Unsupervised Learning and Dimensionality Reduction

Learning Objectives

After completing this lecture, you will be able to:-

- Describe the K-Means algorithm
- Explain the difference between various distance measures include Cosine Distance
- Select parameters for clustering using K-Means, Heirarchical Agglomerative Clustering,
- Discuss the practice and application of dimensionality reduction

Types of Machine Learning

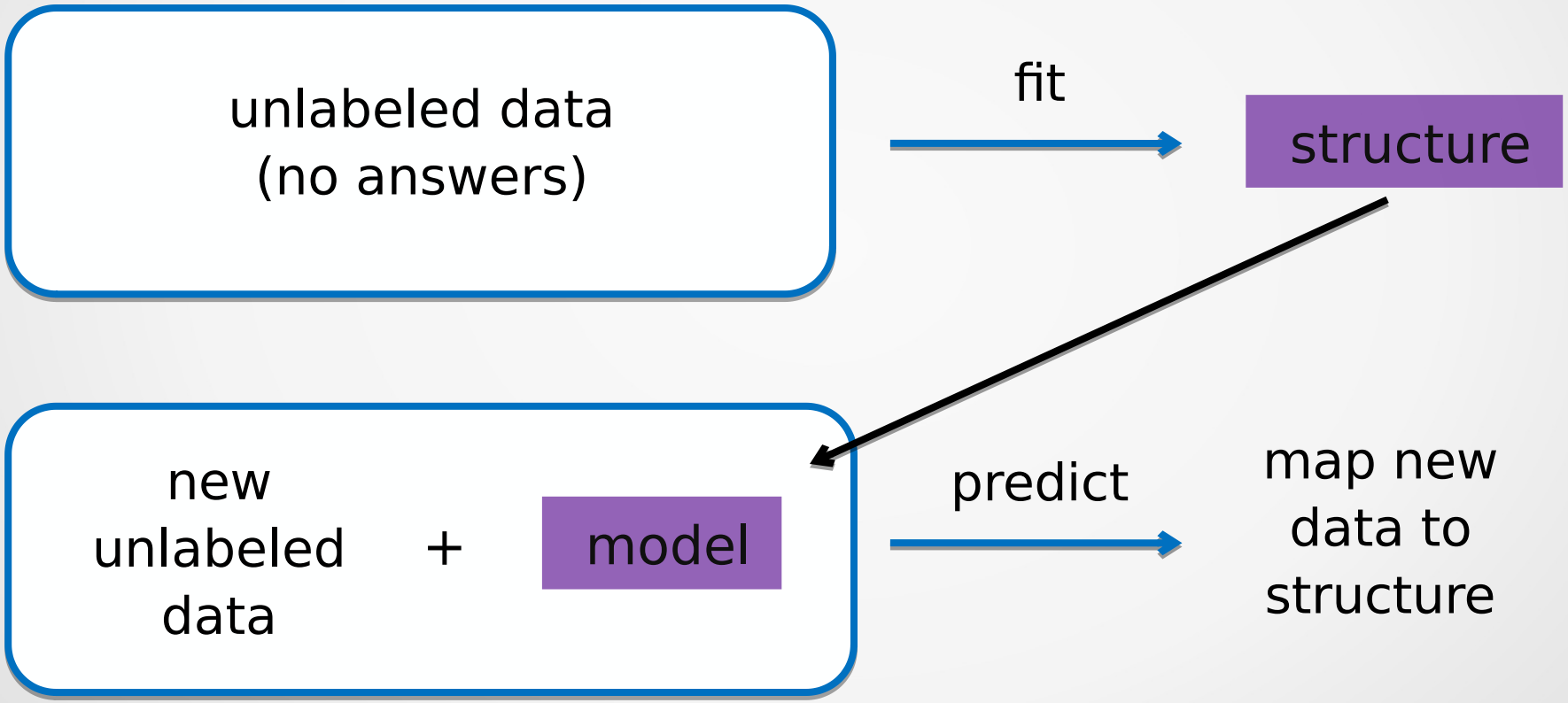
Differing methodology

- Supervised (data points have known outcome)
- Unsupervised (data points have unknown outcome)

Types of Unsupervised Learning

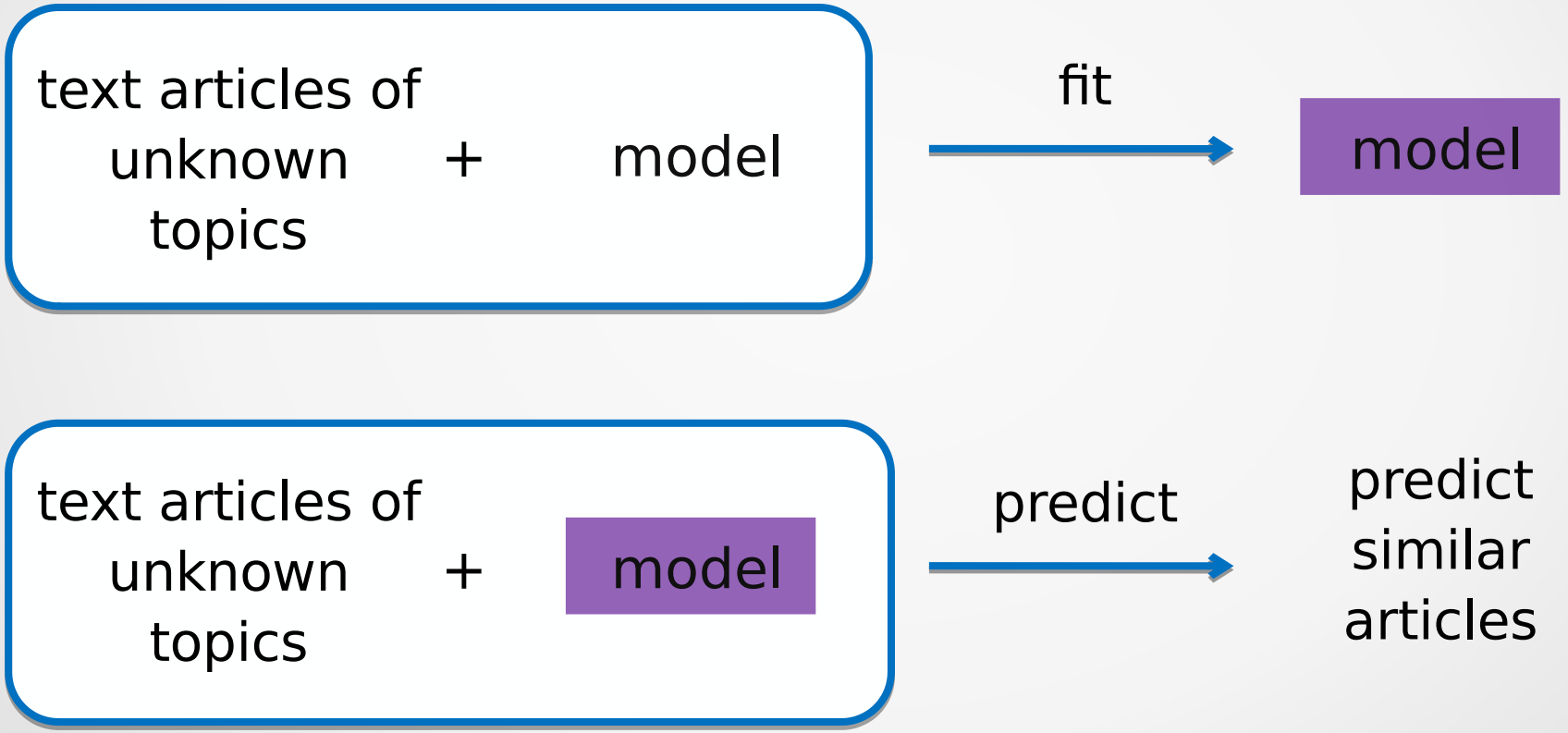
- Clustering (identify unknown structure in data)
- Dimensionality Reduction (use structural characteristics to simplify data)

Unsupervised Learning Overview



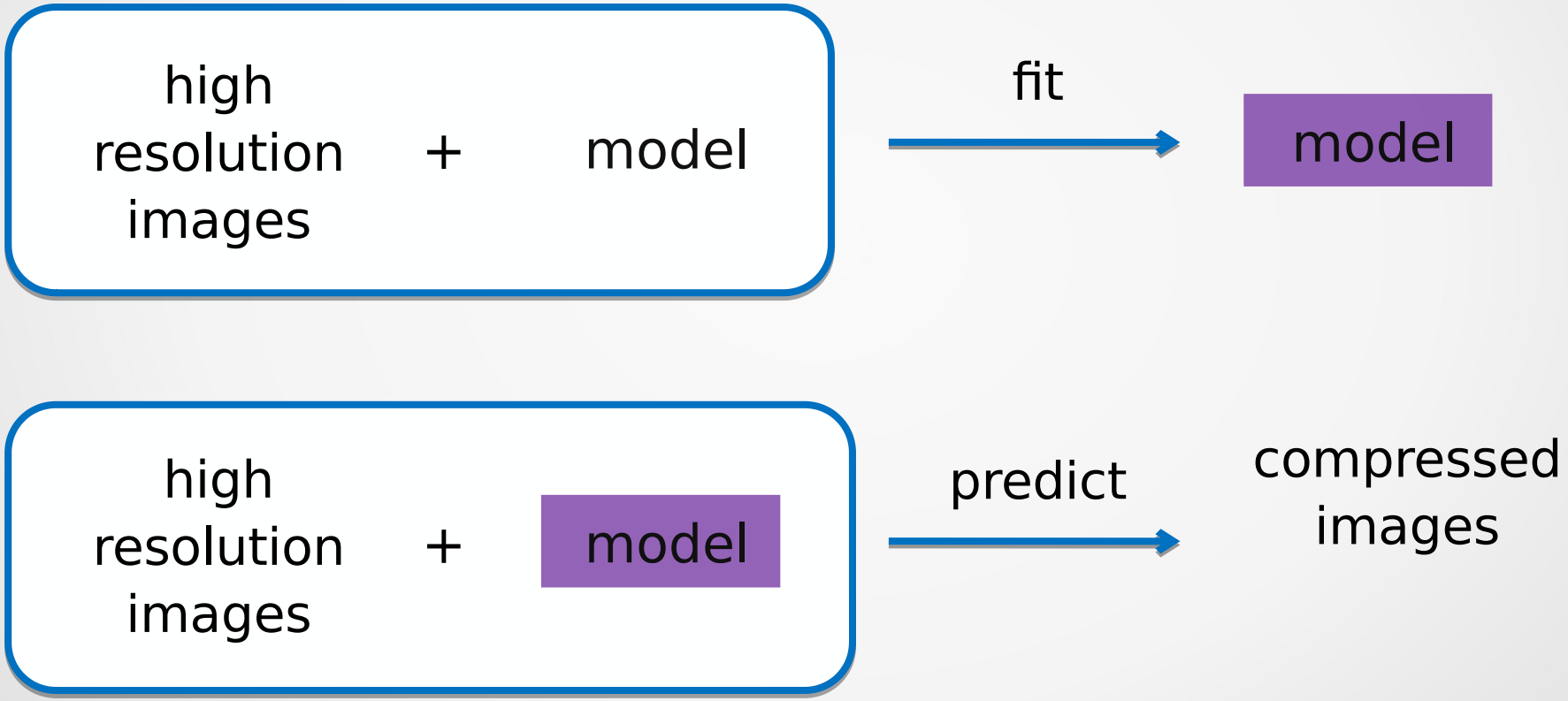
Unsupervised Learning Overview

Clustering: Finding Distinct Groups



Unsupervised Learning Overview

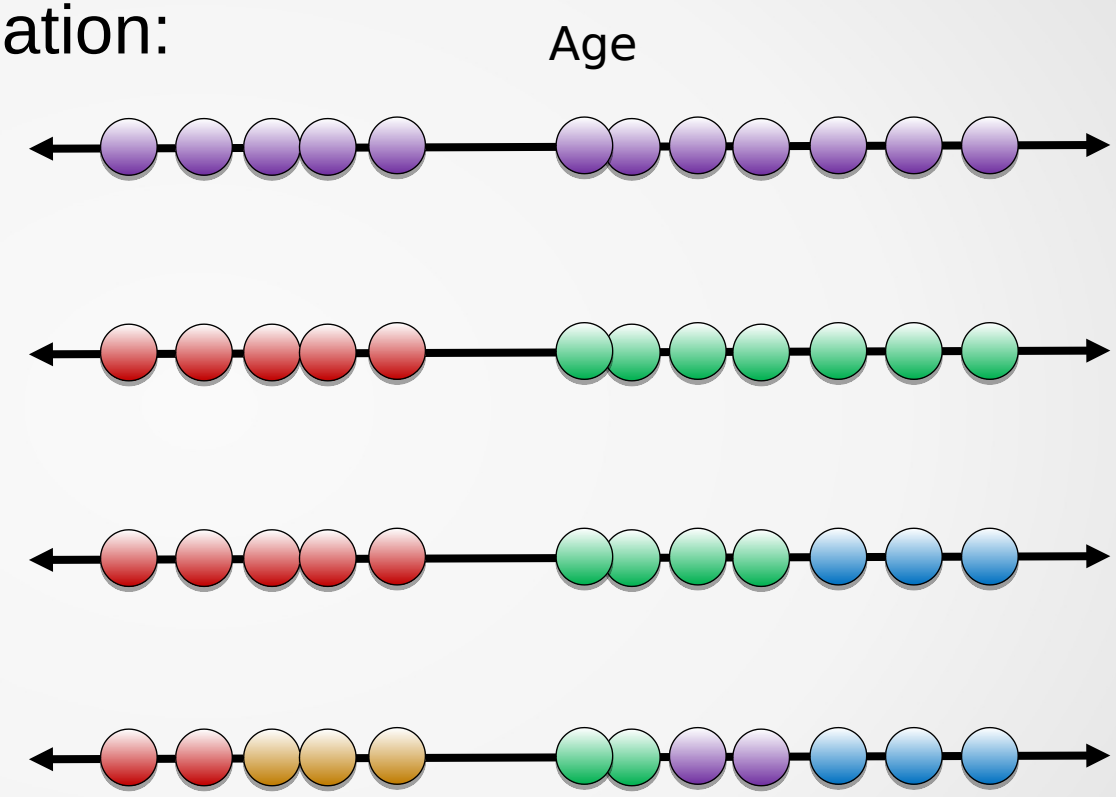
Dimensionality Reduction: Simplifying Structure



