Method

https://www.packtpub.com/books/content/clustering-k-means









Prev Up Next

scikit-learn 0.22.2
Other versions

Please cite us if you use the software.

sklearn.preprocessing.StandardScaler
Examples using sklearn.preprocessing.StandardS

sklearn.preprocessing.StandardScaler

class sklearn.preprocessing.**StandardScaler**(*copy=True*, *with_mean=True*, *with_std=True*) [source]

Standardize features by removing the mean and scaling to unit variance

The standard score of a sample `x` is calculated as:

z = (x - u) / s

where `u` is the mean of the training samples or zero if `with_mean=False`, and `s` is the standard deviation of the training samples or one if `with_std=False`.

Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Mean and standard deviation are then stored to be used on later data using `transform`.

Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data (e.g. Gaussian with 0 mean and unit variance).

For instance many elements used in the objective function of a learning algorithm (such as the RBF kernel of Support Vector Machines or the L1 and L2 regularizers of linear models) assume that all features are centered around 0 and have variance in the same order. If a feature has a variance that is orders of magnitude larger that others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected.

This scaler can also be applied to sparse CSR or CSC matrices by passing `with_mean=False` to avoid breaking the sparsity structure of the data.

Toggle Menu

Clustering - K-means

```
>>> from sklearn.cluster import KMeans
>>> import numpy as np
>>> X = np.array([[1, 2], [1, 4], [1, 0],
...               [10, 2], [10, 4], [10, 0]])
>>> kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
>>> kmeans.labels_
array([1, 1, 1, 0, 0, 0], dtype=int32)
>>> kmeans.predict([[0, 0], [12, 3]])
array([1, 0], dtype=int32)
>>> kmeans.cluster_centers_
array([[10.,  2.],
       [ 1.,  2.]])
```

sklearn.cluster.KMeans

class `sklearn.cluster.KMeans`(*n_clusters=8, *, init='k-means++', n_init=10, max_iter=300, tol=0.0001, precompute_distances='deprecated', verbose=0, random_state=None, copy_x=True, n_jobs='deprecated', algorithm='auto'*) [source]

K-Means clustering.

Read more in the User Guide.

Parameters:

n_clusters : *int, default=8*
The number of clusters to form as well as the number of centroids to generate.

init : *{'k-means++', 'random', ndarray, callable}, default='k-means++'*
Method for initialization:

'k-means++' : selects initial cluster centers for k-mean clustering in a smart way to speed up convergence. See section Notes in k_init for more details.

'random': choose `n_clusters` observations (rows) at random from data for the initial centroids.

If an ndarray is passed, it should be of shape (n_clusters, n_features) and gives the initial centers.

If a callable is passed, it should take arguments X, n_clusters and a random state and return an initialization.

n_init : *int, default=10*
Number of time the k-means algorithm will be run with different centroid seeds. The final results will be the best output of n_init consecutive runs in terms of inertia.

max_iter : *int, default=300*
Maximum number of iterations of the k-means algorithm for a single run.

Reference : https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Prev Up Next

scikit-learn 1.1.2
Other versions

Please cite us if you use the software.

sklearn.metrics.silhouette_score
Examples using sklearn.metrics.silhouette_score

sklearn.metrics.silhouette_score

`sklearn.metrics.silhouette_score(X, labels, *, metric='euclidean', sample_size=None, random_state=None, **kwds)` [source]

Compute the mean Silhouette Coefficient of all samples.

The Silhouette Coefficient is calculated using the mean intra-cluster distance (`a`) and the mean nearest-cluster distance (`b`) for each sample. The Silhouette Coefficient for a sample is `(b - a) / max(a, b)`. To clarify, `b` is the distance between a sample and the nearest cluster that the sample is not a part of. Note that Silhouette Coefficient is only defined if number of labels is `2 <= n_labels <= n_samples - 1`.

This function returns the mean Silhouette Coefficient over all samples. To obtain the values for each sample, use `silhouette_samples`.

The best value is 1 and the worst value is -1. Values near 0 indicate overlapping clusters. Negative values generally indicate that a sample has been assigned to the wrong cluster, as a different cluster is more similar.

Read more in the User Guide.

Parameters::

X : *array-like of shape (n_samples_a, n_samples_a) if metric == "precomputed" or (n_samples_a, n_features) otherwise*
An array of pairwise distances between samples, or a feature array.

labels : *array-like of shape (n_samples,)*
Predicted labels for each sample.

metric : *str or callable, default='euclidean'*
The metric to use when calculating distance between instances in a feature array. If metric is a string, it must be one of the options allowed by `metrics.pairwise.pairwise_distances`. If `X` is the distance array itself, use `metric="precomputed"`.

sample_size : *int, default=None*
The size of the sample to use when computing the Silhouette Coefficient on a random subset of the data. If `sample_size is None`, no sampling is used.

Clustering
2) DBSCAN

- Density based algorithm
- **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise
- It groups points that are closely packed together
- Then expanding clusters in any direction where there are nearby points
- this algorithm does not requires the number of clusters to be specified
- Can identify outliers as noises
- Doesn't perform well when the clusters are of varying density
- Can form any shape of cluster
- Expensive computation to high-dimensional data

Clustering - DBSCAN

Step

1. Initial starting point that has not been visited

2. All points which are within the r distance (eps) are neighborhood points

3. If (number of neighborhood points > minPoints)
   - Assign all neighborhood points to same cluster
   - Repeat step 2 – 3 on all neighborhood points
   - marked all of these point as "visited"

Else
   - the point will be labeled as noise
   - marked all of these point as "visited"

4. Repeat step 1-3 until all points are visited

Clustering - DBSCAN

Step

minPoints = 3

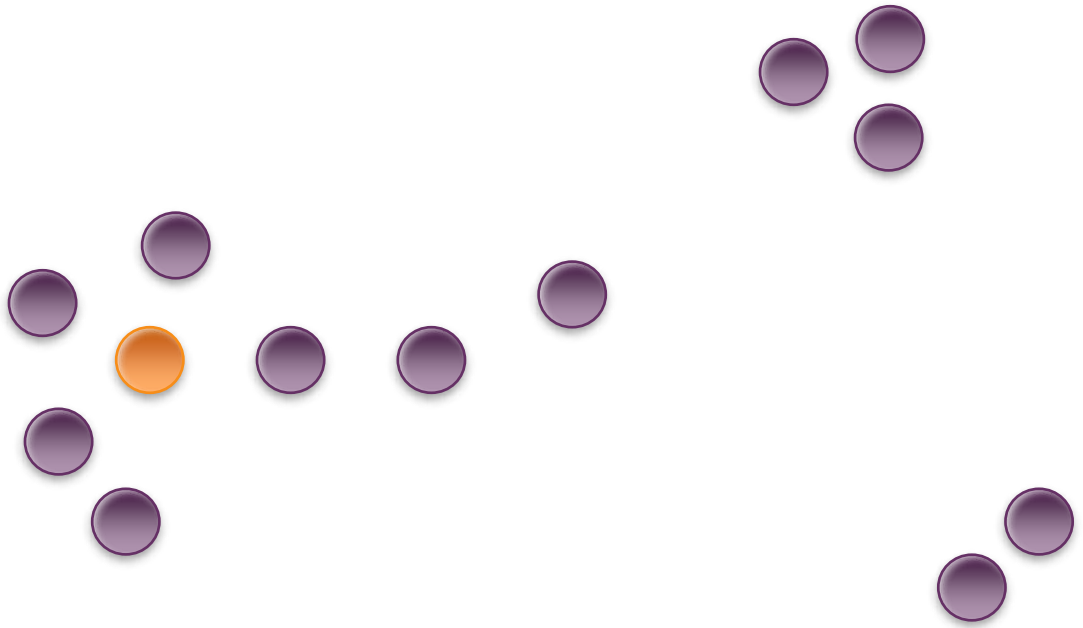
1. Initial starting point that has not been visited

2. All points which are within the r distance are neighborhood points

3. If (number of neighborhood points > minPoints)
   - Assign all neighborhood points to same cluster
   - Repeat step 2 – 3 on all neighborhood points
   - marked all of these point as "visited"

Else
   - the point will be labeled as noise
   - marked all of these point as "visited"

4. Repeat step 1-3 until all points are visited



Clustering - DBSCAN

Step

1. Initial starting point that has not been visited

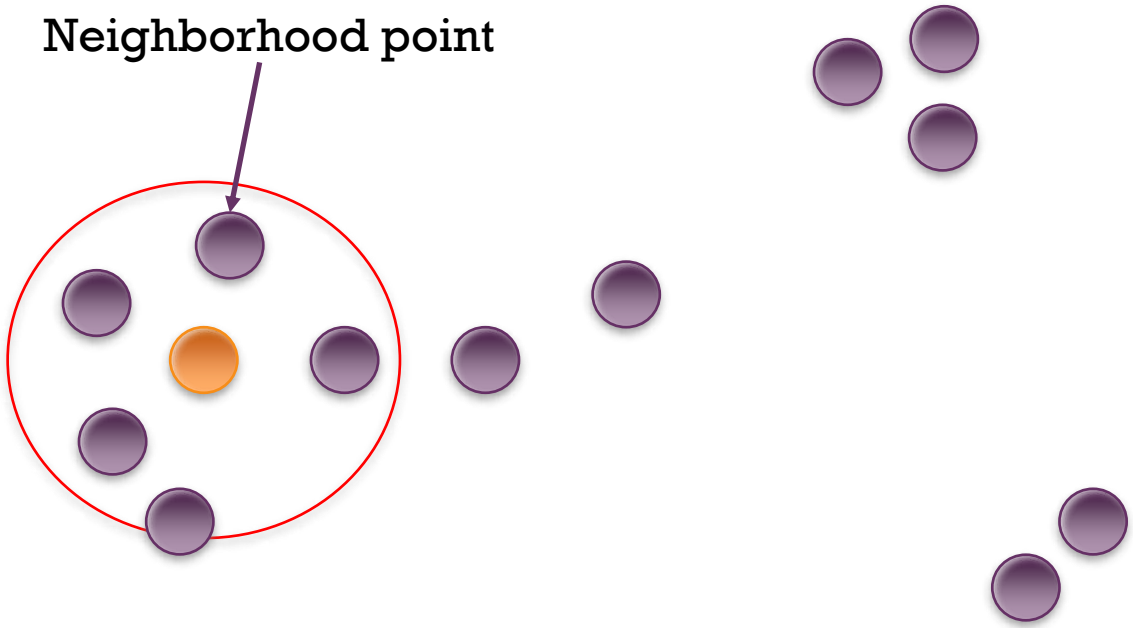
2. All points which are within the r distance are neighborhood points