Introduction to Clustering

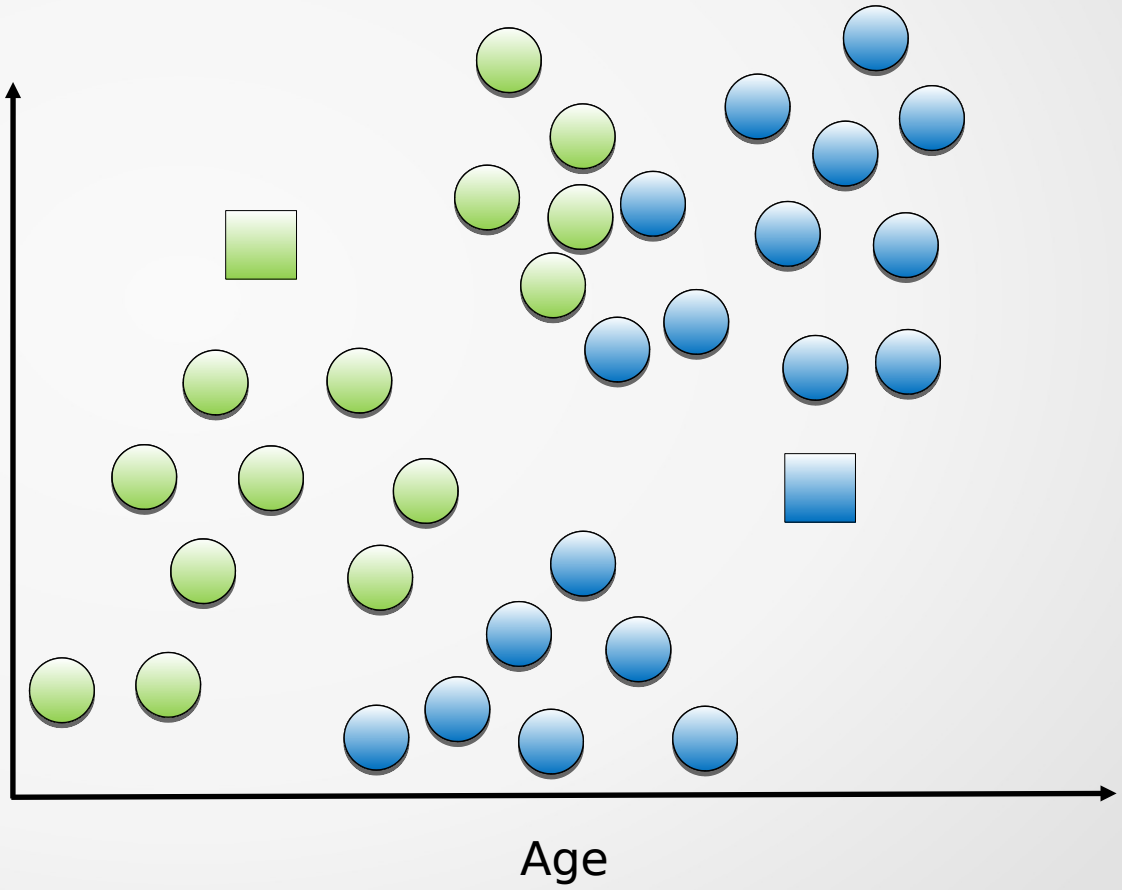
Users of a web application:

- One feature (age)
- Two clusters
- Three clusters
- Five clusters



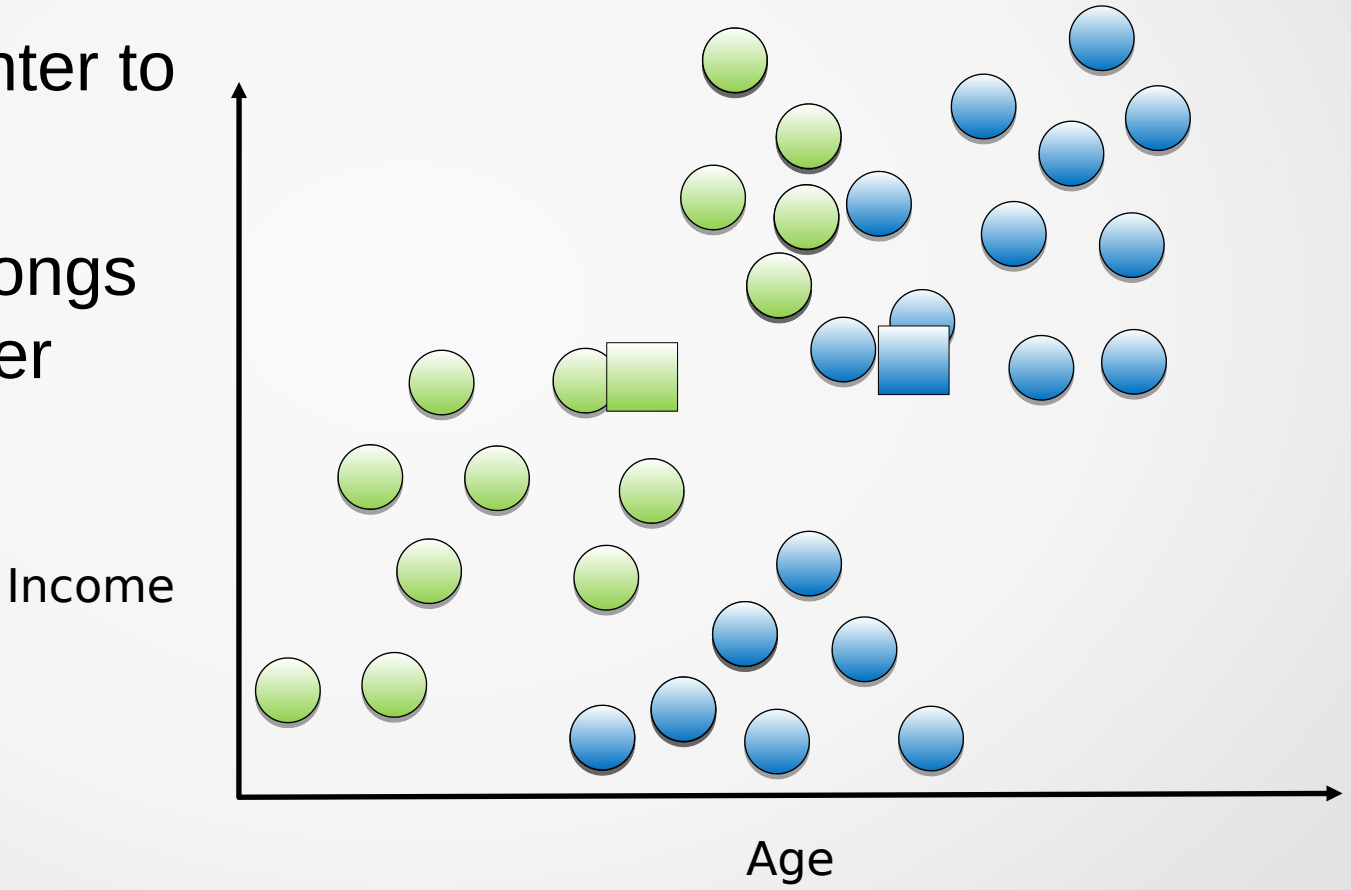
K-Means Algorithm

- K = 2 (find two clusters)
- Randomly assign cluster centers
- Each point belongs to closest center



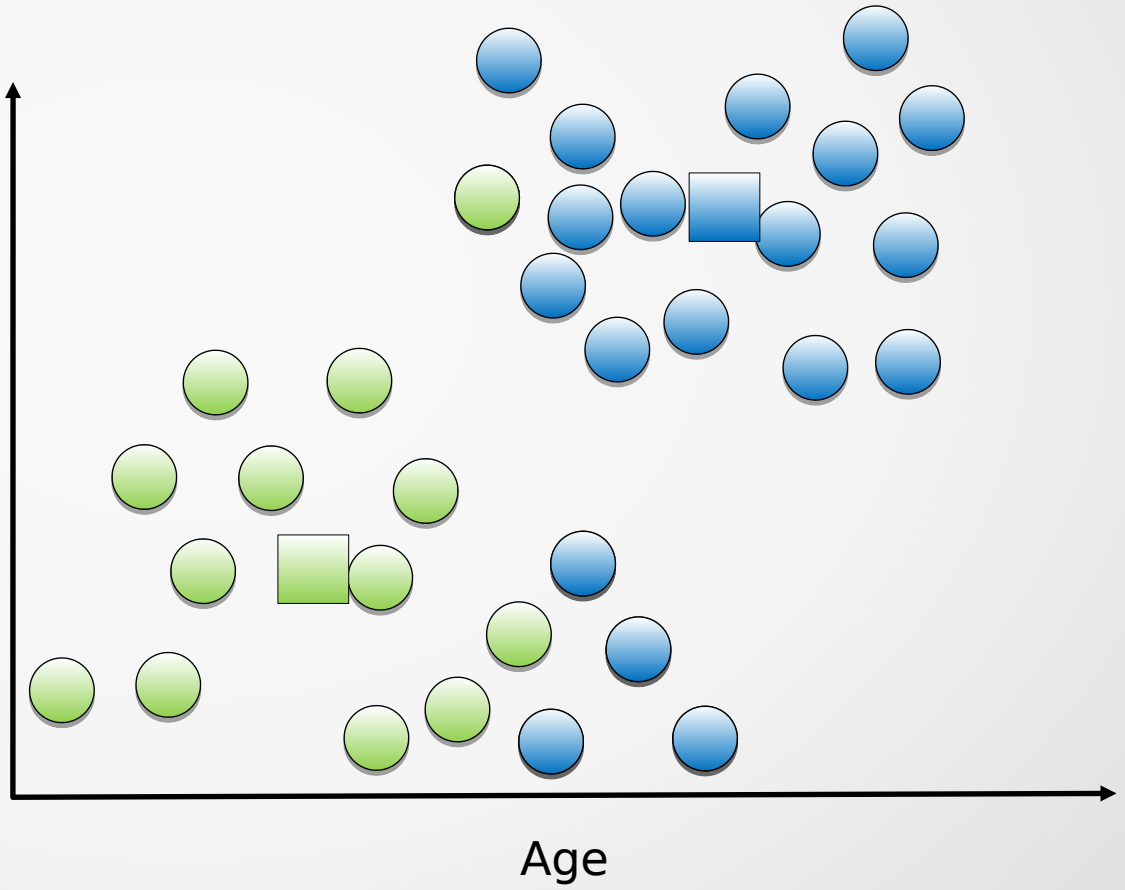
K-Means Algorithm

- K = 2
- Move each center to cluster's mean
- Each point belongs to closest center



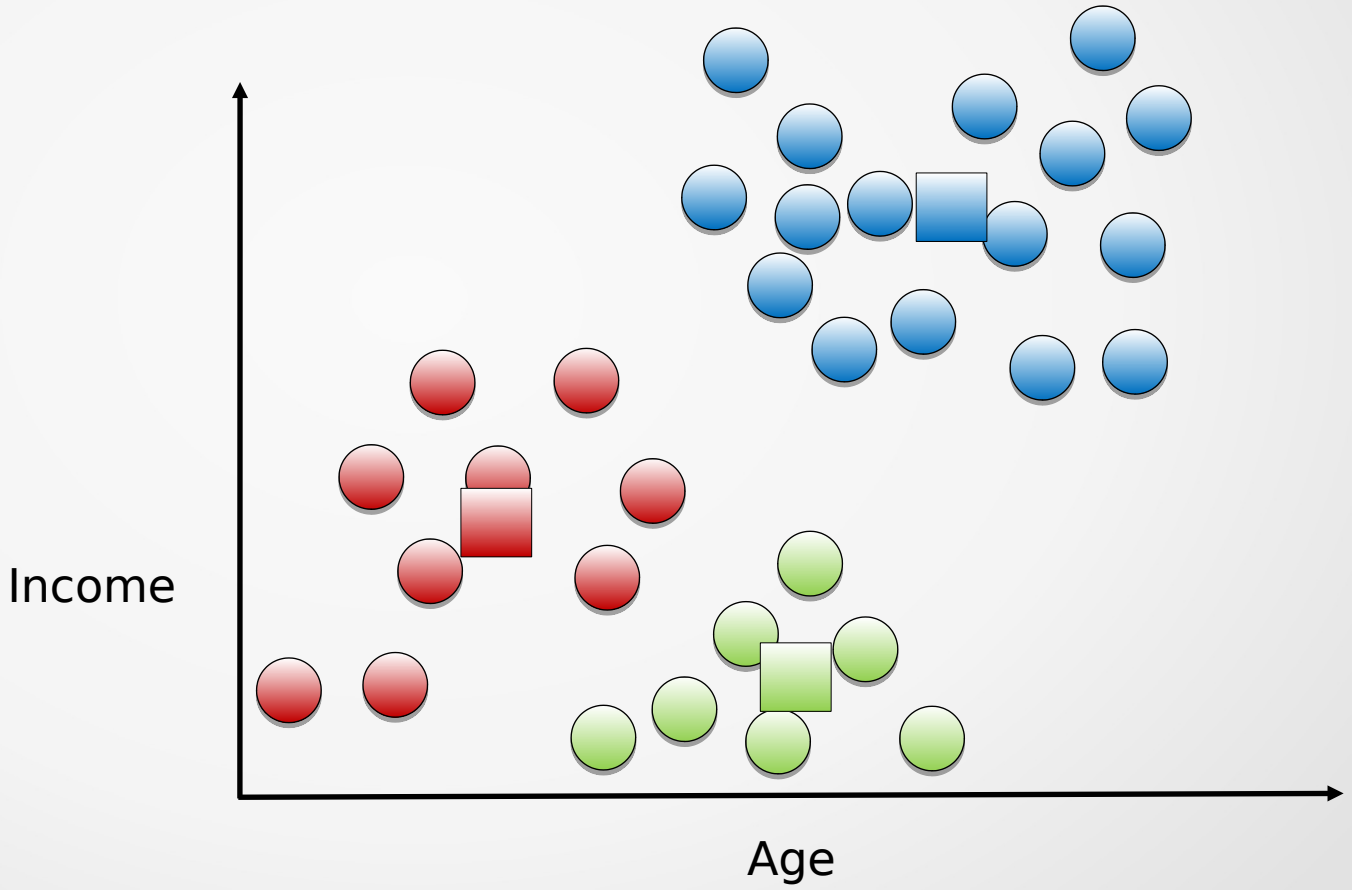
K-Means Algorithm

- K = 2
- Move each center to cluster's mean
- Each point belongs to closest center
- Once points stop changing, algorithm has converged



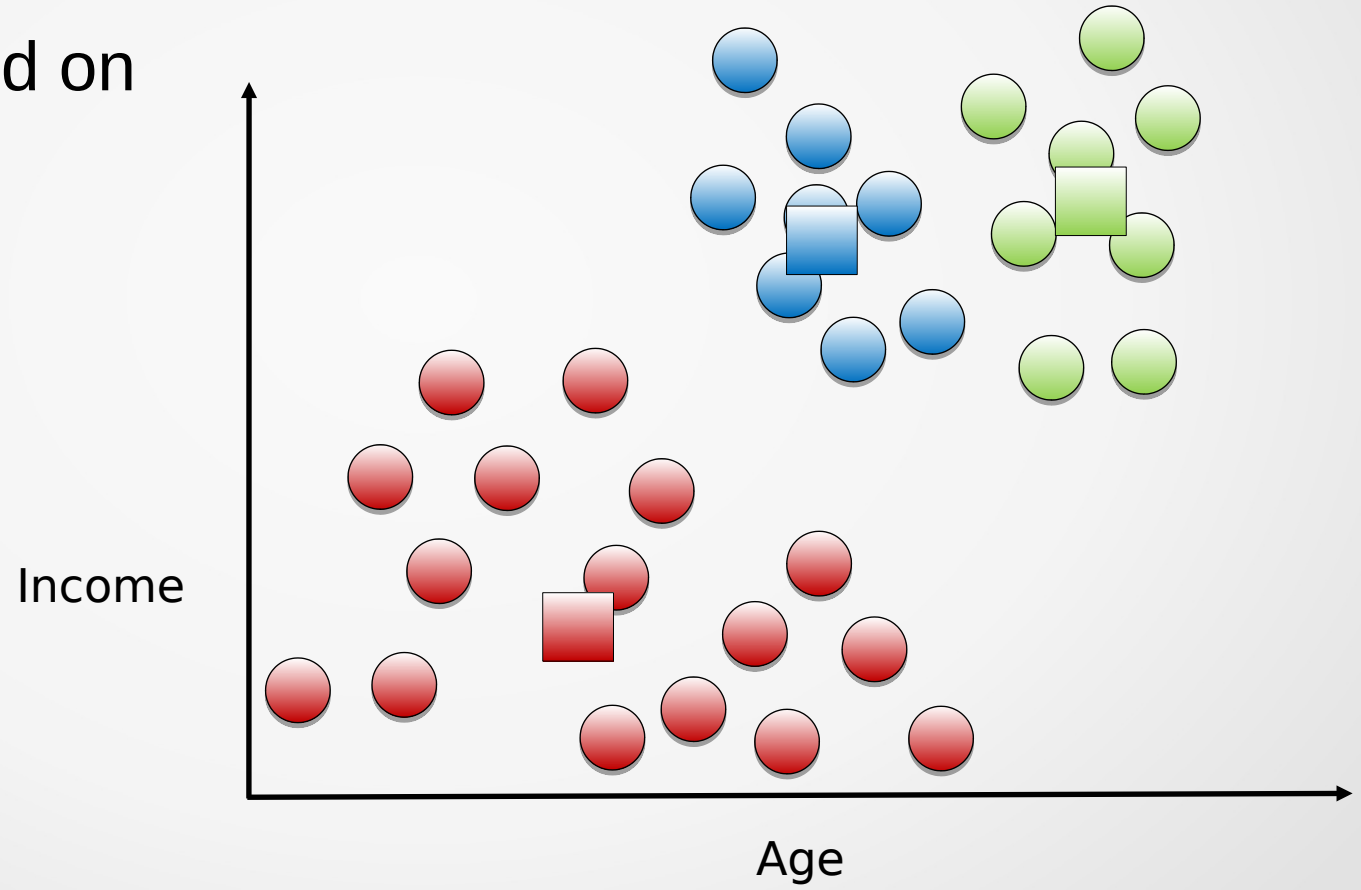
K-Means Algorithm

- K = 3



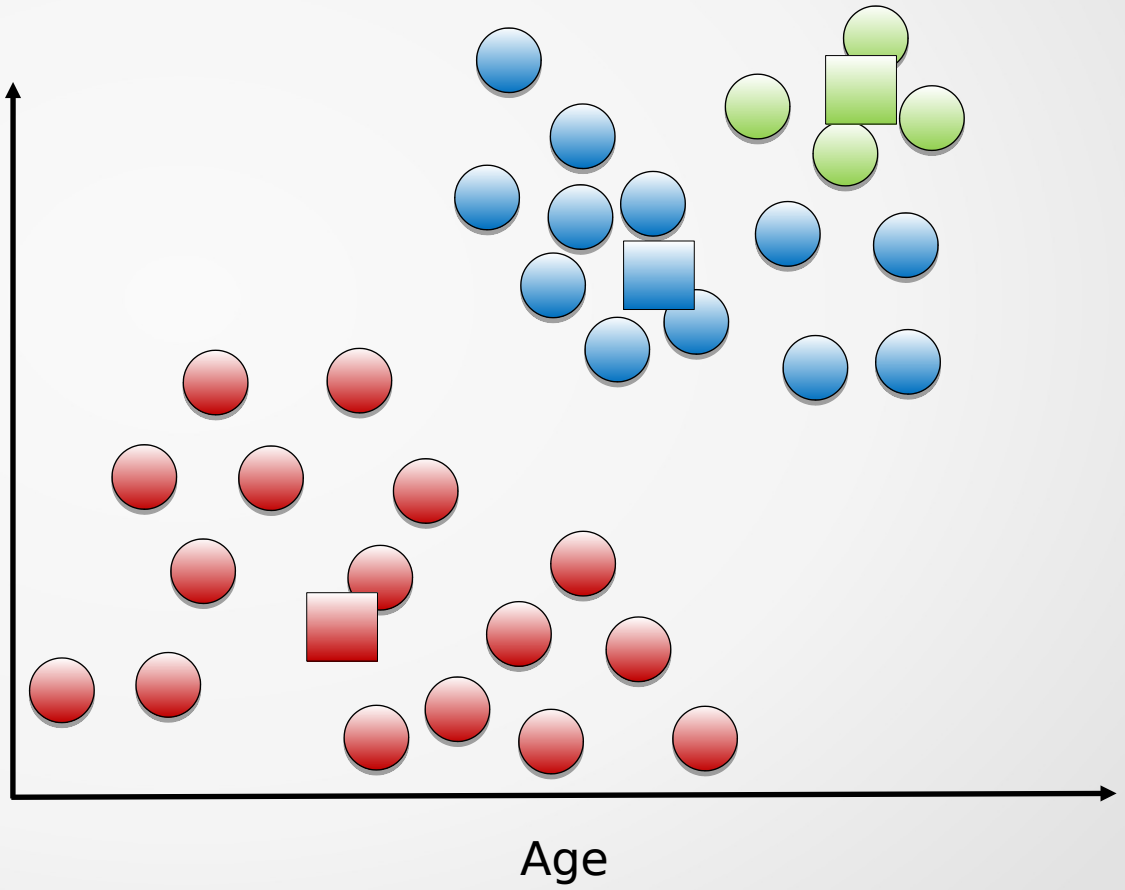
K-Means Algorithm

- K = 3
- Results depend on initial cluster assignment



K-Means Algorithm

- K = 3
- Results depend on initial cluster assignment
- Which model is the right one?



Selecting the Right (Best) Model

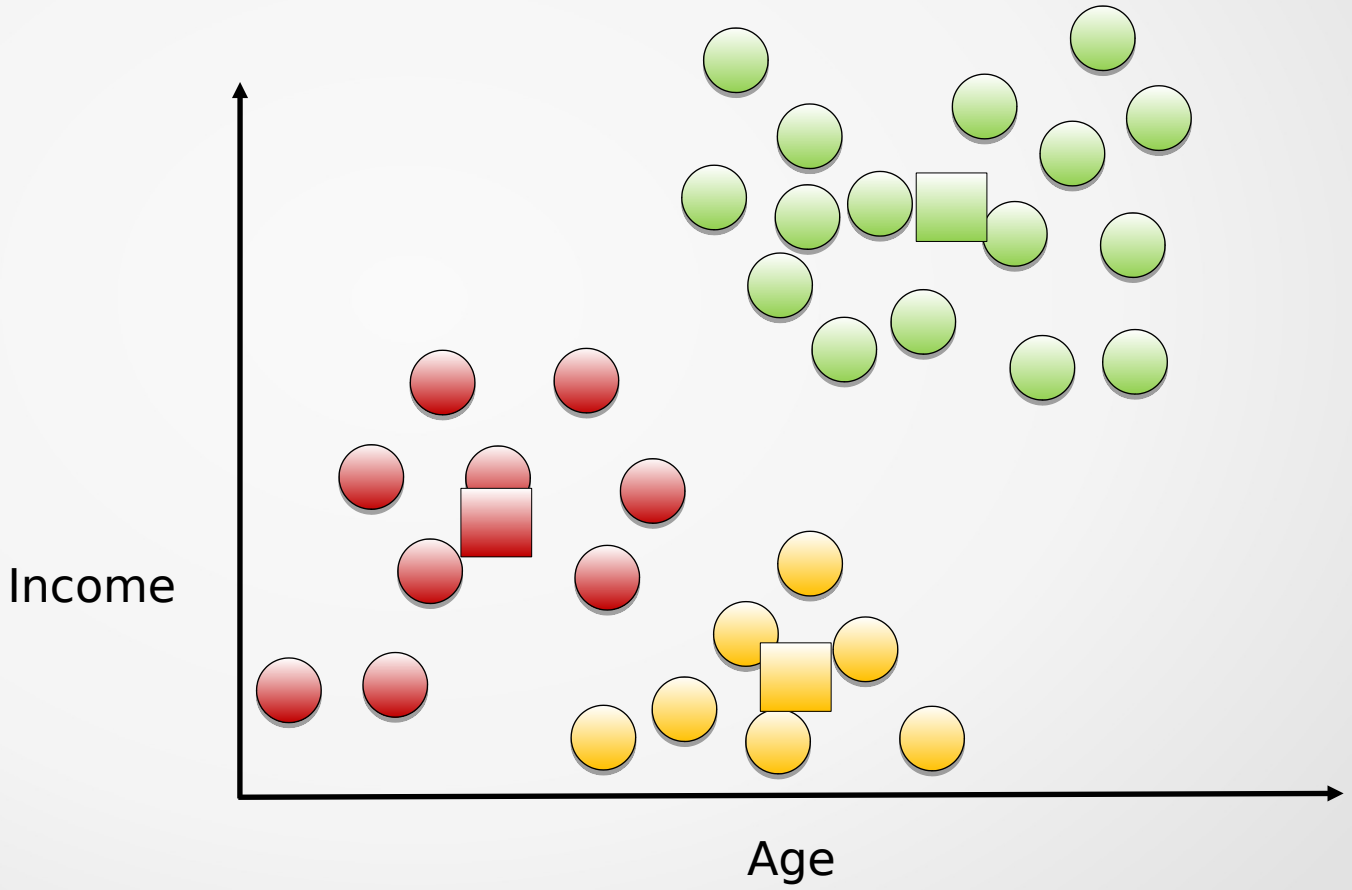
- Inertia: sum of squared distance from each point (x_i) to its cluster (C_k)

$$\sum_{i=1}^{n} (x_i - C_k)^2$$

- Smaller value corresponds to tighter clusters
- Other metrics can also be used
- Initiate algorithm multiple times, take model with the best score