3. If (number of neighborhood points > minPoints)
   - Assign all neighborhood points to same cluster
   - Repeat step 2 – 3 on all neighborhood points
   - marked all of these point as "visited"

Else
   - the point will be labeled as noise
   - marked all of these point as "visited"

4. Repeat step 1-3 until all points are visited

minPoints = 3



Clustering - DBSCAN

Step

1. Initial starting point that has not been visited

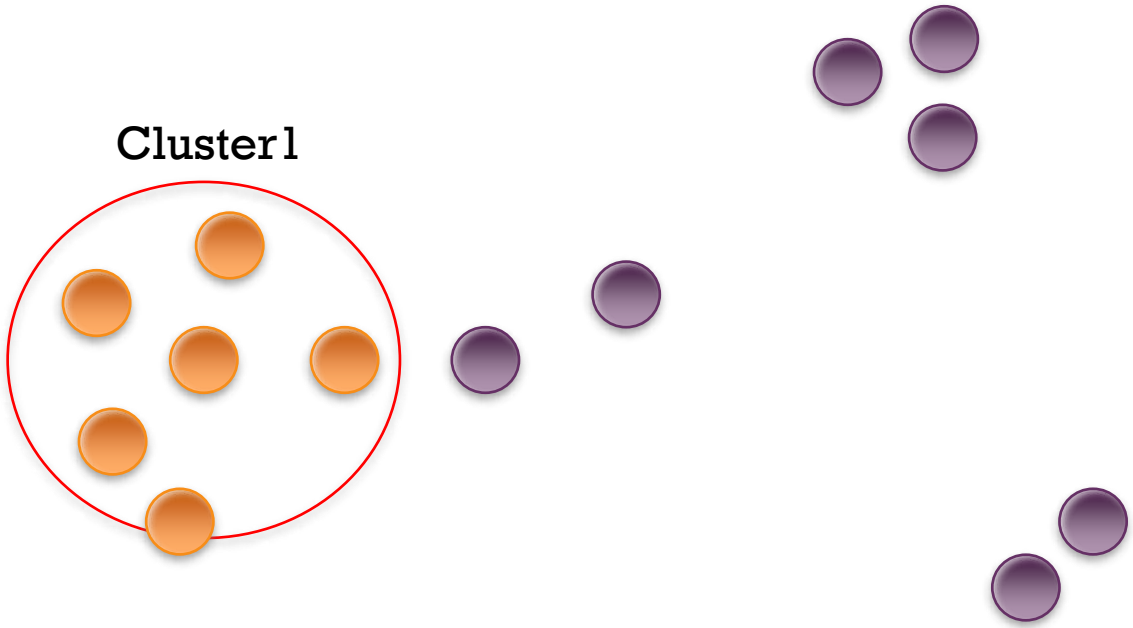
2. All points which are within the r distance are neighborhood points

3. If (number of neighborhood points > minPoints)
   - Assign all neighborhood points to same cluster
   - Repeat step 2 – 3 on all neighborhood points
   - marked all of these point as "visited"

Else
   - the point will be labeled as noise
   - marked all of these point as "visited"

4. Repeat step 1-3 until all points are visited

minPoints = 3



Clustering - DBSCAN

Step

1. Initial starting point that has not been visited

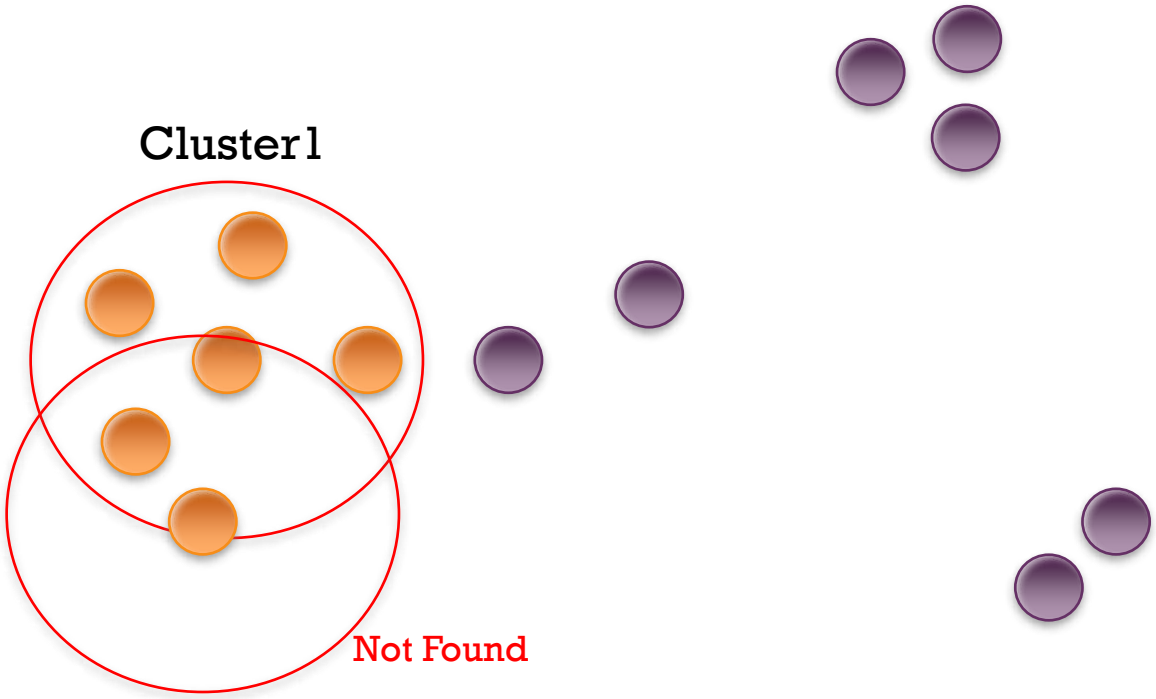
2. All points which are within the r distance are neighborhood points

3. If (number of neighborhood points > minPoints)
   - Assign all neighborhood points to same cluster
   - Repeat step 2 – 3 on all neighborhood points
   - marked all of these point as "visited"

Else
   - the point will be labeled as noise
   - marked all of these point as "visited"

4. Repeat step 1-3 until all points are visited

minPoints = 3



Clustering - DBSCAN

Step

minPoints = 3

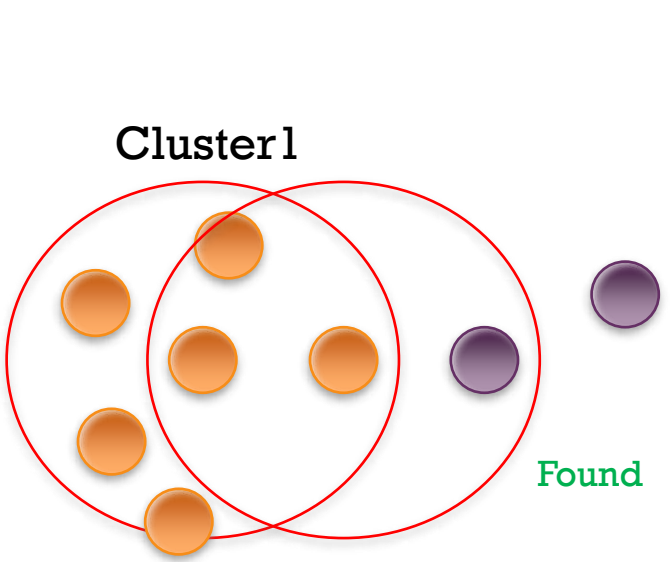
1. Initial starting point that has not been visited

2. All points which are within the r distance are neighborhood points

3. If (number of neighborhood points > minPoints)
   - Assign all neighborhood points to same cluster
   - Repeat step 2 – 3 on all neighborhood points
   - marked all of these point as "visited"

Else
   - the point will be labeled as noise
   - marked all of these point as "visited"

4. Repeat step 1-3 until all points are visited



Clustering - DBSCAN

Step

1. Initial starting point that has not been visited

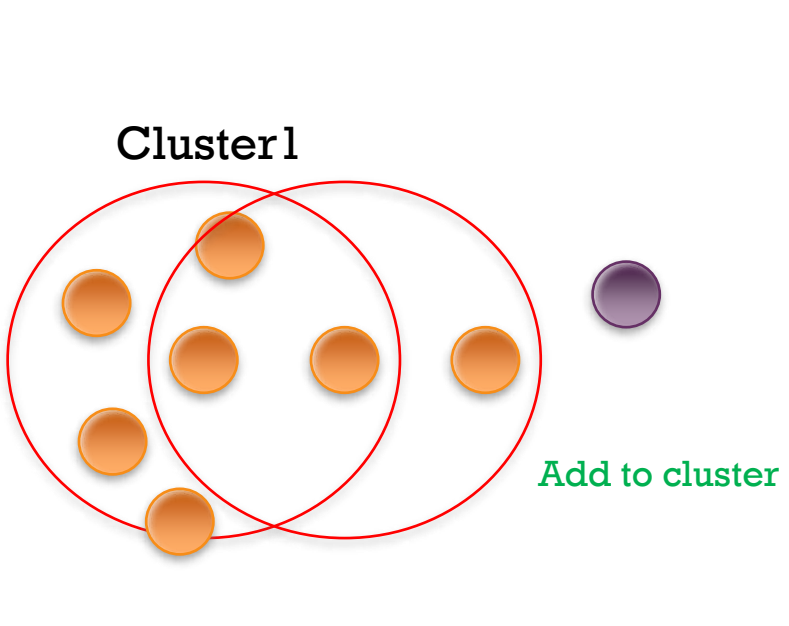
2. All points which are within the r distance are neighborhood points