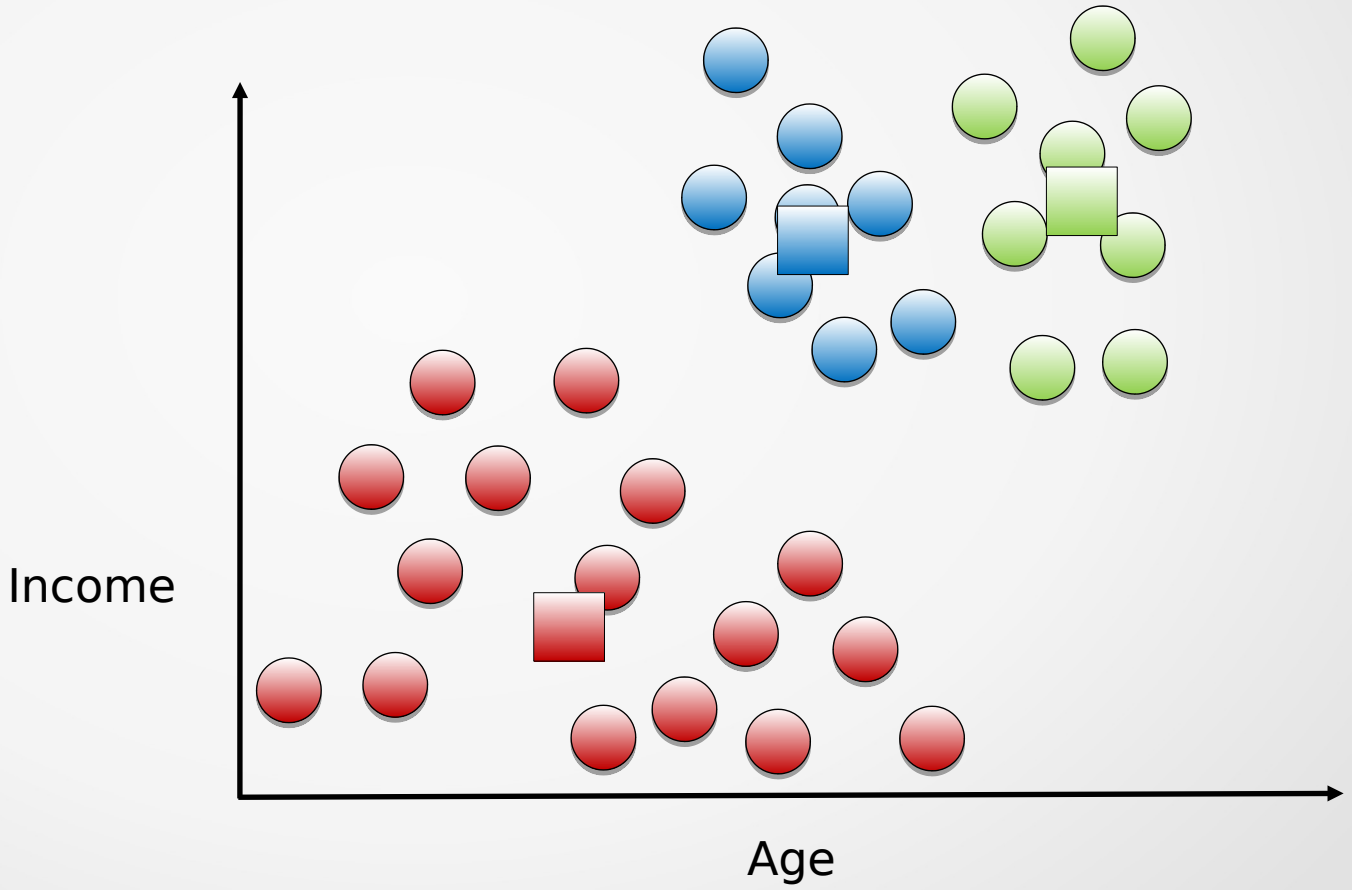
K-Means Algorithm – Selection

Inertia = 12.645



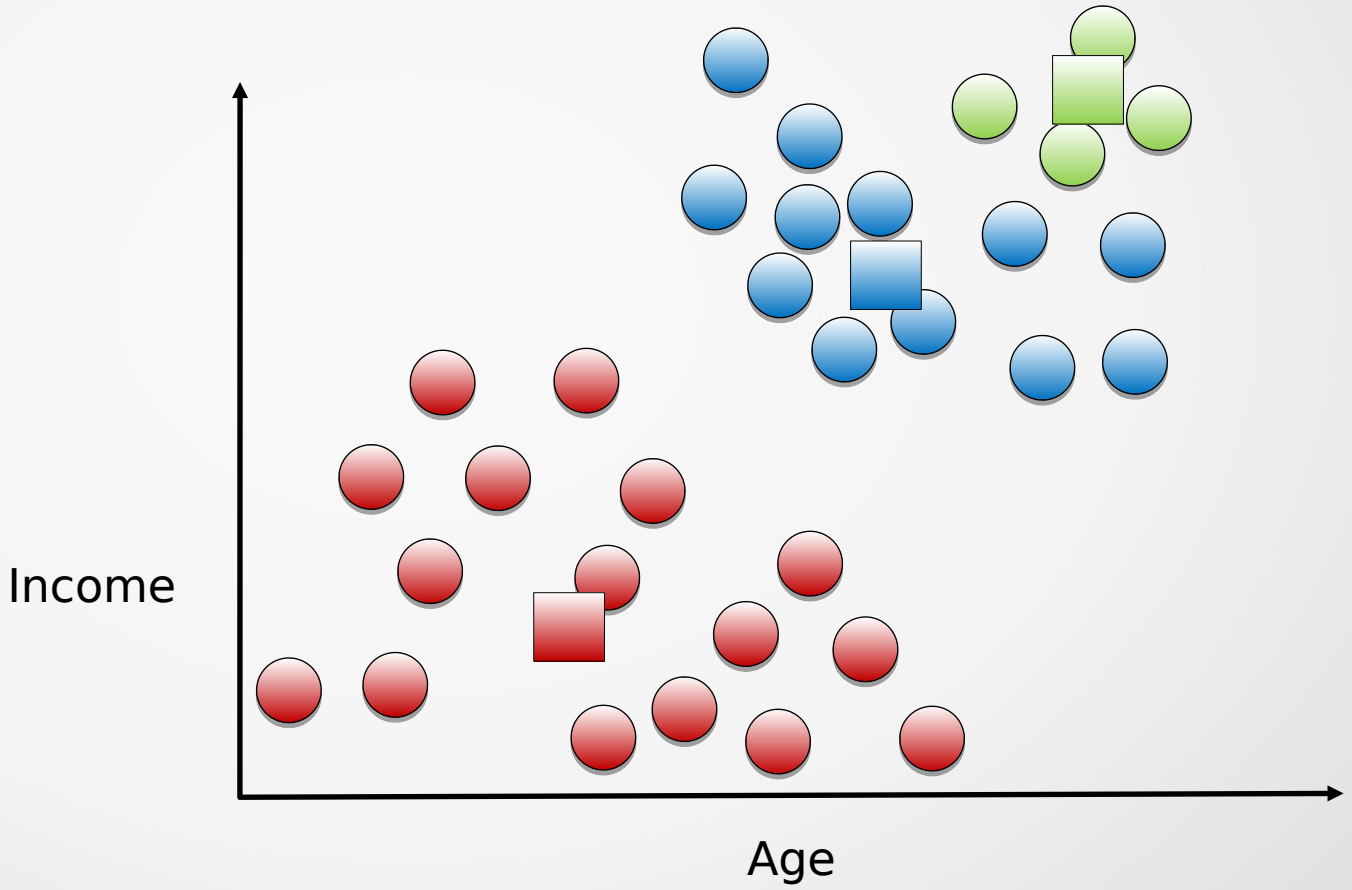
K-Means Algorithm – Selection

Inertia = 12.943



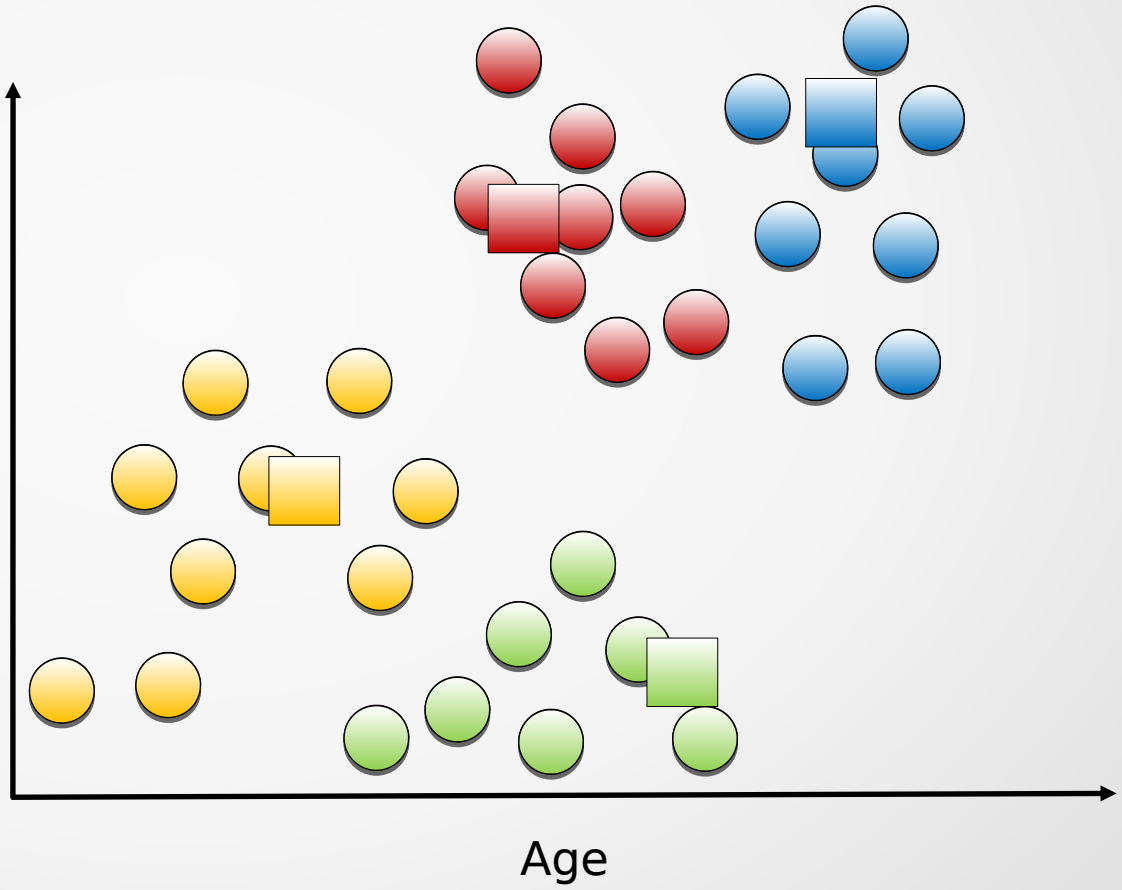
K-Means Algorithm – Selection

Inertia = 13.112



Smarter K-Means Cluster Initialization

- Pick one point at random as initial point
- Pick next points (repeatedly) with $1/distance^2$ probability
- Assign clusters

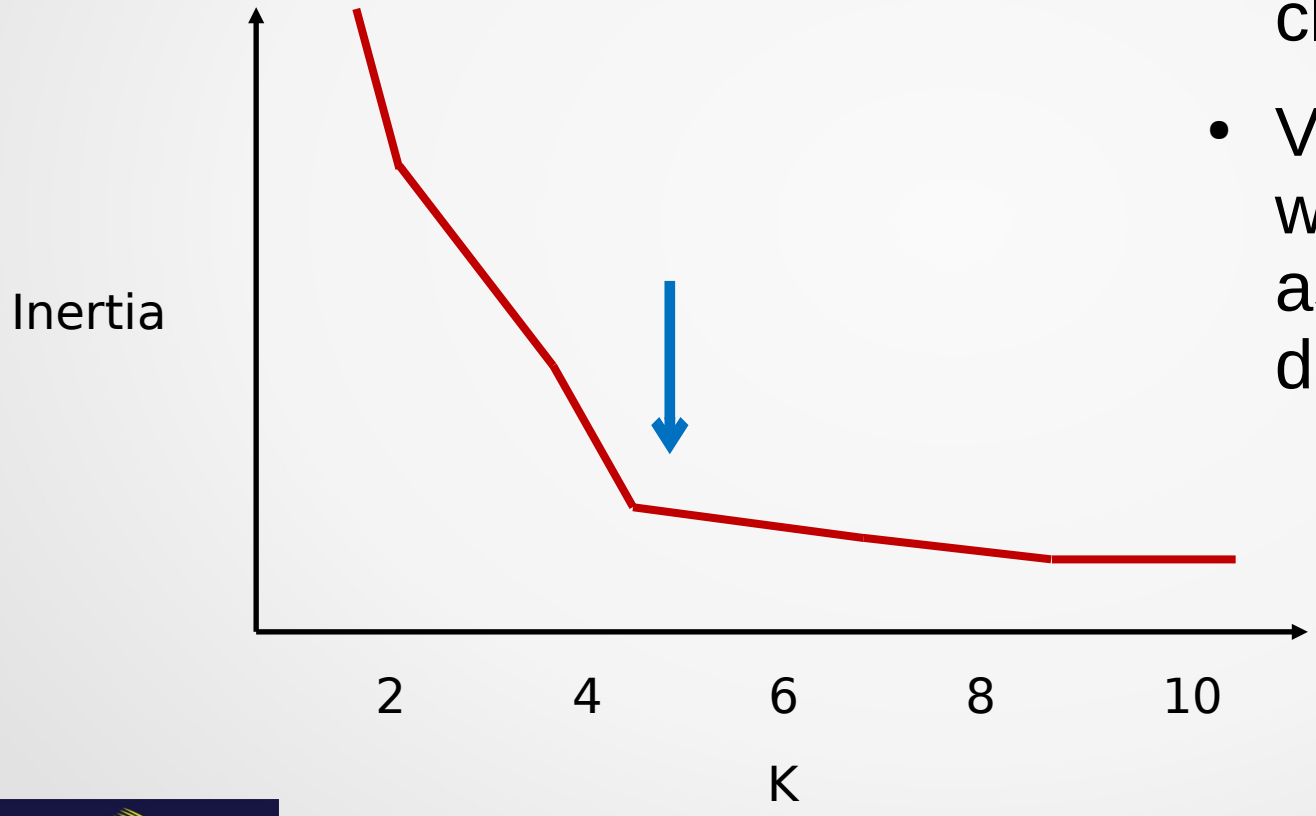


Choosing Right Number Of Clusters

Sometimes the question has a K

- Clustering similar jobs on 4 CPU cores (K=4)
- A clothing design in 10 different sizes to cover most people (K=10)
- A navigation interface for browsing scientific papers with 20 disciplines (K=20)

Choosing Right Number Of Clusters



- Inertia measures distance of point to cluster
- Value decreases with increasing K as long as cluster density increases

K-Means Syntax

- Import the class containing the clustering method

```
from sklearn.cluster import KMeans
```

- Create an instance of the class

```
kmeans = Kmeans(n_clusters=3,  init='kmean++')
```

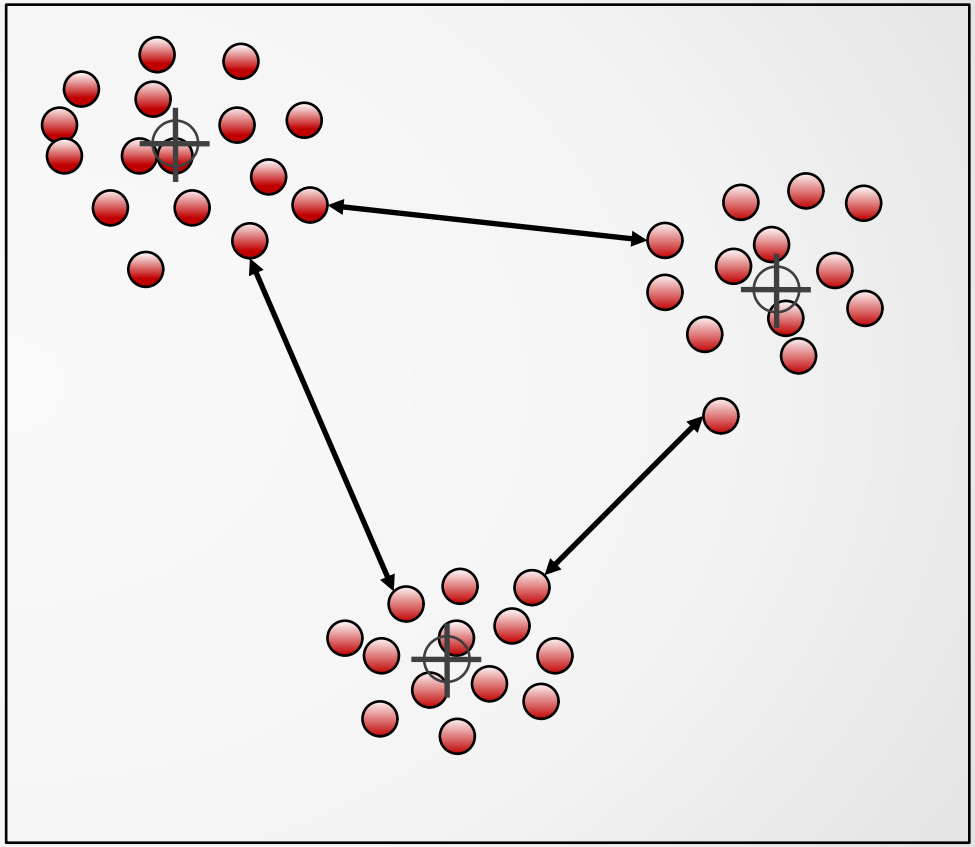
- Fit the instance on the data and then predict clusters for new data

```
kmeans = kmeans.fit(x1)
y_predict = kmeans.predict(x2)
```

- Can also be used in batch mode with MiniBatchKMeans.

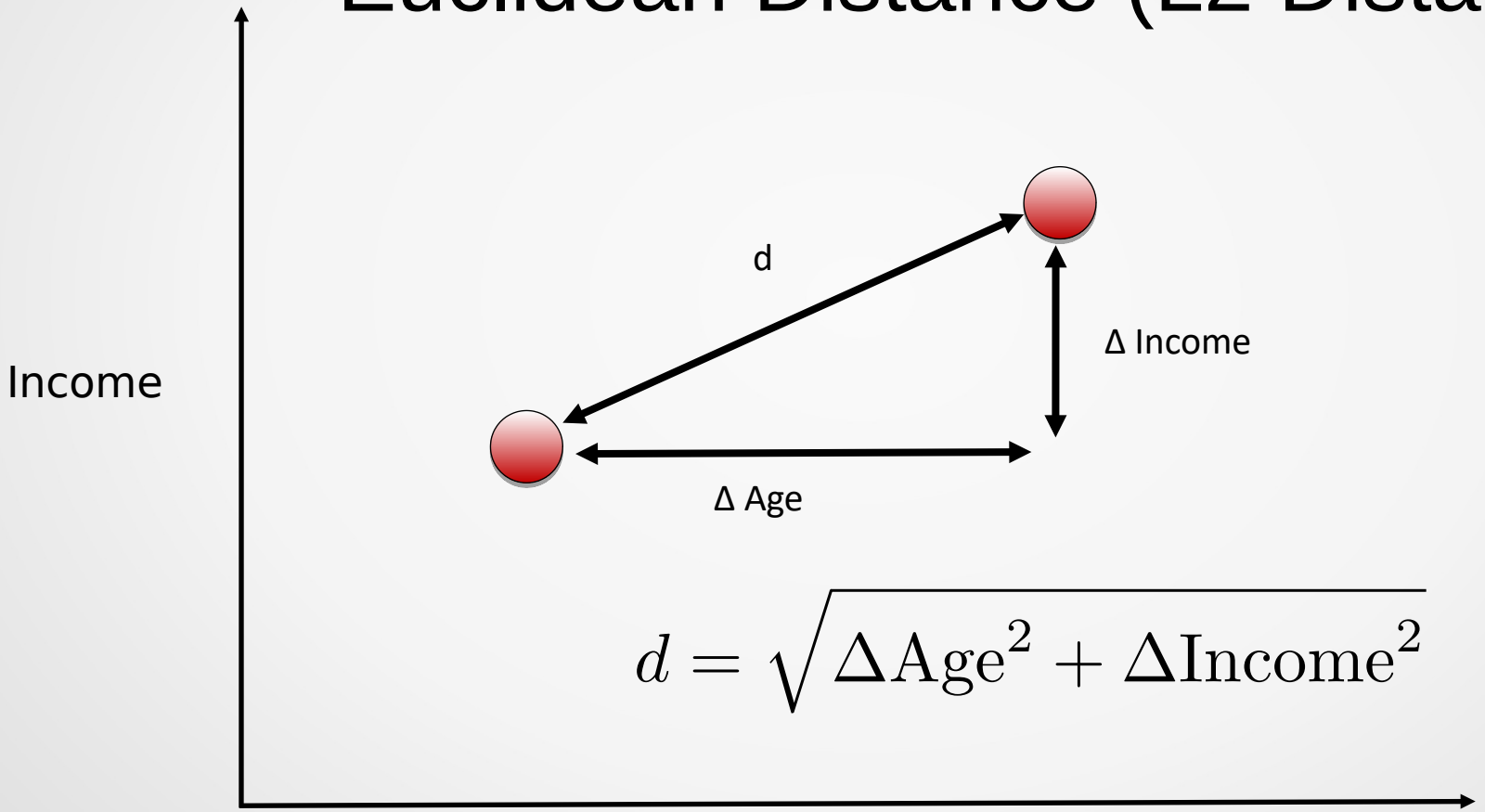
Distance Metric

- Choice of distance metric is extremely important to clustering success
- Each metric has strengths and most appropriate use-cases
- Sometimes choice of distance metric is also based on empirical evaluation