3. If (number of neighborhood points > minPoints)
   - Assign all neighborhood points to same cluster
   - Repeat step 2 – 3 on all neighborhood points
   - marked all of these point as "visited"

Else
   - the point will be labeled as noise
   - marked all of these point as "visited"

4. Repeat step 1-3 until all points are visited

minPoints = 3



Clustering - DBSCAN

Step

minPoints = 3

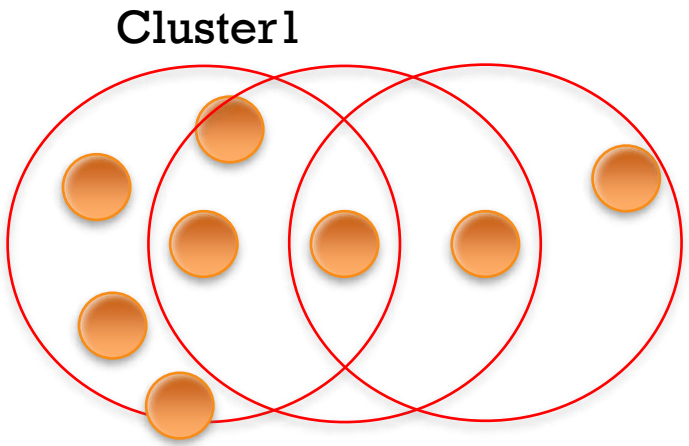
1. Initial starting point that has not been visited

2. All points which are within the r distance are neighborhood points

3. If (number of neighborhood points > minPoints)
   - Assign all neighborhood points to same cluster
   - Repeat step 2 – 3 on all neighborhood points
   - marked all of these point as "visited"

Else
   - the point will be labeled as noise
   - marked all of these point as "visited"

4. Repeat step 1-3 until all points are visited



Clustering - DBSCAN

Step

1. Initial starting point that has not been visited

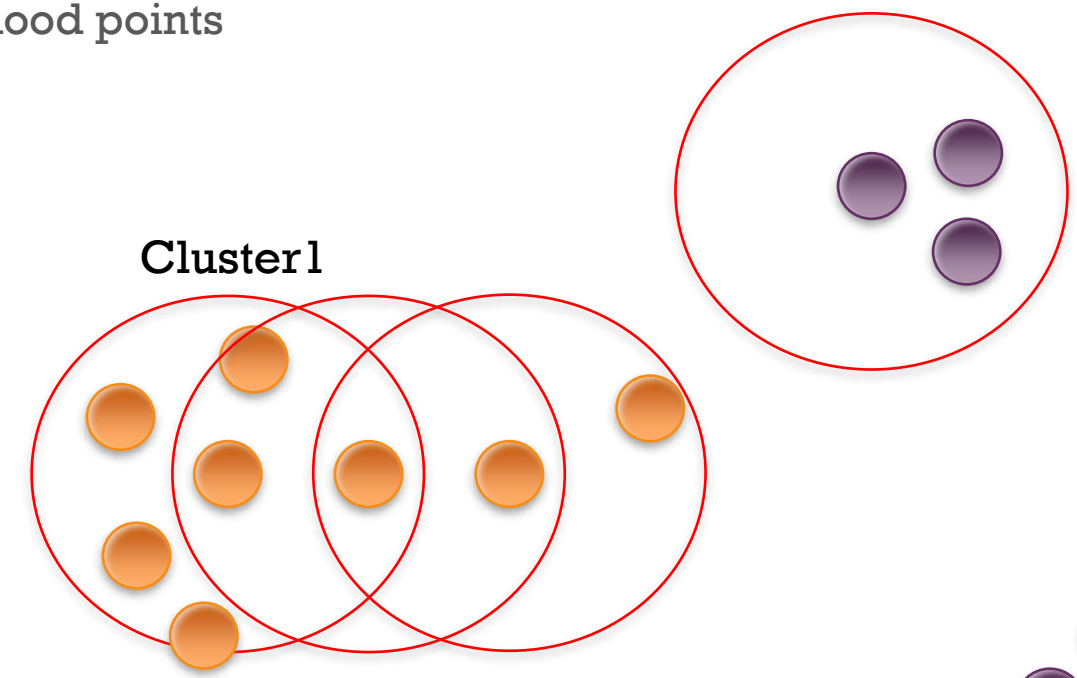
2. All points which are within the r distance are neighborhood points

3. If (number of neighborhood points > minPoints)
   - Assign all neighborhood points to same cluster
   - Repeat step 2 – 3 on all neighborhood points
   - marked all of these point as "visited"

Else
   - the point will be labeled as noise
   - marked all of these point as "visited"

4. Repeat step 1-3 until all points are visited

minPoints = 3



Clustering - DBSCAN

Step

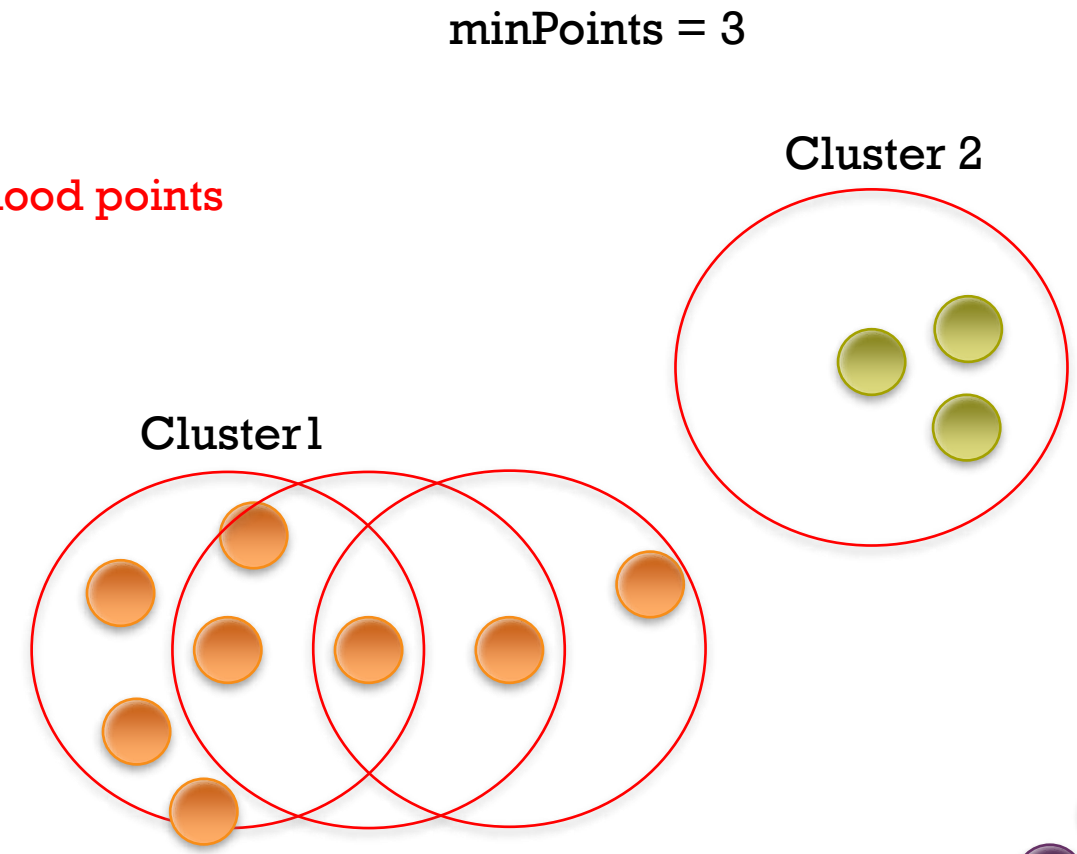
1. Initial starting point that has not been visited

2. All points which are within the r distance are neighborhood points

3. If (number of neighborhood points > minPoints)
   - Assign all neighborhood points to same cluster
   - Repeat step 2 – 3 on all neighborhood points
   - marked all of these point as "visited"

Else
   - the point will be labeled as noise
   - marked all of these point as "visited"