Distance Metric

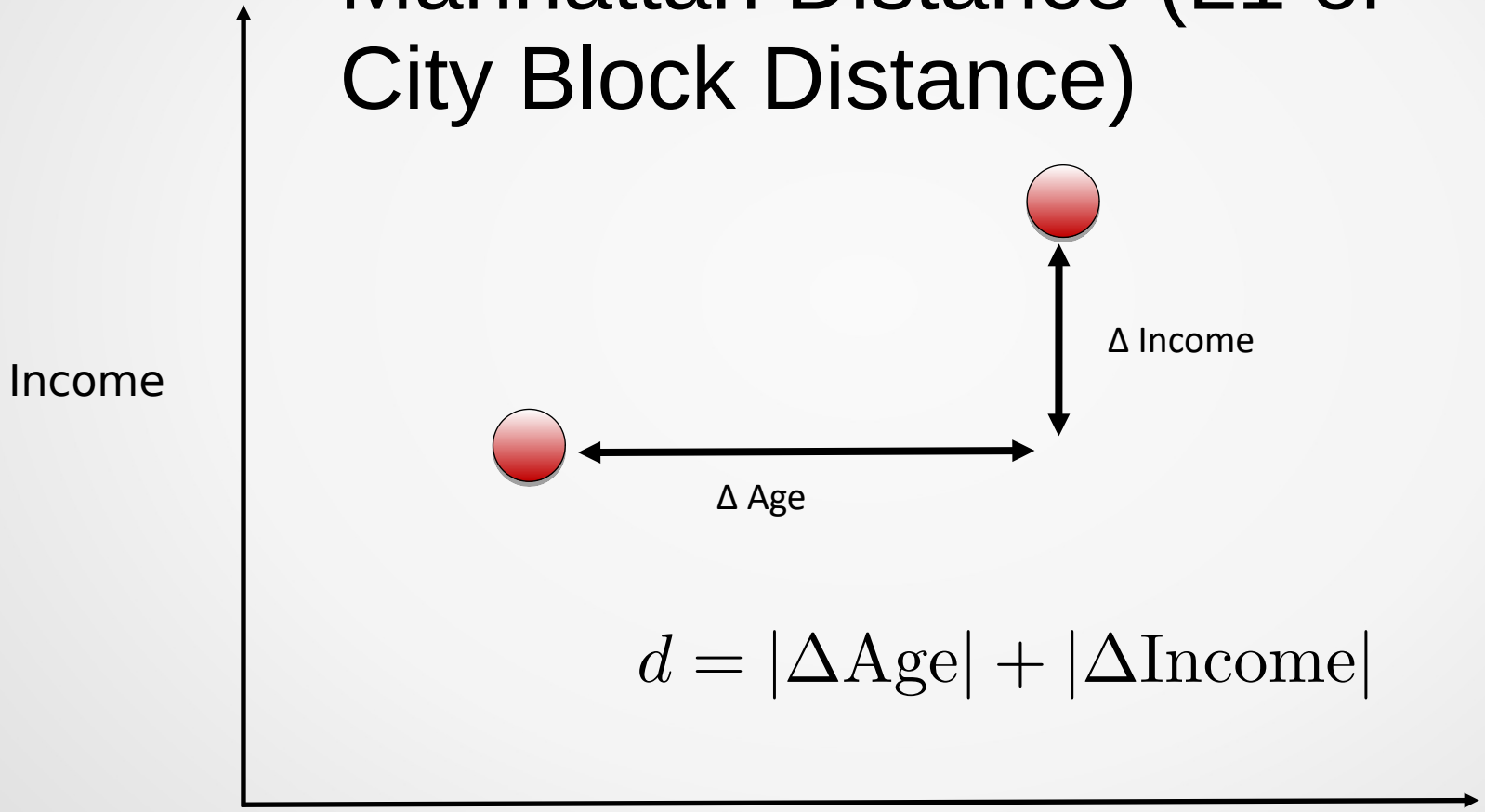
Euclidean Distance (L2 Distance)



$$d = \sqrt{\Delta\text{Age}^2 + \Delta\text{Income}^2}$$

Distance Metric

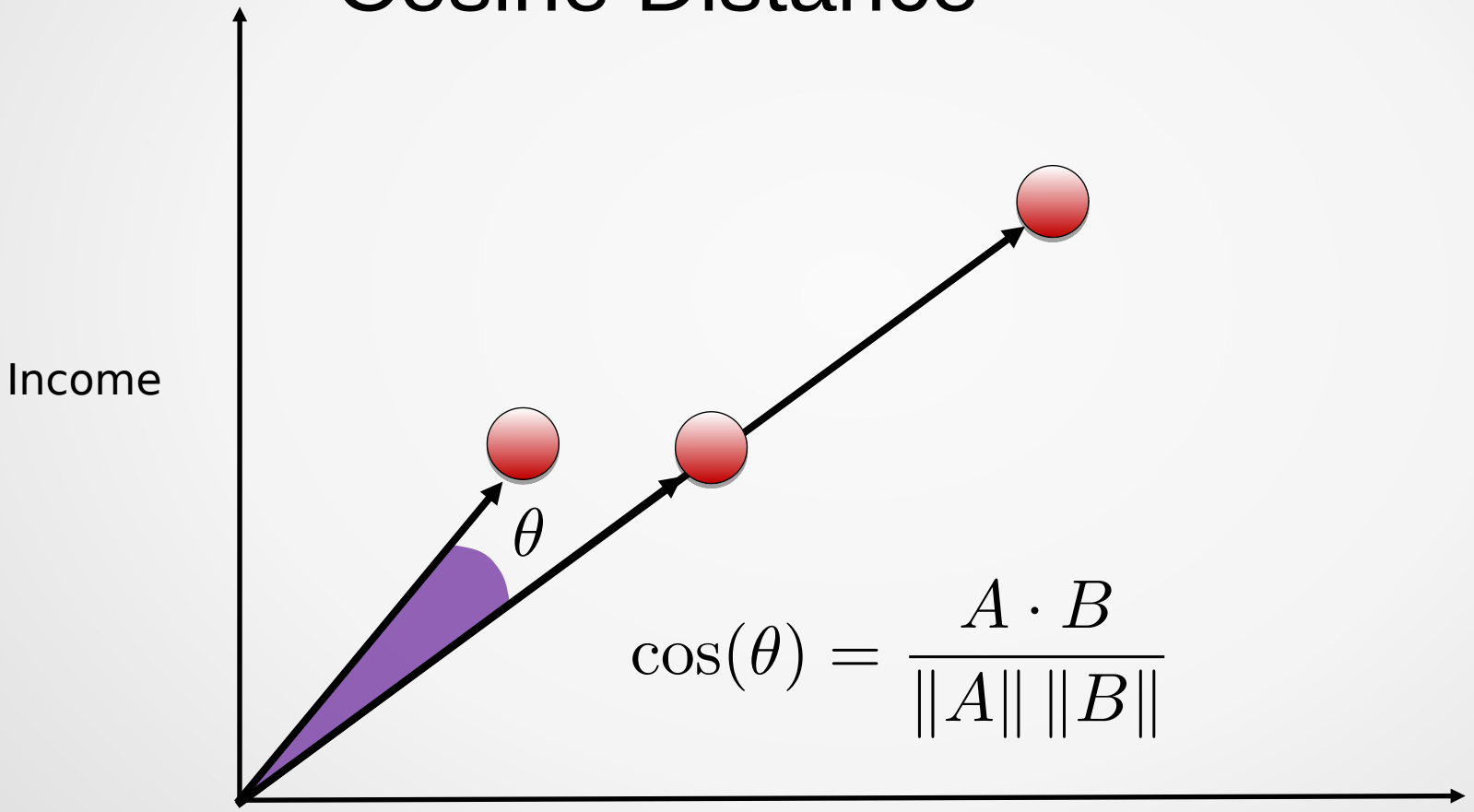
Manhattan Distance (L1 or City Block Distance)



$$d = |\Delta \text{Age}| + |\Delta \text{Income}|$$

Distance Metric

Cosine Distance



$$\cos(\theta) = \frac{A \cdot B}{\|A\| \, \|B\|}$$

Distance Metric

Euclidean vs Cosine Distance

- Euclidean is useful for coordinate based measurements
- Cosine is better for data such as text where location of occurrence is less important
- Euclidean distance is more sensitive to curse of dimensionality (we'll cover this later)

Distance Metric

Jaccard Distance (applies to sets like word occurrence)

- **Sentence A**: “I like chocolate ice cream.”
  - set A = { I , like , chocolate , ice , cream }
- **Sentence B**: “Do I want chocolate cream or vanilla cream?”
  - set B = { Do , I , want , chocolate , cream , or , vanilla }

$$1 - \frac{A \cap B}{A \cup B} = 1 - \frac{\text{len}(\text{shared})}{\text{len}(\text{unique})} = 1 - \frac{3}{9}$$

Distance Metrics Syntax

- Import the general pairwise distance function

```
from sklearn.metrics import pairwise_distances
```

- Calculate the distances

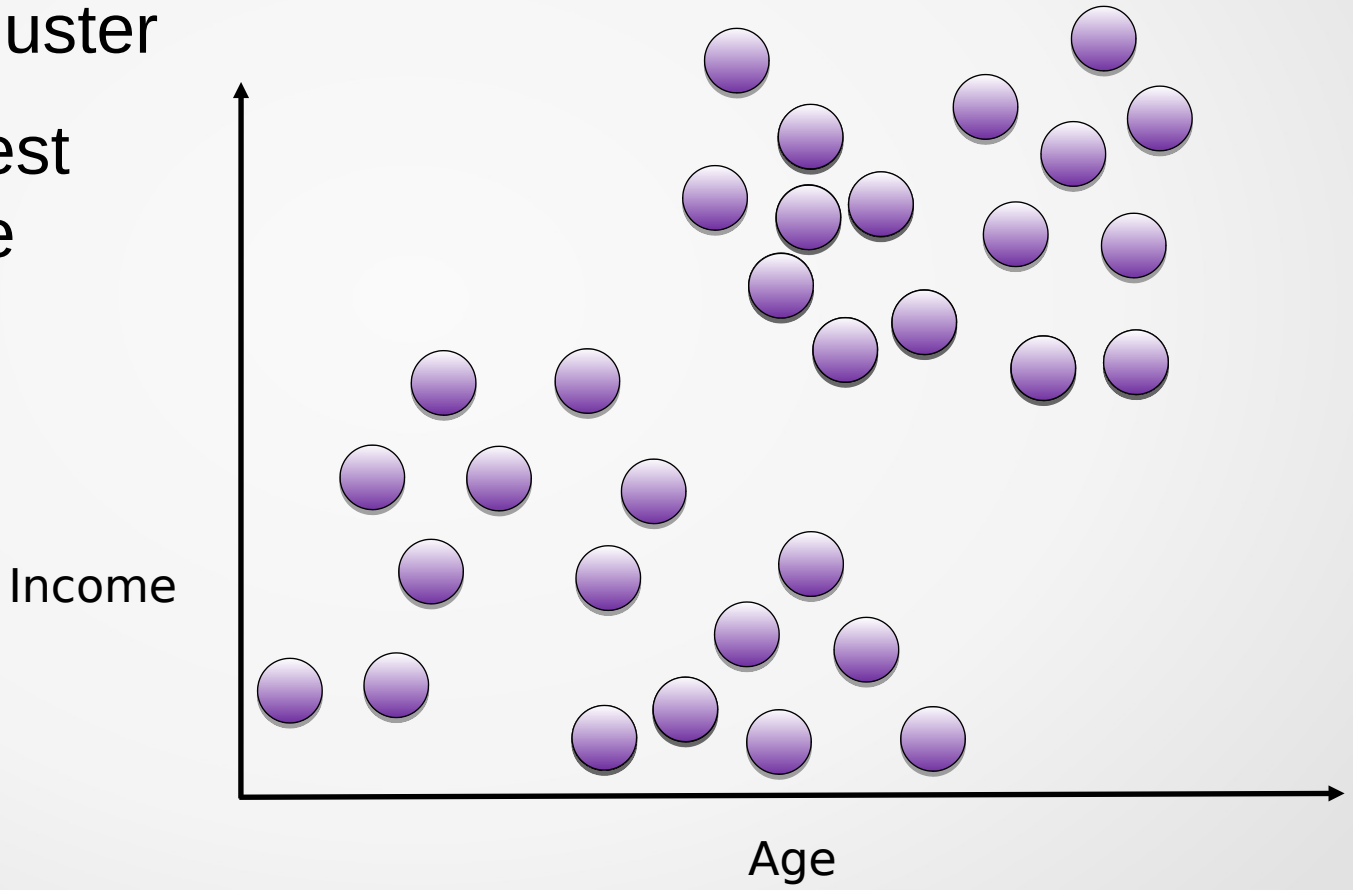
```
dist = pairwise_distances(X, Y, metric='euclidean')
```

- Other distance metric choices are: cosine, manhattan, jaccard, etc.
- Distance metric methods can also be imported specifically e.g.

```
from sklearn.metrics import euclidean_distances
```

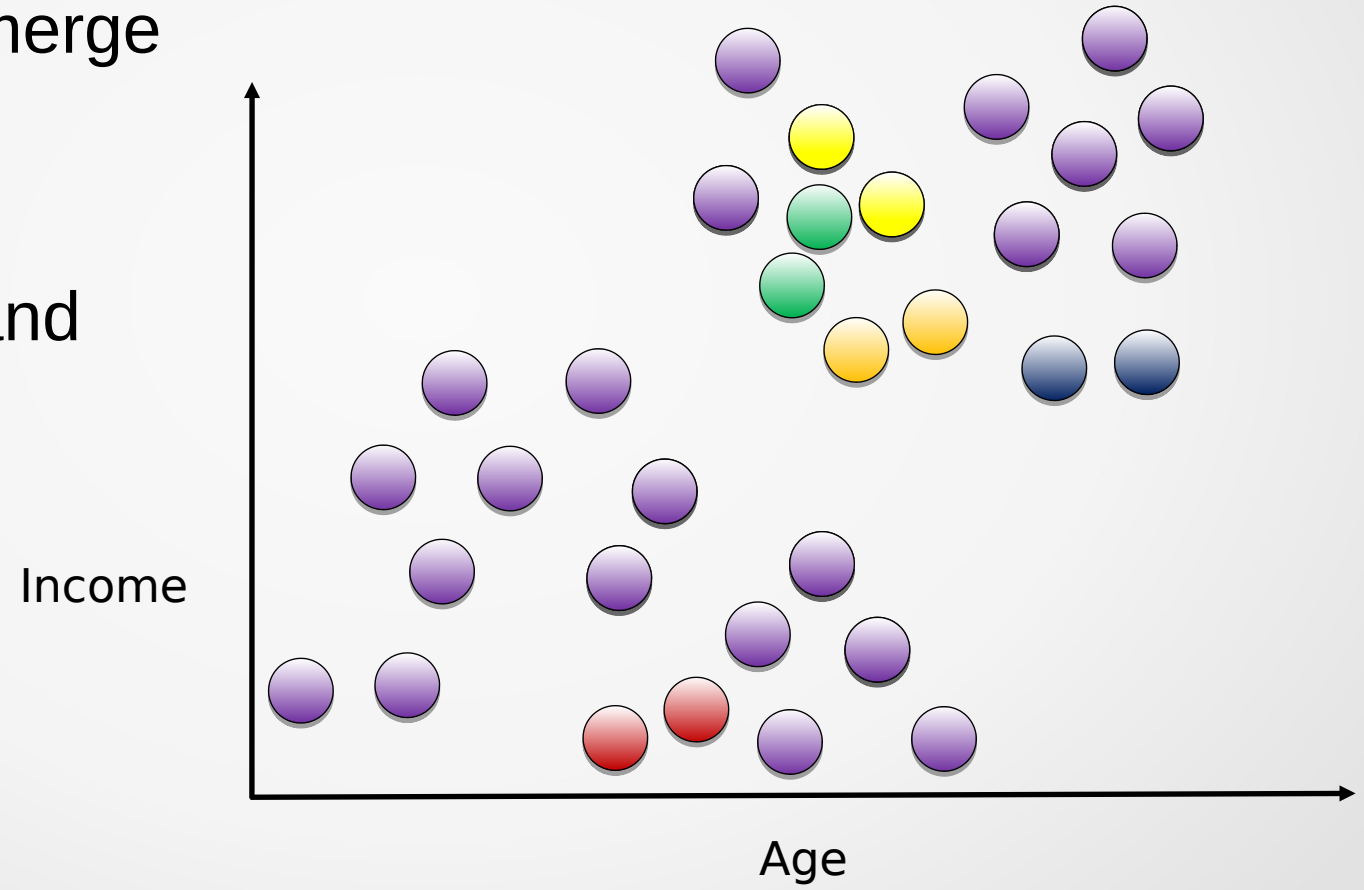
Heirarchical Agglomerative Clustering

- Find closest pair, merge into a cluster
- Find next closest pair and merge
- Keep merging closest pairs



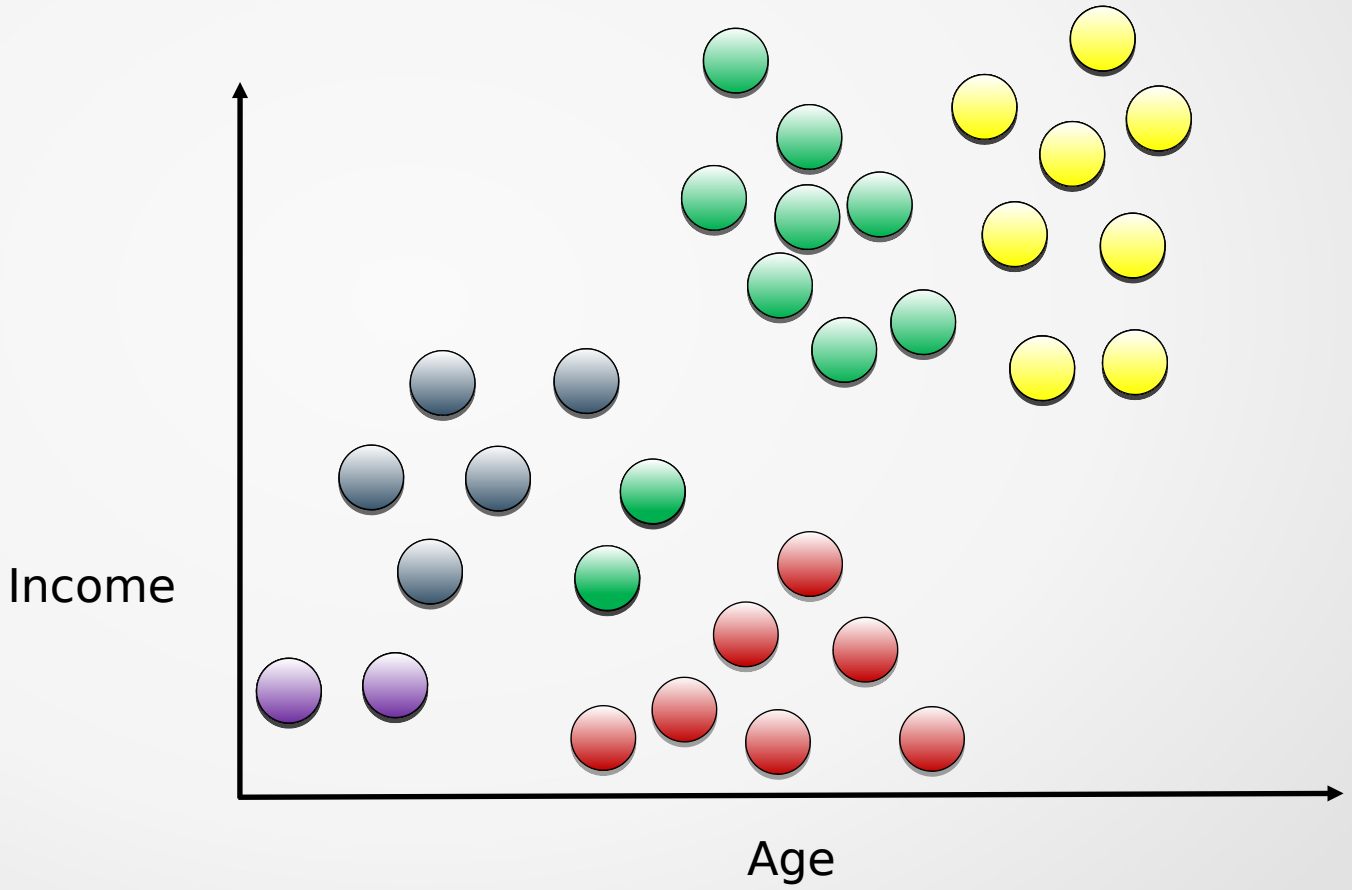
Heirarchical Agglomerative Clustering

- If the closest pair is two clusters, merge them
- Keep merging closest pairs and clusters



Heirarchical Agglomerative Clustering

- 6 clusters
- 5 clusters
- 4 clusters
- 3 clusters
- 2 clusters
- 1 cluster



Heirarchical Agglomerative Clustering

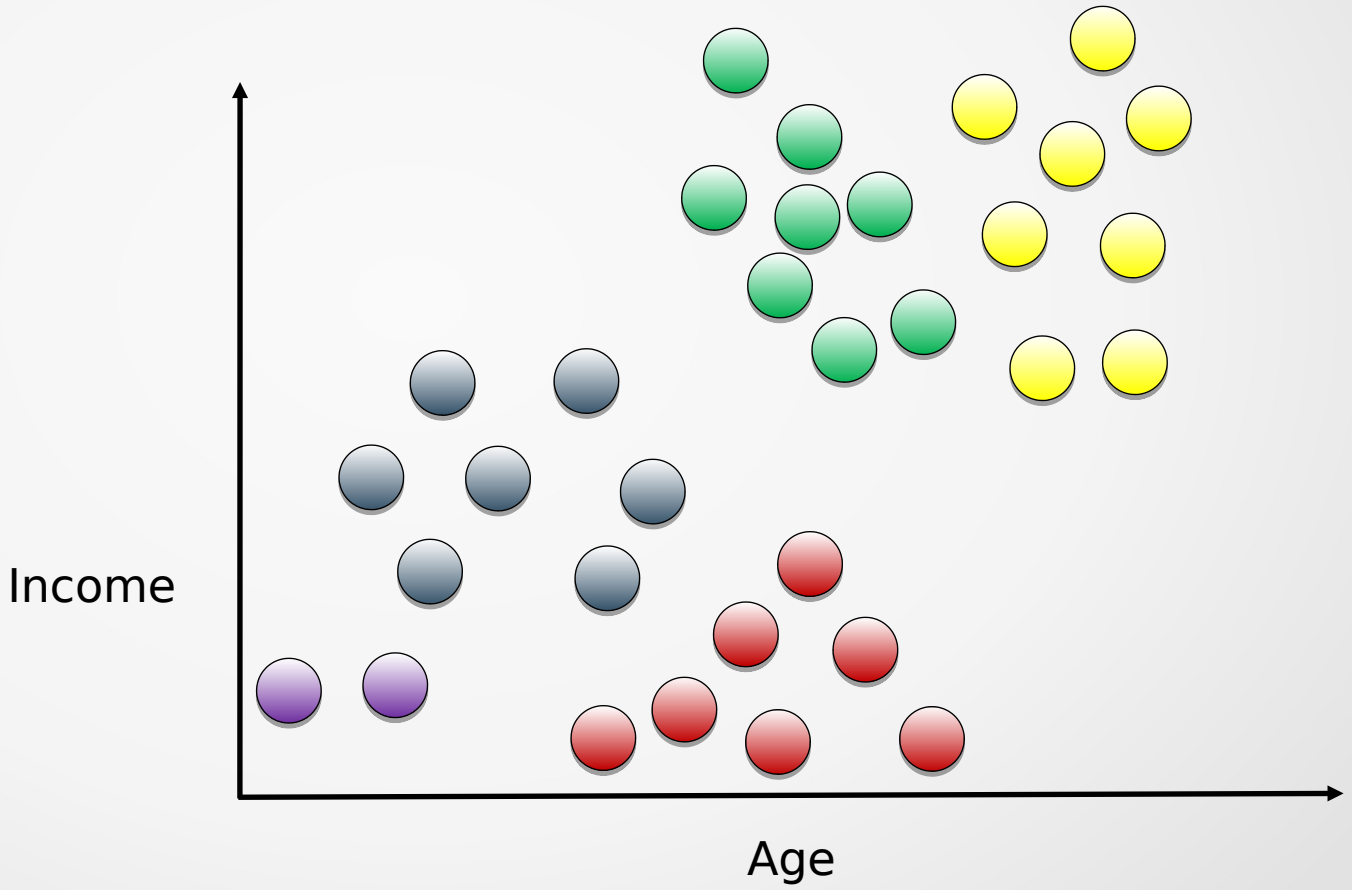
Agglomerative Clustering stopping conditions