4. Repeat step 1-3 until all points are visited



Clustering - DBSCAN

Step

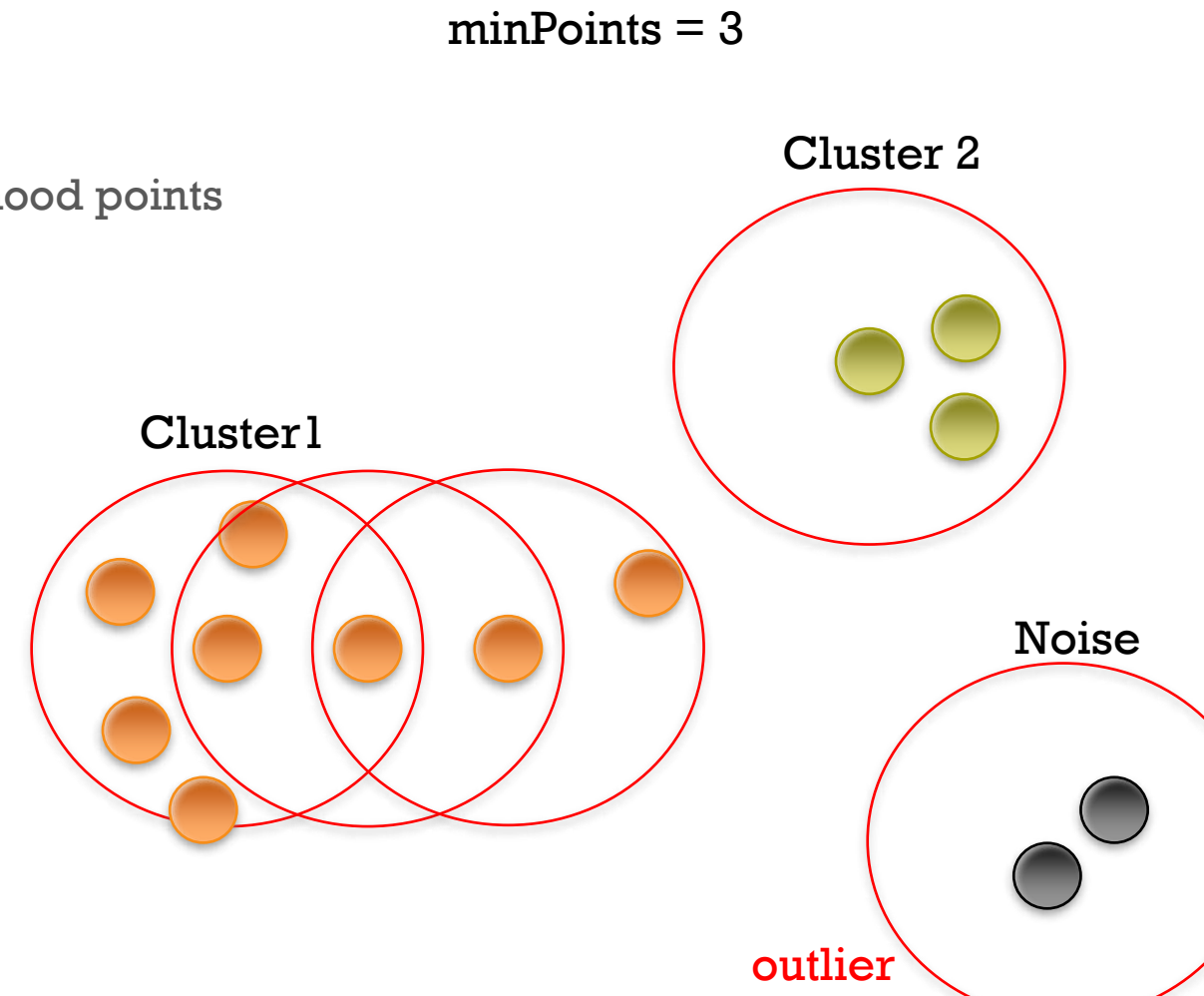
1. Initial starting point that has not been visited

2. All points which are within the r distance are neighborhood points

3. If (number of neighborhood points > minPoints)
   - Assign all neighborhood points to same cluster
   - Repeat step 2 – 3 on all neighborhood points
   - marked all of these point as "visited"

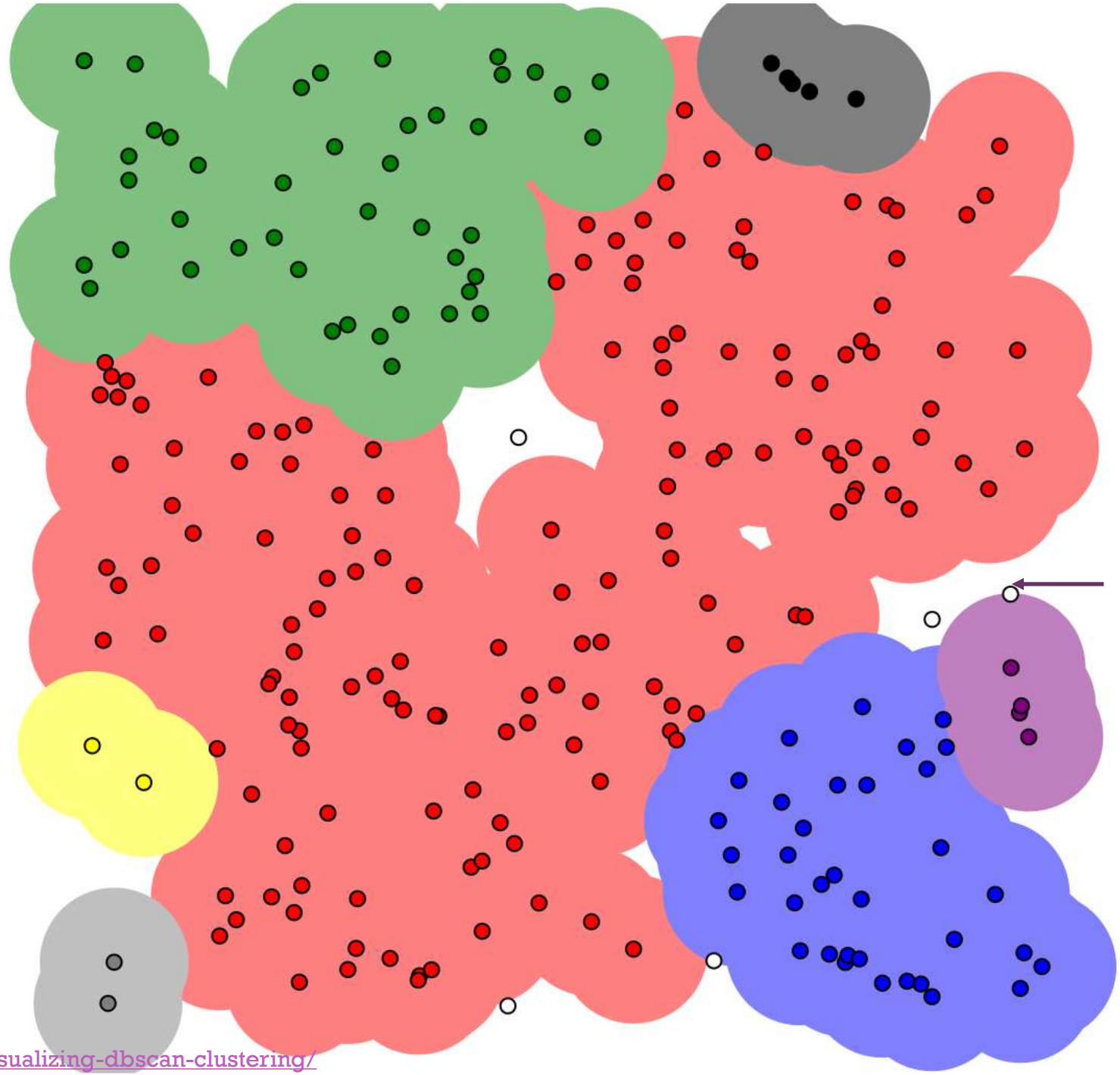
Else
   - the point will be labeled as noise
   - marked all of these point as "visited"

4. Repeat step 1-3 until all points are visited



Clustering - DBSCAN

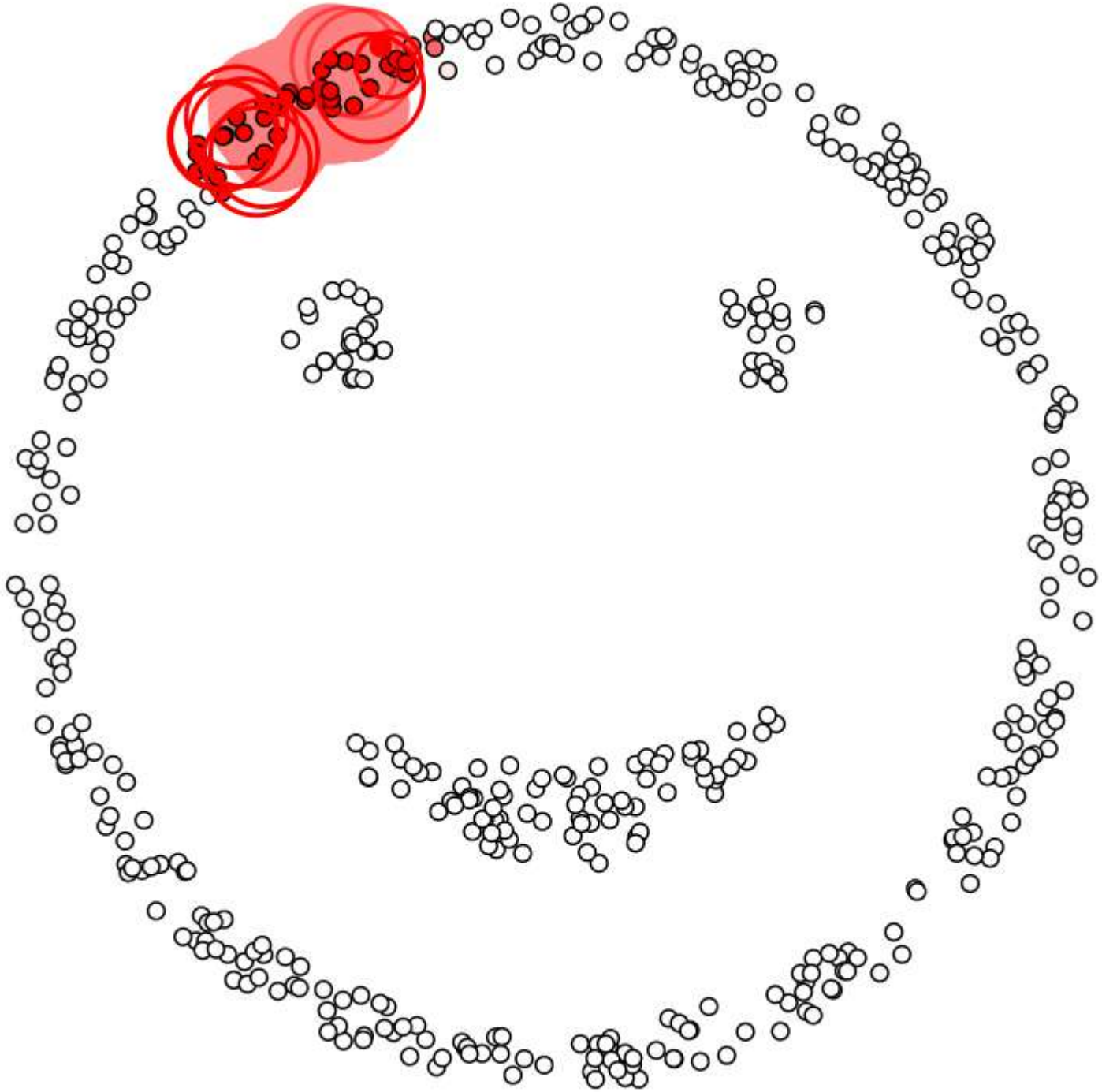
- Try yourself
- Round 1
  - Epsilon =1
  - minPoints = 4
- Round 1
  - Epsilon =1
  - minPoints = 2
- Round 1
  - Epsilon =1.5
  - minPoints = 4



Can't form cluster, outlier

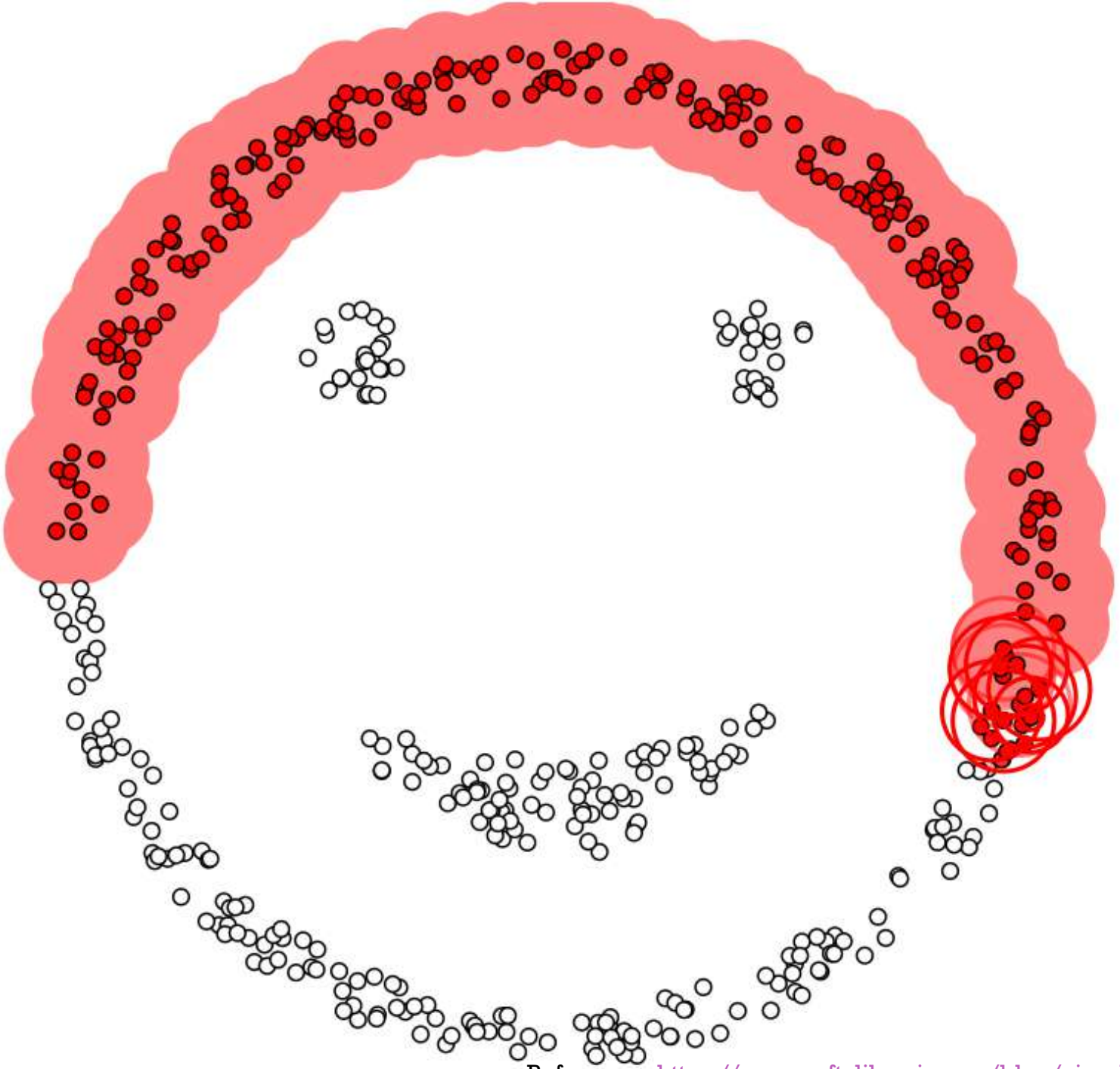
Reference : https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

Clustering - DBSCAN



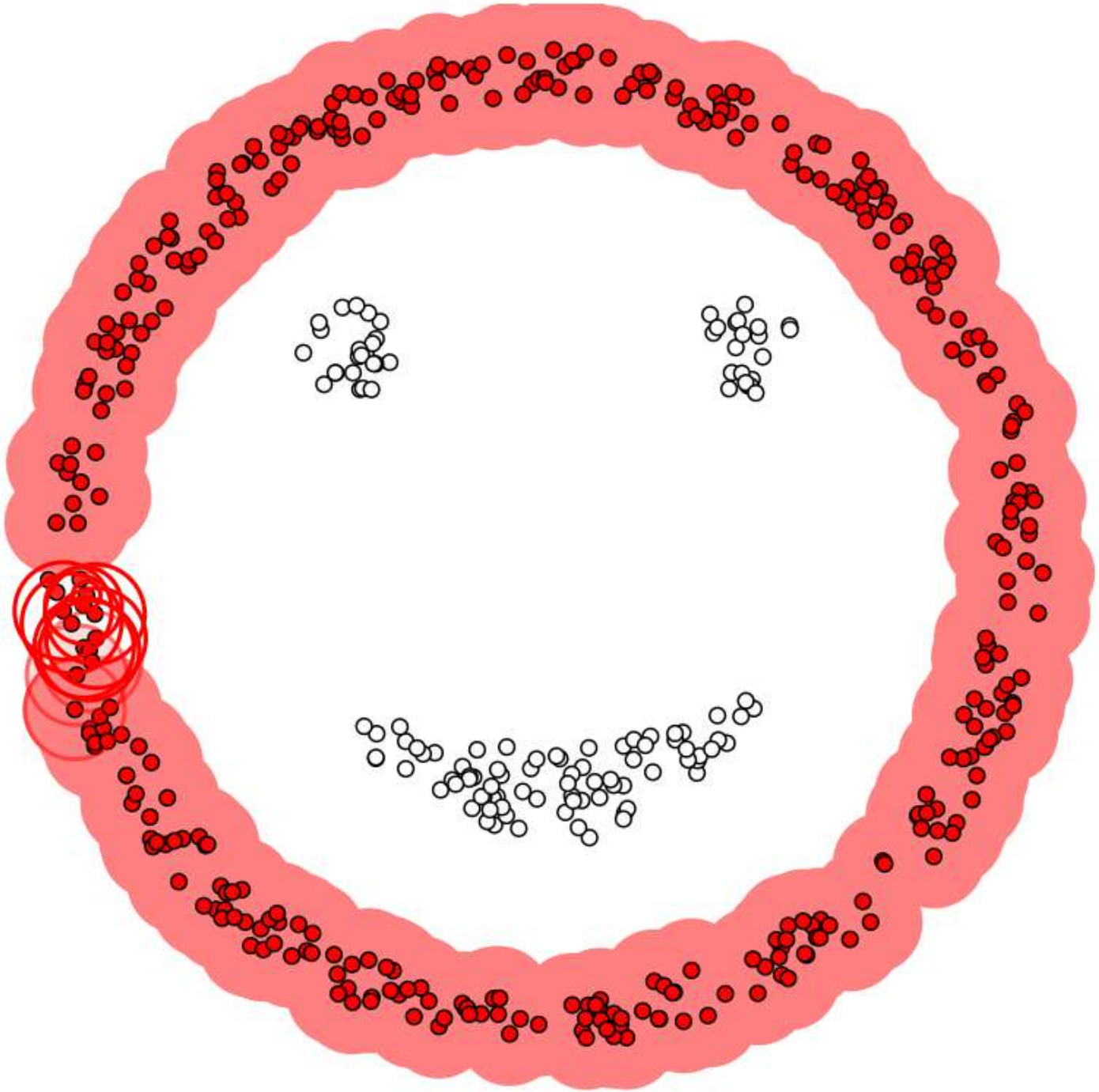
Reference : https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