Condition 1 — the correct number of clusters is reached

Condition 2 — minimum average cluster distance reaches a set value

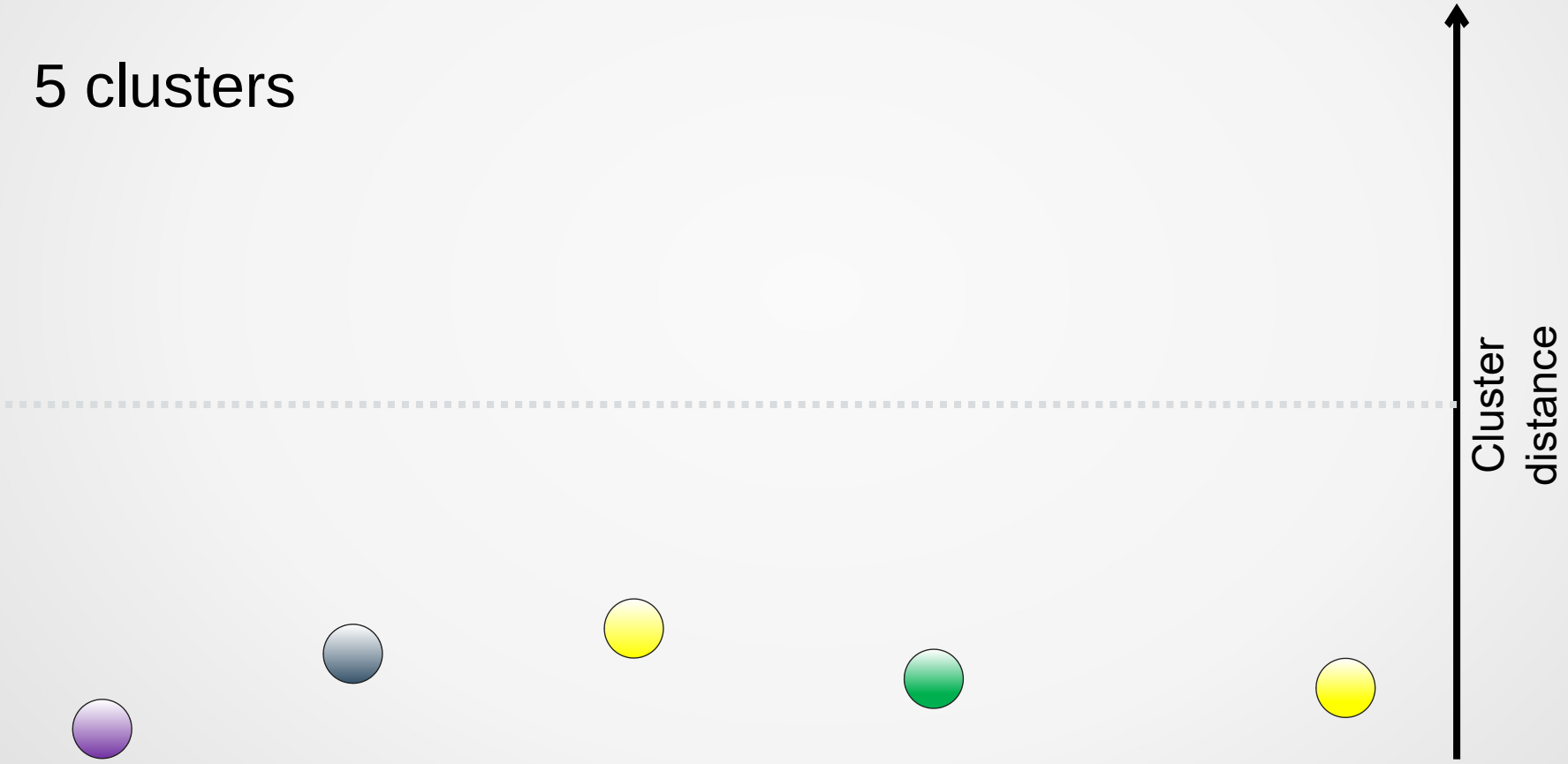
Heirarchical Agglomerative Clustering

5 clusters



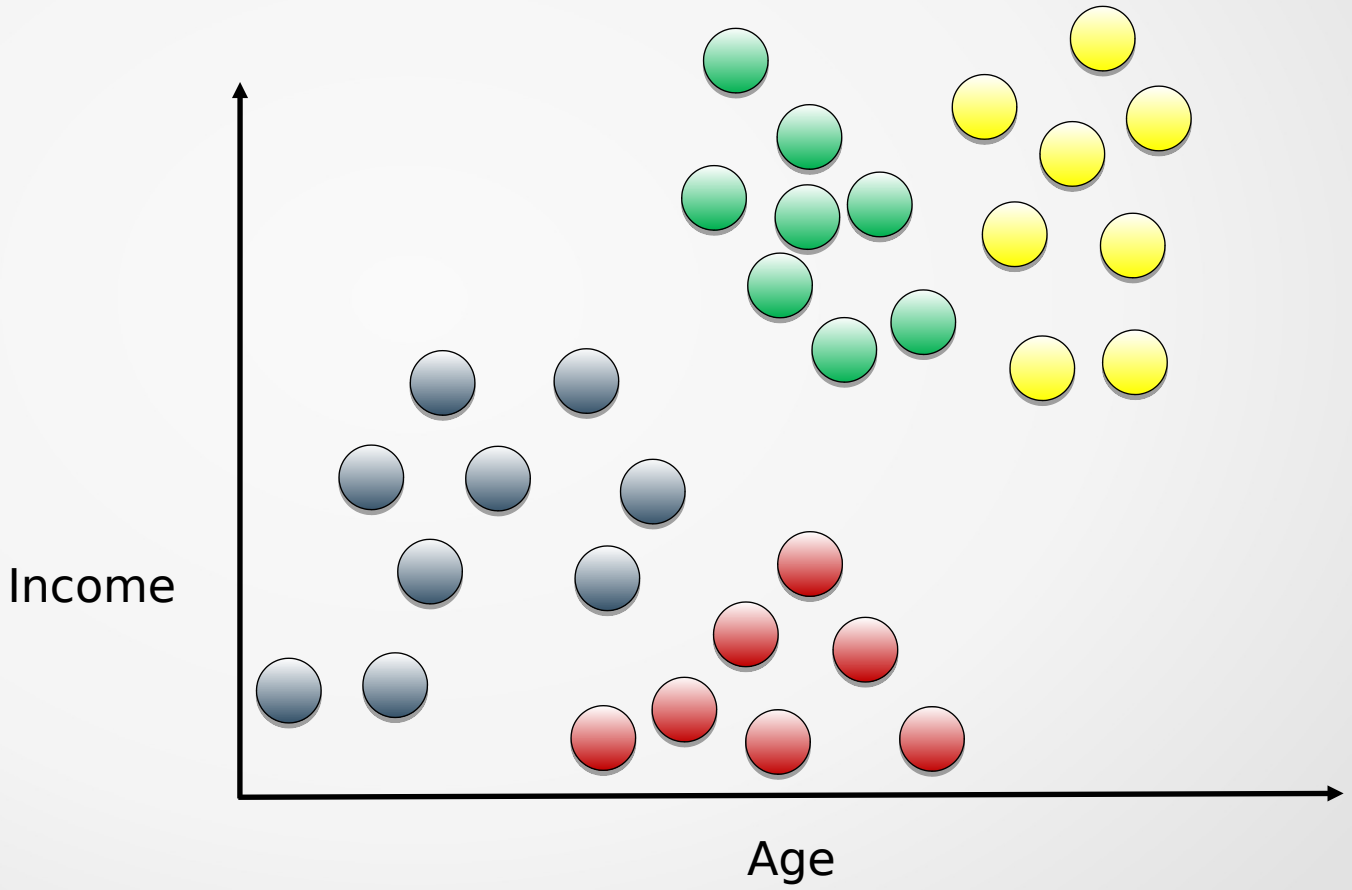
Heirarchical Agglomerative Clustering

5 clusters



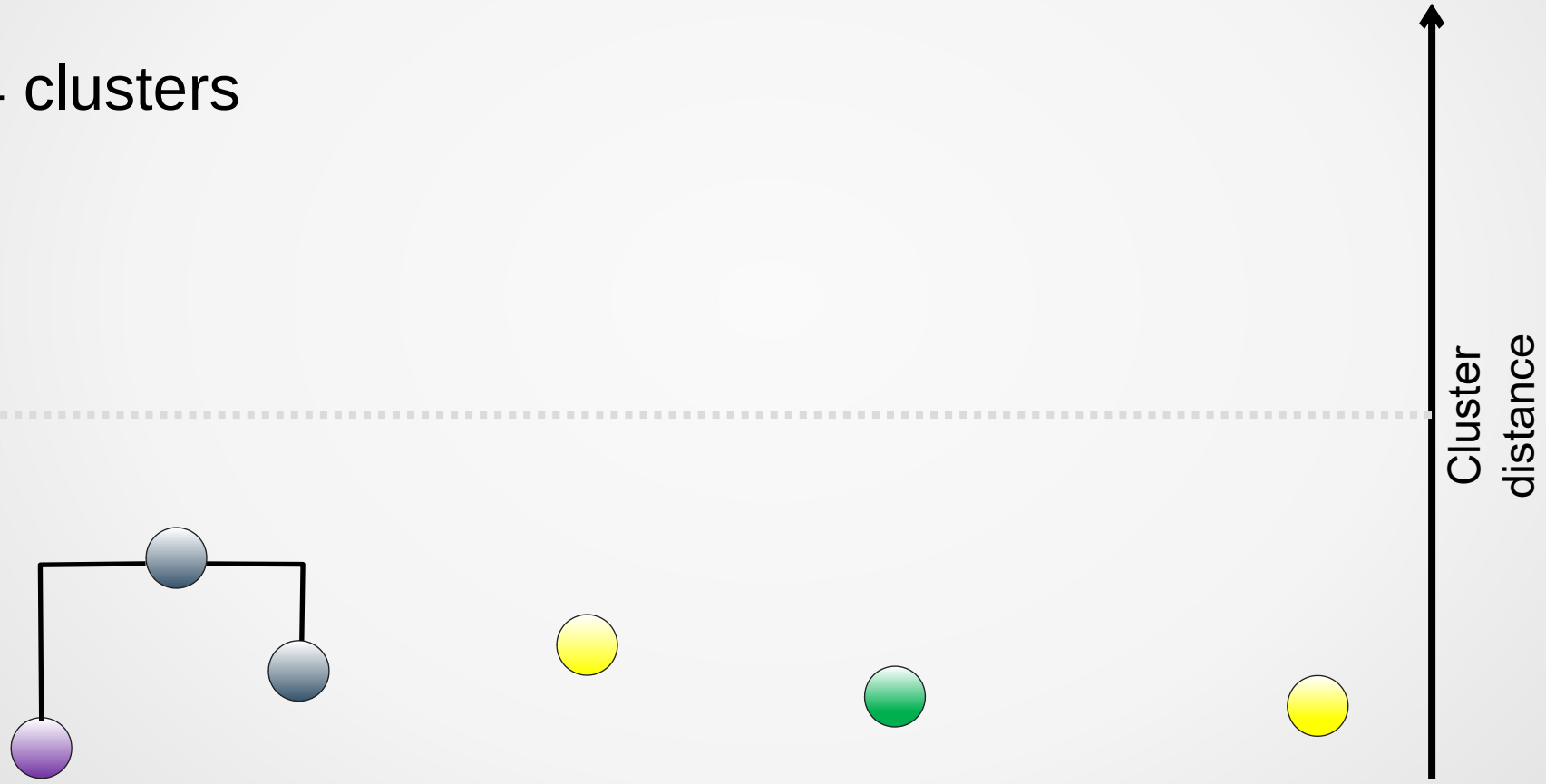
Heirarchical Agglomerative Clustering

4 clusters



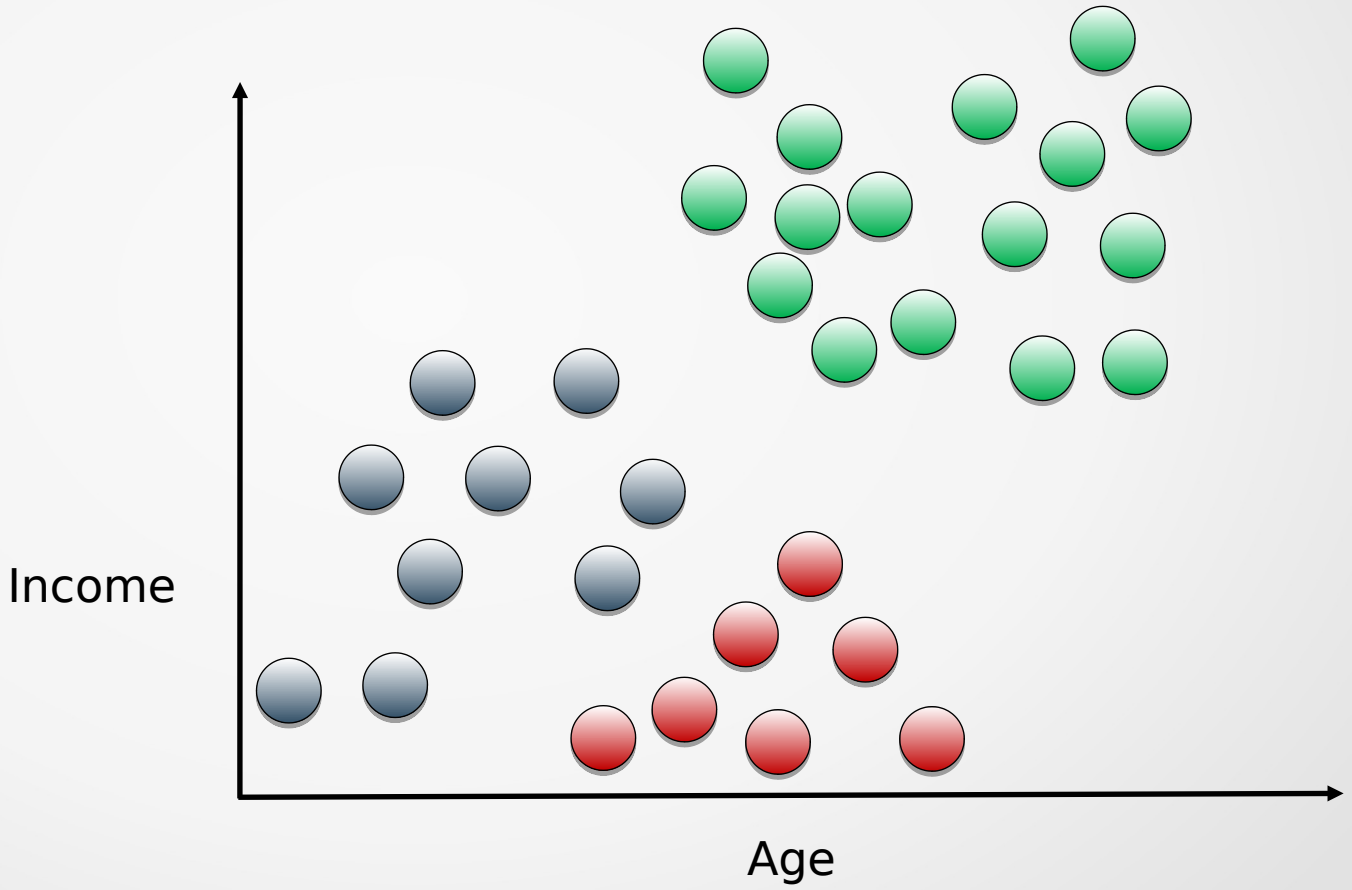
Heirarchical Agglomerative Clustering

4 clusters



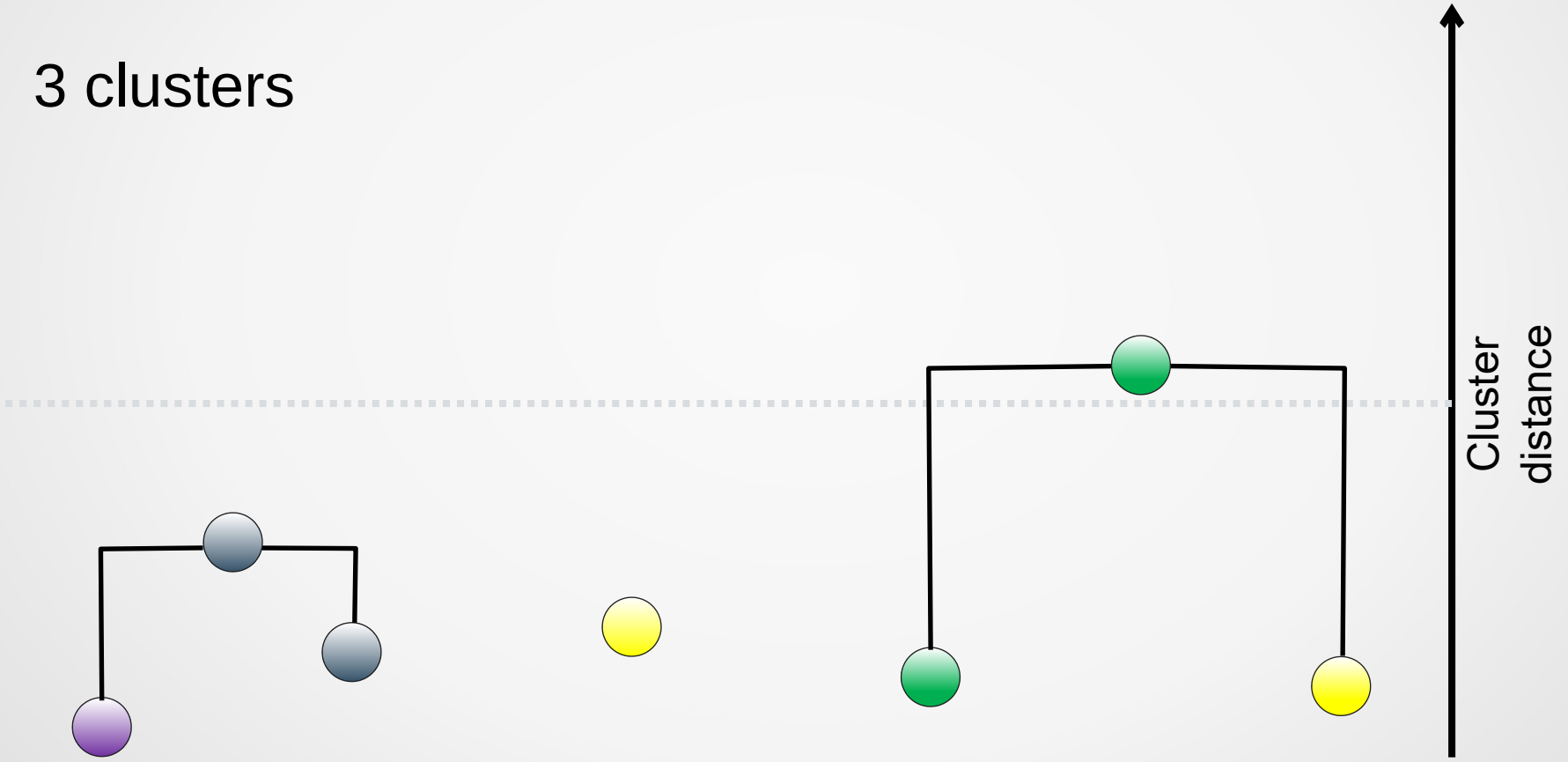
Heirarchical Agglomerative Clustering