Clustering - DBSCAN



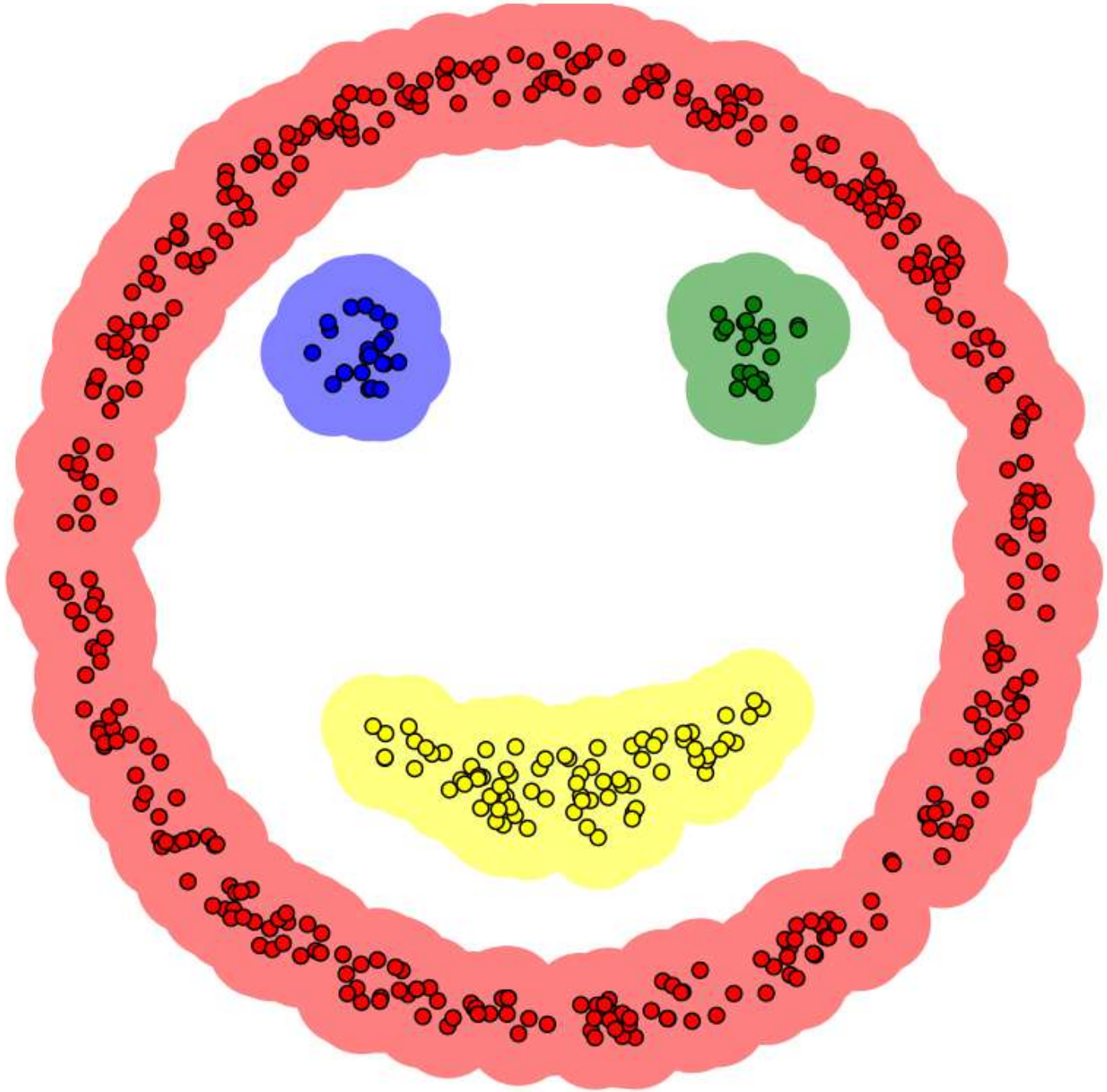
Reference : https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

Clustering - DBSCAN



Reference : https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

Clustering - DBSCAN



Reference : https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

Clustering - DBSCAN

sklearn.cluster.DBSCAN

class `sklearn.cluster.` **`DBSCAN`**(*eps=0.5, *, min_samples=5, metric='euclidean', metric_params=None, algorithm='auto', leaf_size=30, p=None, n_jobs=None*) [source]

Perform DBSCAN clustering from vector array or distance matrix.

DBSCAN - Density-Based Spatial Clustering of Applications with Noise. Finds core samples of high density and expands clusters from them. Good for data which contains clusters of similar density.

Read more in the User Guide.

Parameters:

eps : *float, default=0.5*
The maximum distance between two samples for one to be considered as in the neighborhood of the other. This is not a maximum bound on the distances of points within a cluster. This is the most important DBSCAN parameter to choose appropriately for your data set and distance function.

min_samples : *int, default=5*
The number of samples (or total weight) in a neighborhood for a point to be considered as a core point. This includes the point itself.

metric : *string, or callable, default='euclidean'*
The metric to use when calculating distance between instances in a feature array. If metric is a string or callable, it must be one of the options allowed by `sklearn.metrics.pairwise_distances` for its metric parameter. If metric is "precomputed", X is assumed to be a distance matrix and must be square. X may be a Glossary, in which case only "nonzero" elements may be considered neighbors for DBSCAN.

New in version 0.17: metric *precomputed* to accept precomputed sparse matrix.

metric_params : *dict, default=None*
Additional keyword arguments for the metric function.

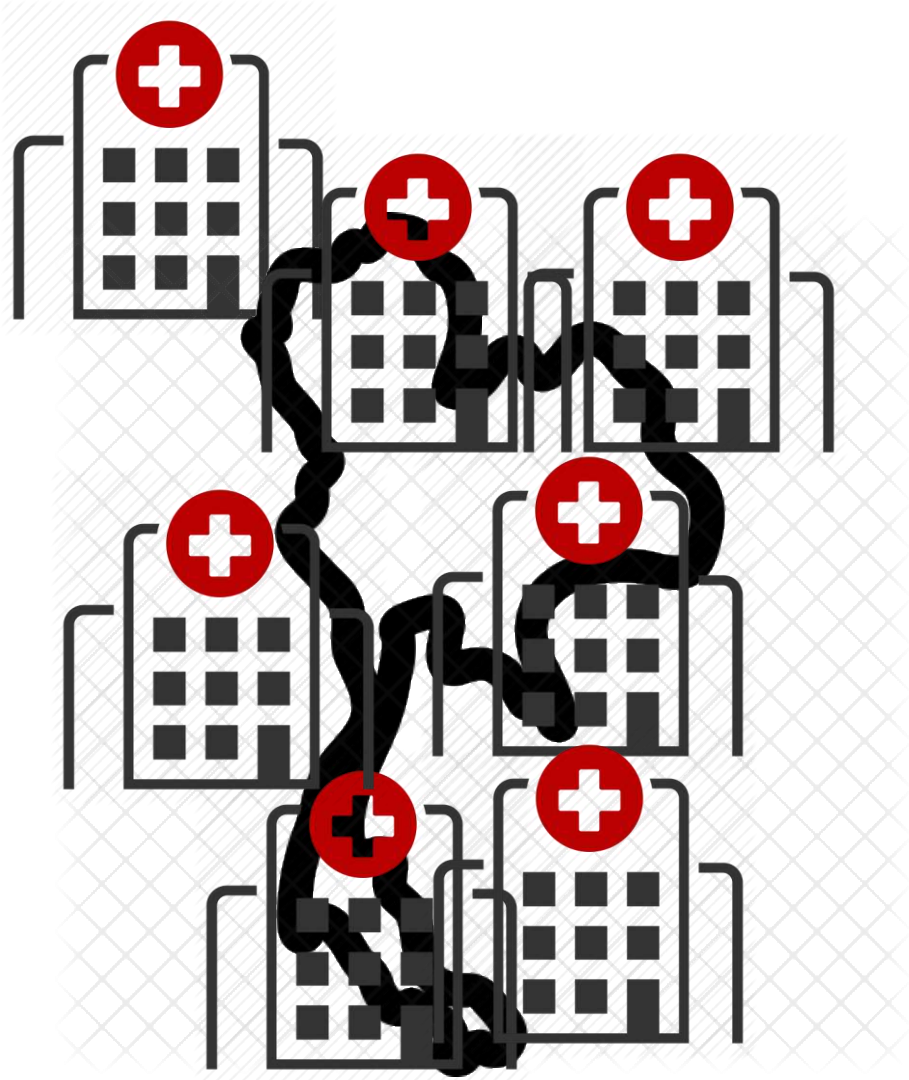
New in version 0.19.

algorithm : *{'auto', 'ball_tree', 'kd_tree', 'brute'}, default='auto'*
The algorithm to be used by the NearestNeighbors module to compute pointwise distances and find nearest neighbors. See NearestNeighbors module documentation for details.

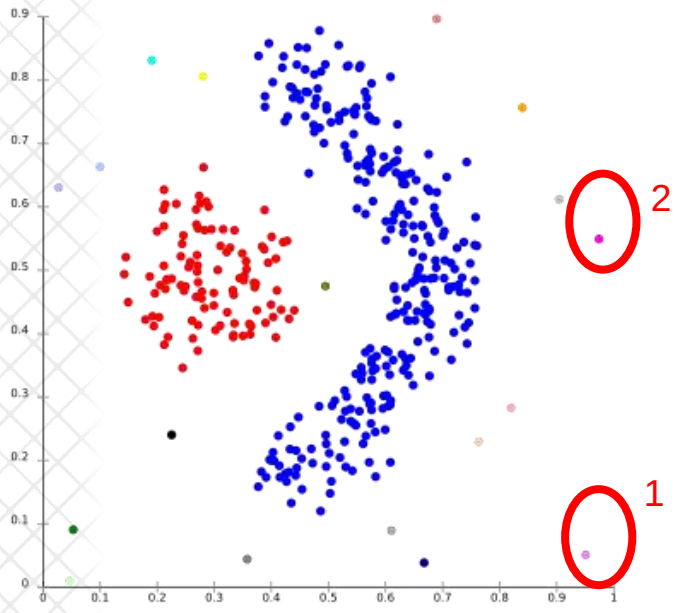
Fraud Detection (Unsupervised Learning)

สำนักงานประกันสังคม

Social Security Office



Filter



บอกได้ว่าควรตรวจสอบ รพ. ใด และควรตรวจสอบไล่จากสาเหตุใด

Market Basket Analysis

- Discover associations between items
- Count combinations of items that occur together frequently in transactions

$$Support(\{X\} \rightarrow \{Y\}) = \frac{Transactions\ containing\ both\ X\ and\ Y}{Total\ number\ of\ transactions}$$

$$Confidence(\{X\} \rightarrow \{Y\}) = \frac{Transactions\ containing\ both\ X\ and\ Y}{Transactions\ containing\ X}$$

$$Lift(\{X\} \rightarrow \{Y\}) = \frac{(Transactions\ containing\ both\ X\ and\ Y)/(Transactions\ containing\ X)}{Fraction\ of\ transactions\ containing\ Y}$$

Market Basket Analysis (Association Rule Mining)

Rule	Support	Confidence
Apple => Donut	2/5	2/3
Coconut => Apple	2/5	2/4
Apple => Coconut	2/5	2/3
Banana & Coconut => Donut	1/5	1/3

Association Rule Mining (cont.)

- Wal-Mart customers who purchase Barbie dolls have a 60% likelihood of also purchasing one of three types of candy bars [Forbes, Sept 8, 1997]

- Strategies?
  1. Put them closer together in the store.
  2. Put them far apart in the store.
  3. Package candy bars with the dolls.
  4. Package Barbie + candy + poorly selling item.
  5. Raise the price on one and lower it on the other.
  6. Offer Barbie accessories for proofs of purchase.
  7. Do not advertise candy and Barbie together.
  8. Offer candies in the shape of a Barbie doll.

Caution in Association Rule Mining

1. • Basket size: per bill, customer, day
2. • Item level: SKU, product category

mlxtend version: 0.17.2

apriori

apriori(df, min_support=0.5, use_colnames=False, max_len=None, verbose=0, low_memory=False)

Get frequent itemsets from a one-hot DataFrame

Parameters

- `df` : pandas DataFrame

  pandas DataFrame the encoded format. Also supports DataFrames with sparse data; for more info, please see (https://pandas.pydata.org/pandas-docs/stable/ user_guide/sparse.html#sparse-data-structures)

  Please note that the old pandas SparseDataFrame format is no longer supported in mlxtend >= 0.17.2.

  The allowed values are either 0/1 or True/False. For example,

```
 Apple  Bananas   Beer  Chicken   Milk   Rice
0     True    False   True     True  False   True
1     True    False   True    False  False   True
2     True    False   True    False  False  False
```

association_rules

association_rules(df, metric='confidence', min_threshold=0.8, support_only=False)

Generates a DataFrame of association rules including the metrics 'score', 'confidence', and 'lift'

Parameters

- `df` : pandas DataFrame

  pandas DataFrame of frequent itemsets with columns ['support', 'itemsets']

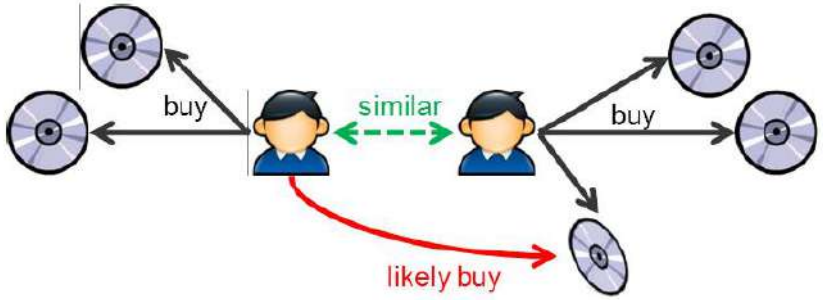
- `metric` : string (default: 'confidence')

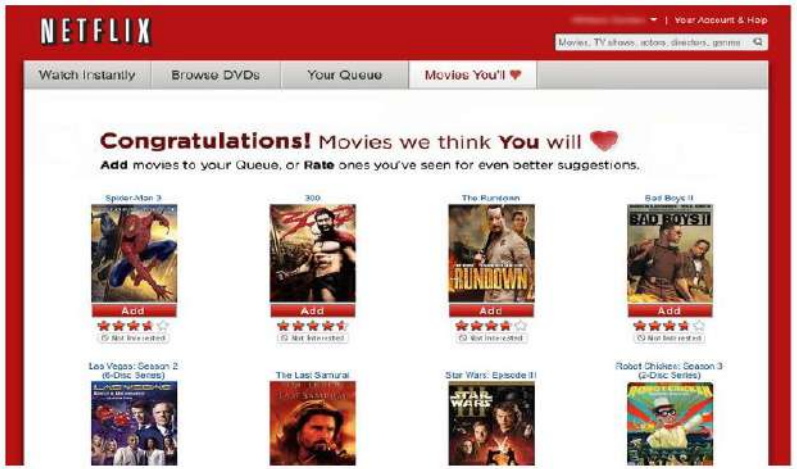
  Metric to evaluate if a rule is of interest. **Automatically set to 'support' if `support_only=True` .** Otherwise, supported metrics are 'support', 'confidence', 'lift',

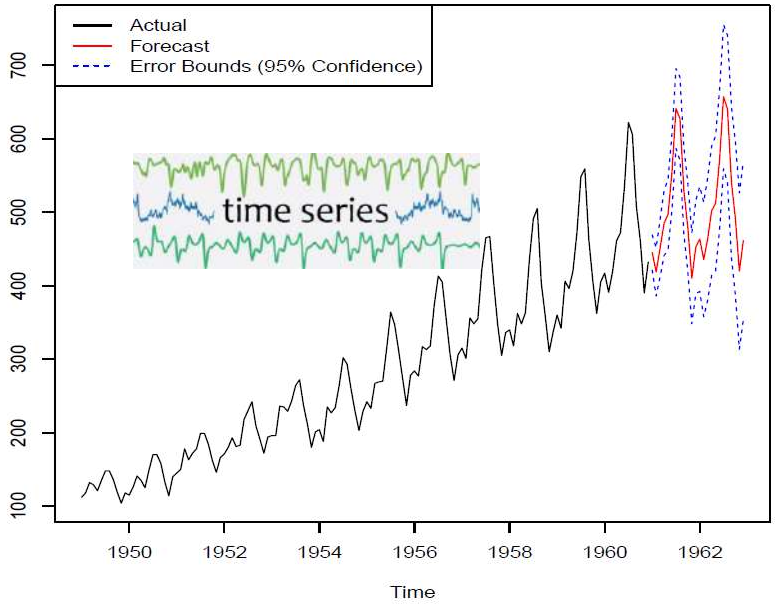
'leverage', and 'conviction' These metrics are computed as follows:

```
- support(A->C) = support(A+C) [aka 'support'], range: [0, 1]

- confidence(A->C) = support(A+C) / support(A), range: [0, 1]

- lift(A->C) = confidence(A->C) / support(C), range: [0, inf]

- leverage(A->C) = support(A->C) - support(A)*support(C),
range: [-1, 1]

- conviction = [1 - support(C)] / [1 - confidence(A->C)],
```

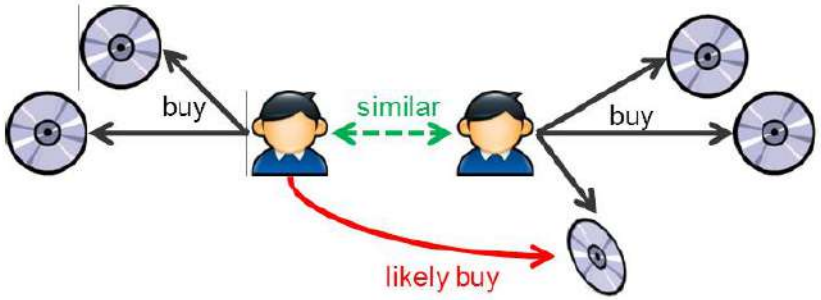
Special tasks

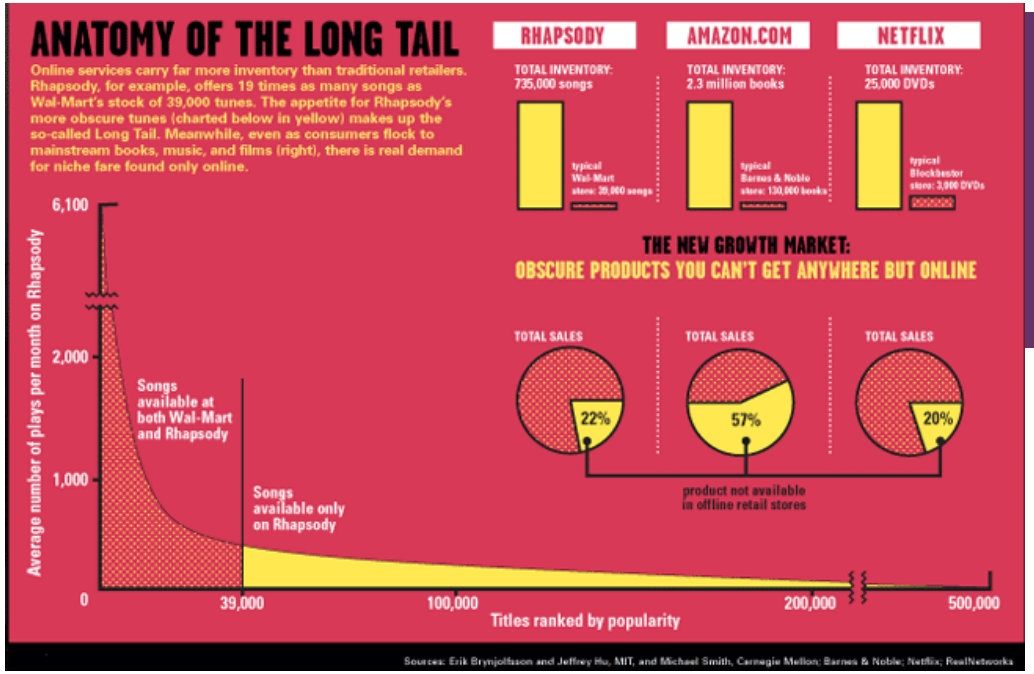


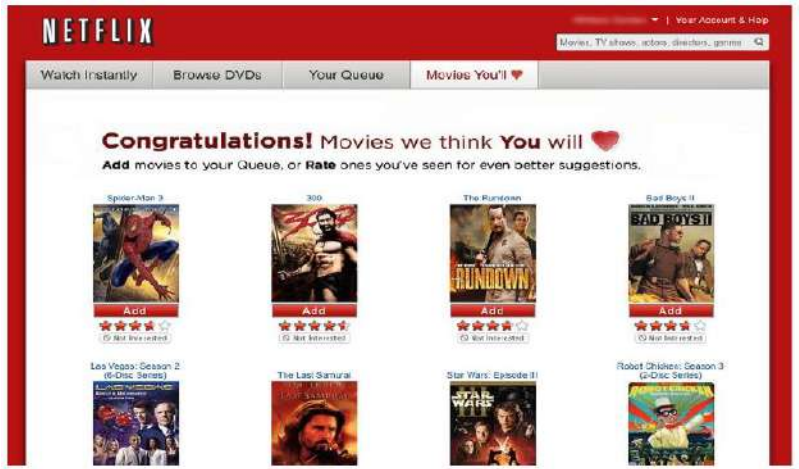




Recommendation system



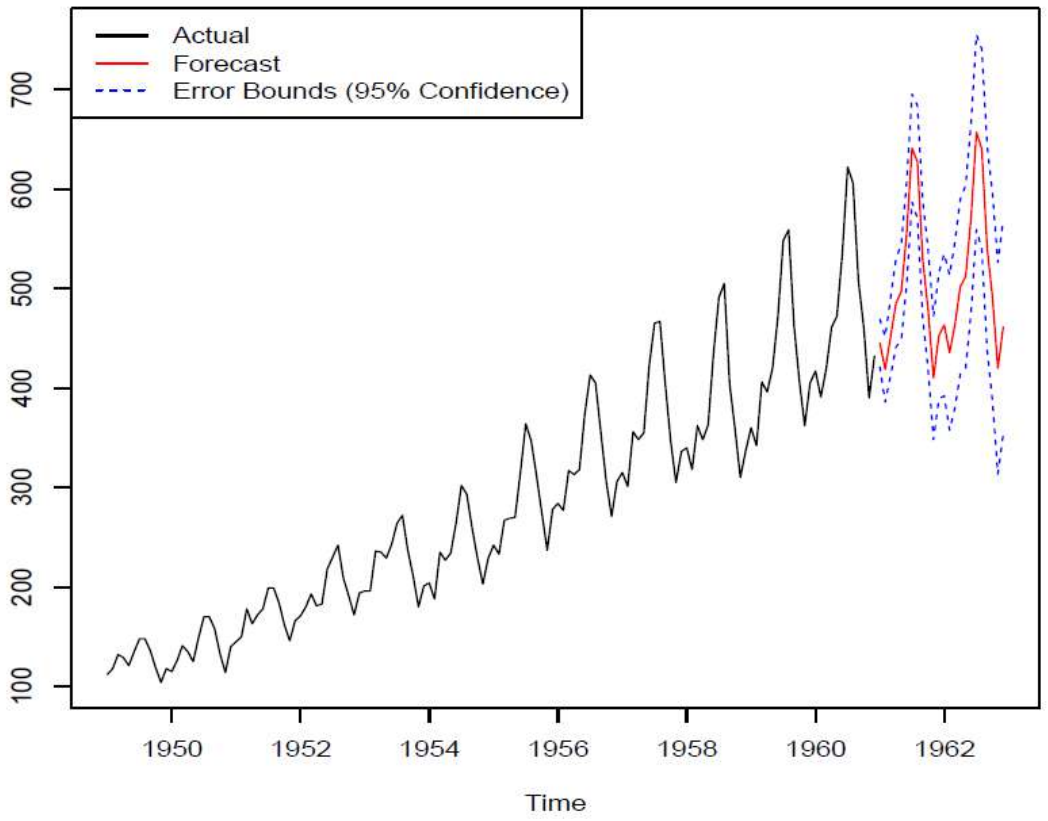




	Harry potter	X-Men	Hobbit	Argo	Pirates
101	5	2	4	?	?
102	?	?	5	2	?
103	1	2	?	?	3
104					
105					
…					

	Harry potter	X-Men	Hobbit	Argo	Pirates
101	5	2	4	1	3
102	4	1	5	2	3
103	1	2	4	1	3
104					
105					
…					

Time Series Analysis (Trend Forecasting)



- Techniques
  - **ARIMA (Autoregressive integrated moving average)**
  - Exponential Smoothing
  - Neural Networks
  - Deep Learning
- Sample Applications
  - Customer trend forecasting
  - Revenue trend forecasting
  - Rainfall forecasting
  - Remaining useful life forecasting (preventive maintenance)

Text Mining

https://ischool.syr.edu/infospace/2013/04/23/what-is-text-mining/

- Text mining, which is sometimes referred to "text analytics" is one way to make qualitative or "unstructured" data **usable by a computer**.
- Convert from unstructured to structured data

Parse → Stop → Stem

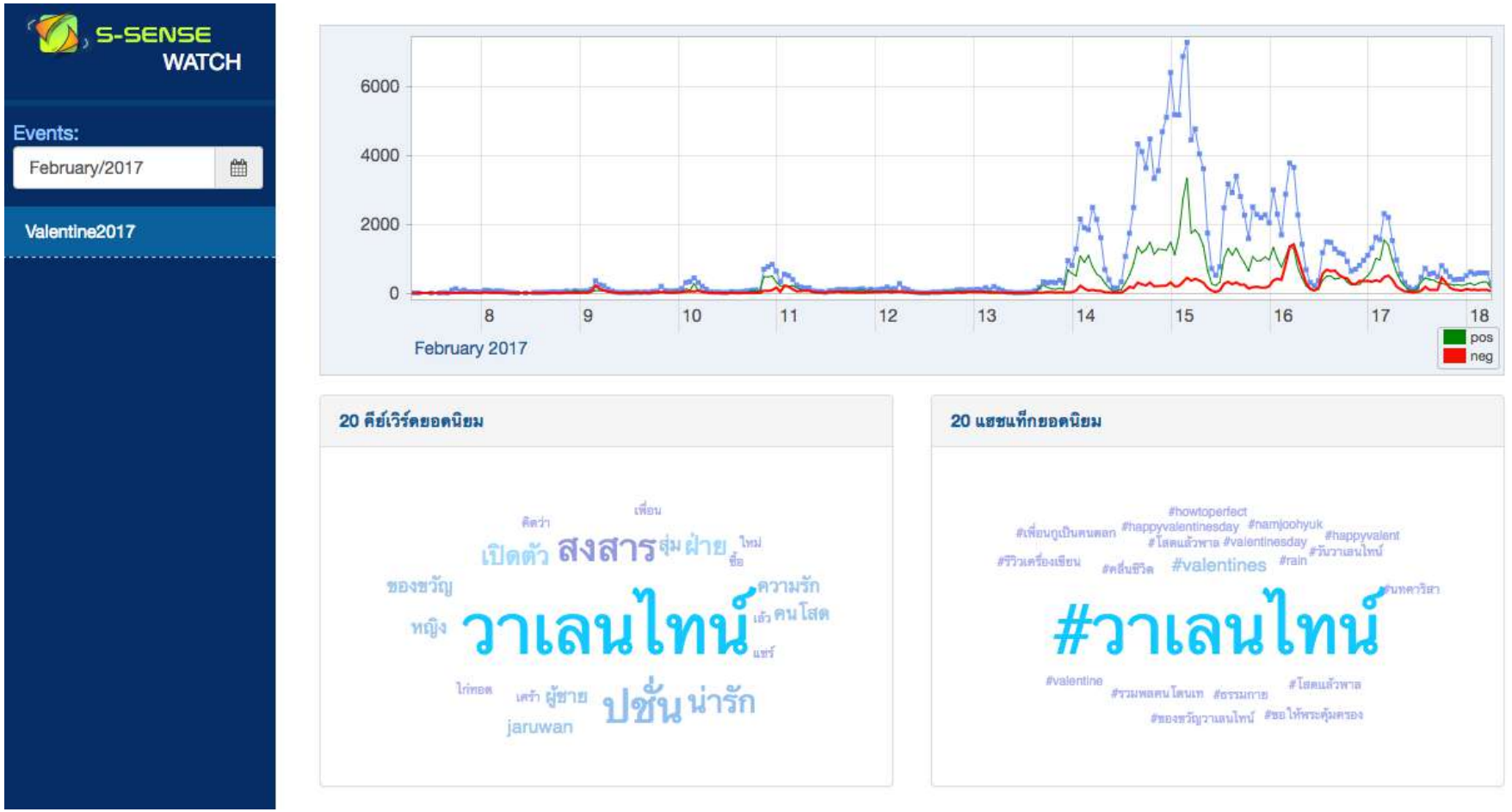
Comments	Good	Like	Hate	Sentiment
Tweet1	7	8	0	😄
Tweet2	1	0	10	😠
Tweet3	2	9	1	😄

Text Mining (cont.): Sentiment Analysis

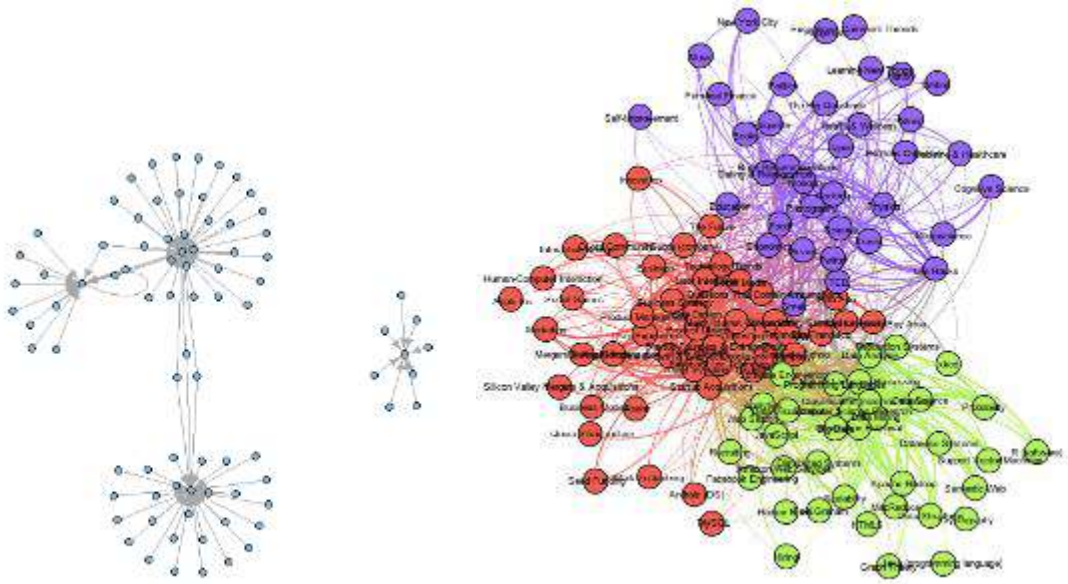
http://pop.ssense.in.th/

Text Mining (cont.): Emerging Trend Analysis

http://www.ssense.in.th/watch/



Social Network Analysis



- Techniques
  - Centrality: degree, closeness, betweenness, transitivity
  - Community detection
  - Graph Clustering
- Sample Applications
  - Influencer detection
  - Community detection

+

Any Questions?