3 clusters



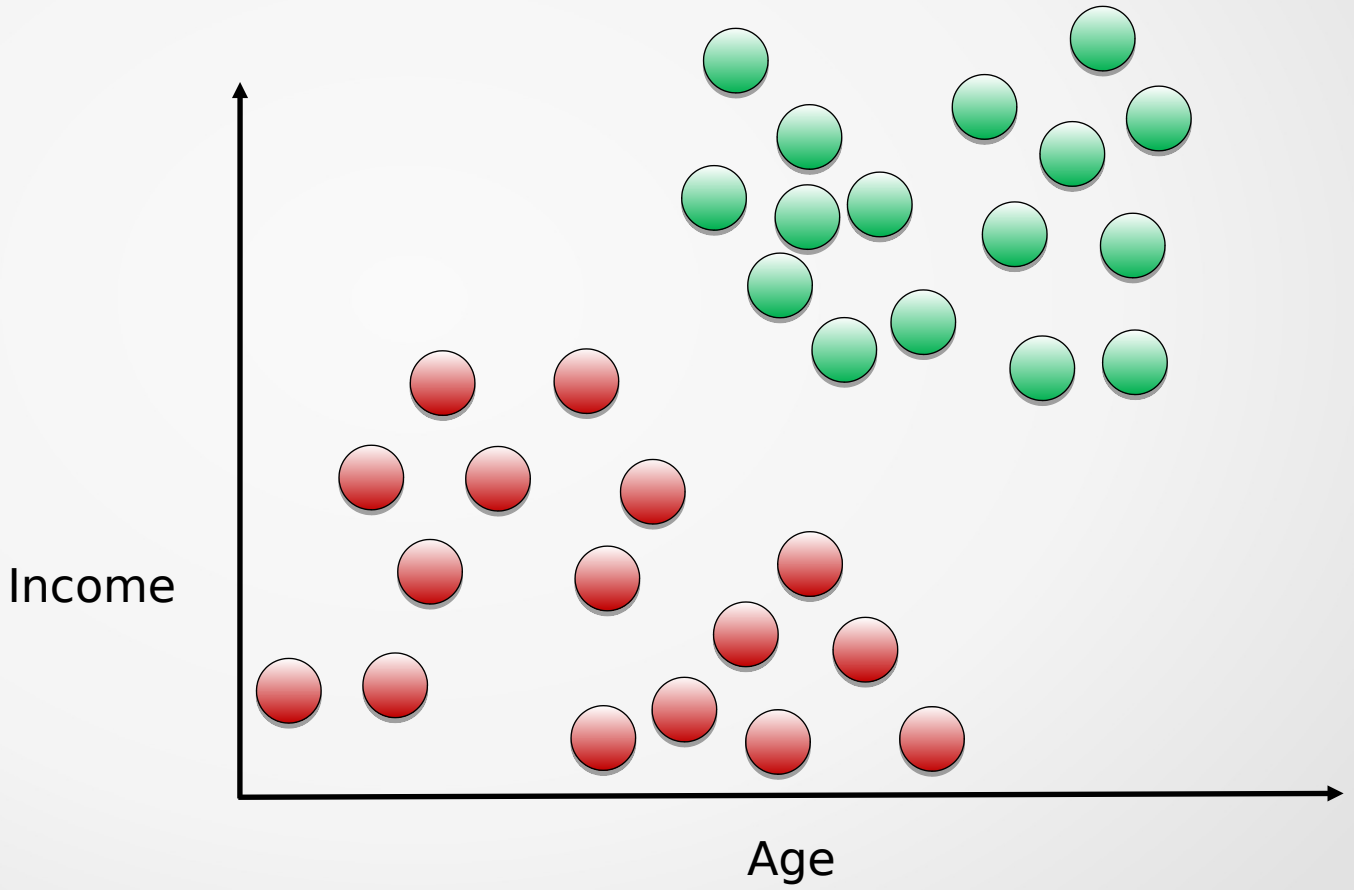
Heirarchical Agglomerative Clustering

3 clusters



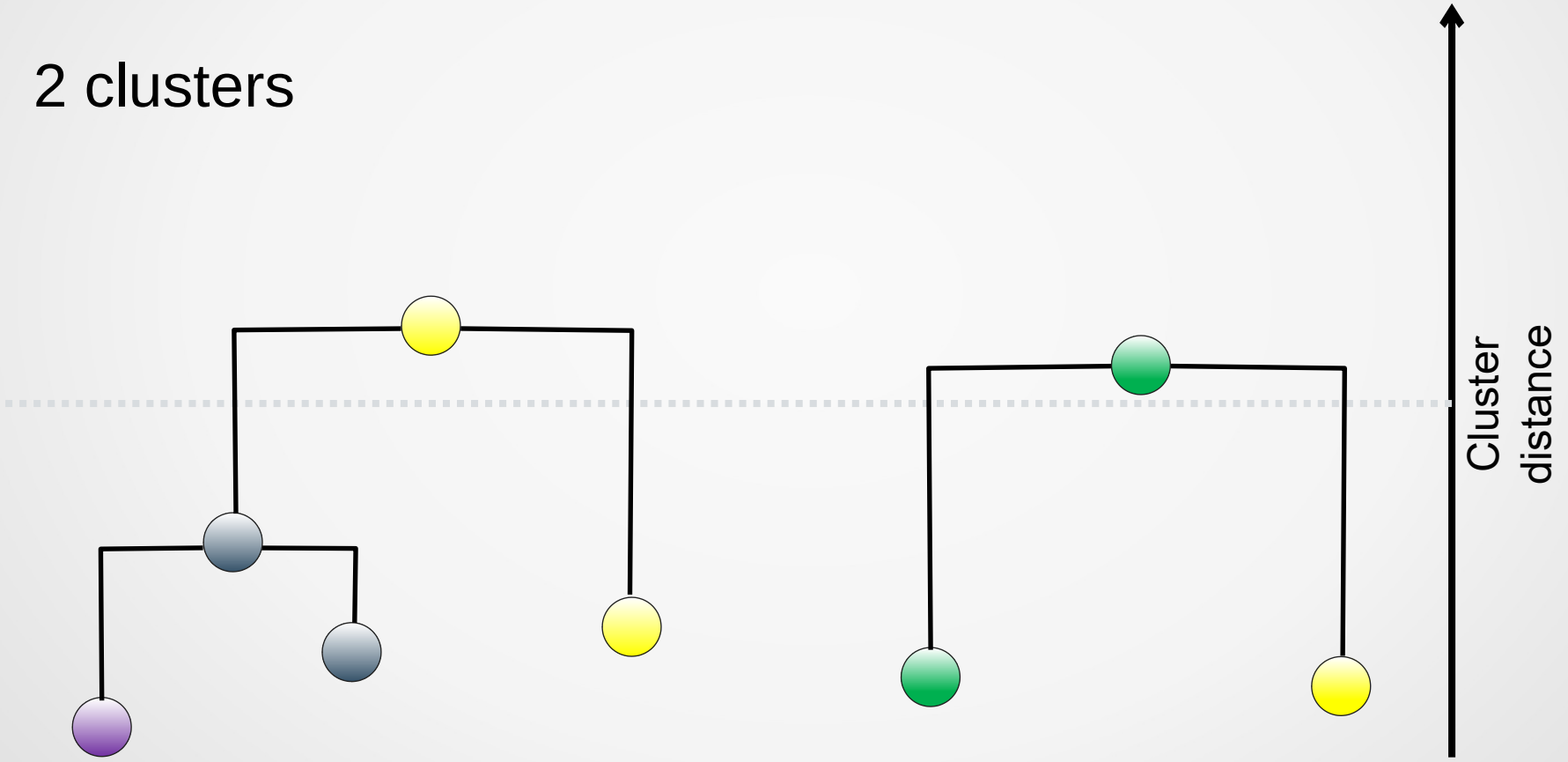
Heirarchical Agglomerative Clustering

2 clusters



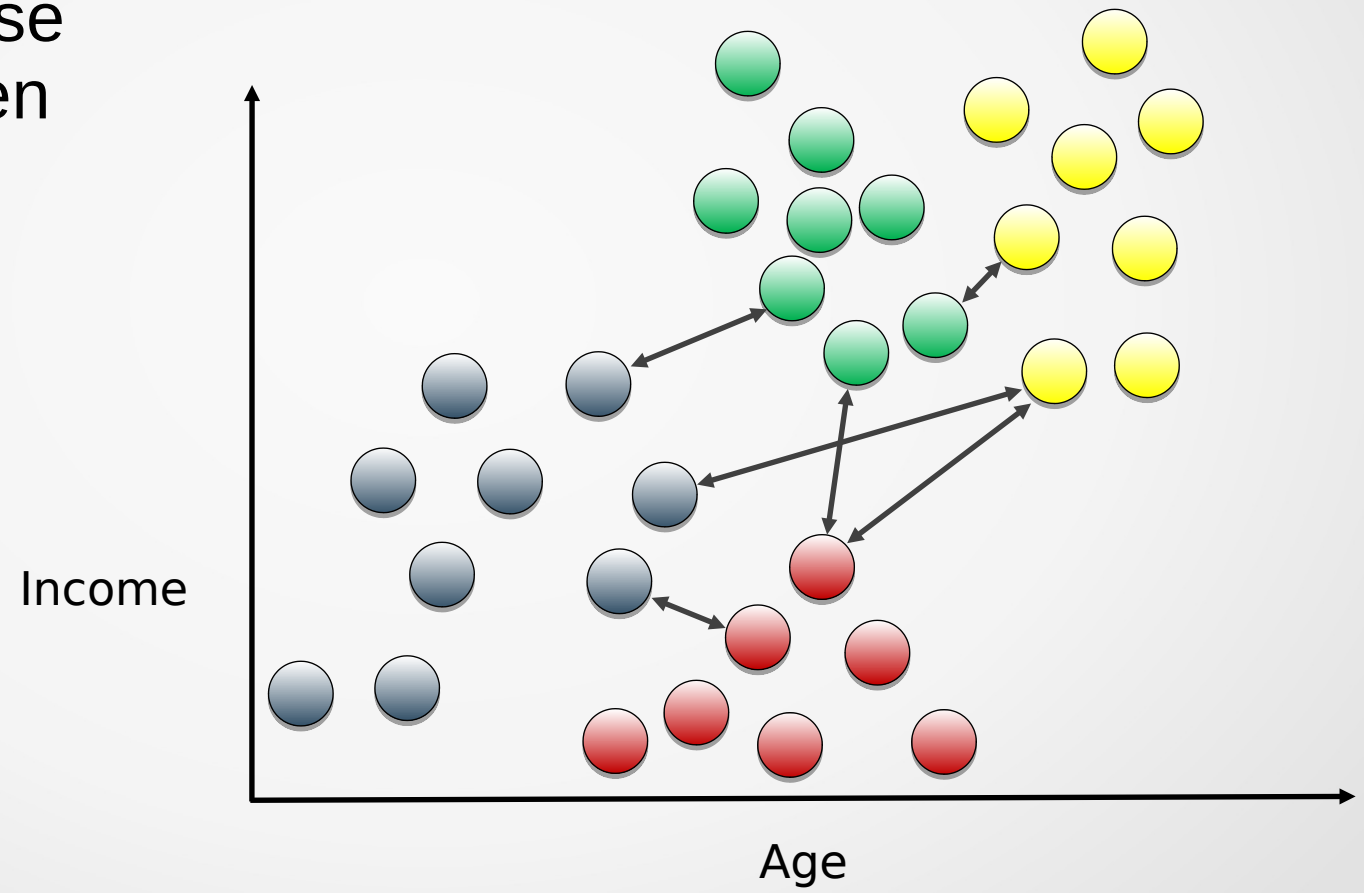
Heirarchical Agglomerative Clustering

2 clusters



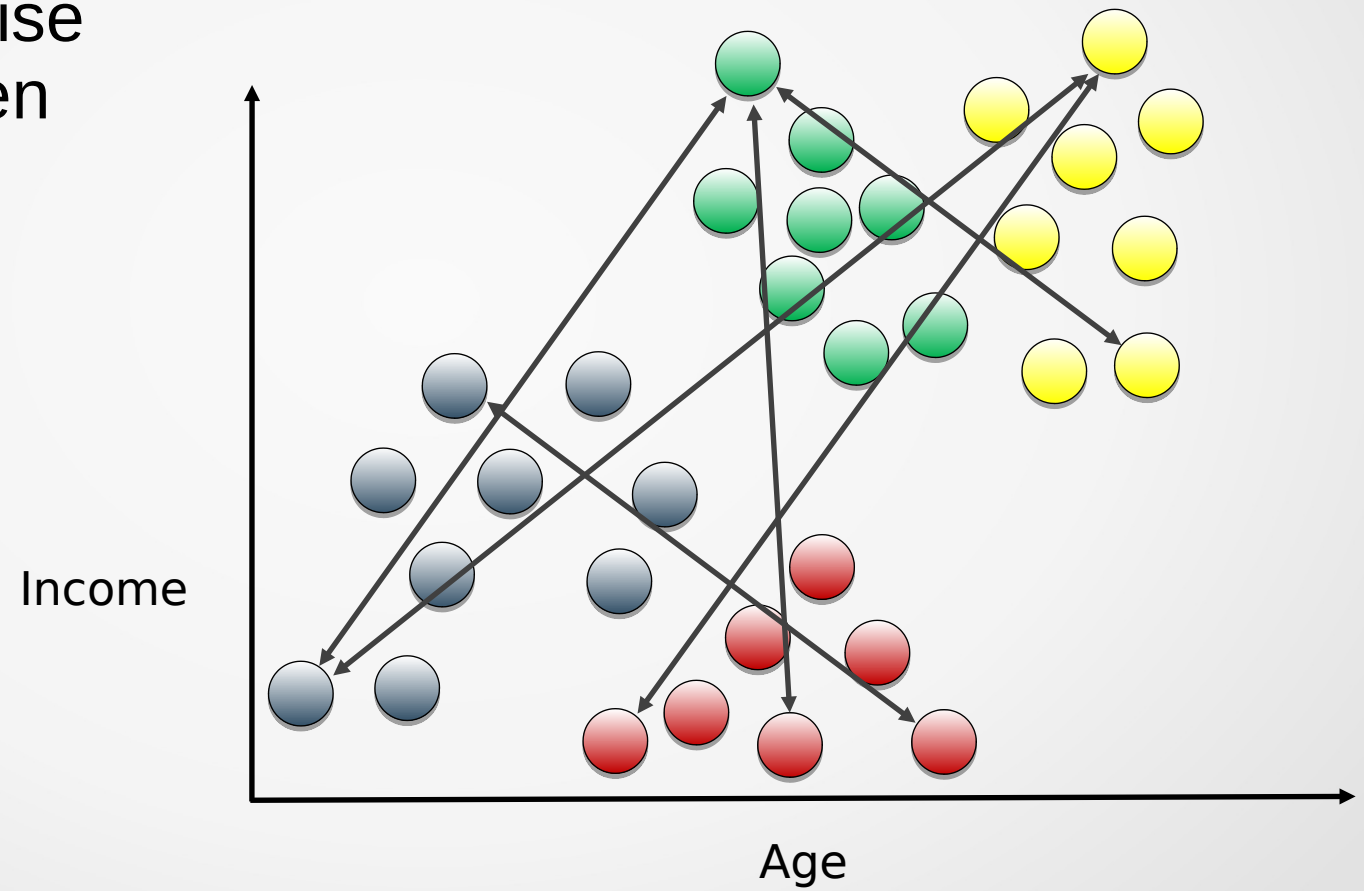
Hierarchical Linkage Types

Single linkage: minimum pairwise distance between clusters



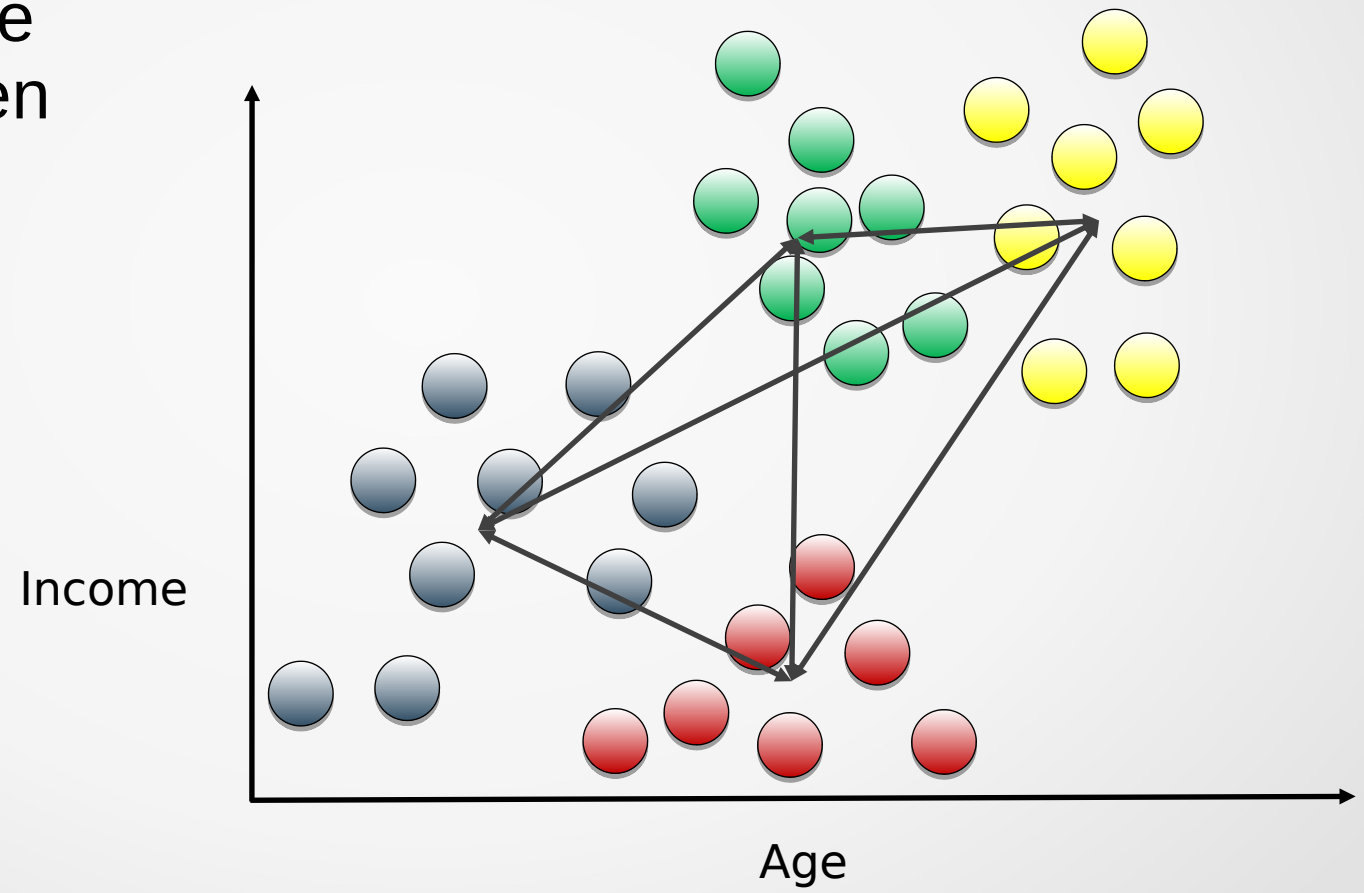
Hierarchical Linkage Types

Complete linkage: maximum pairwise distance between clusters



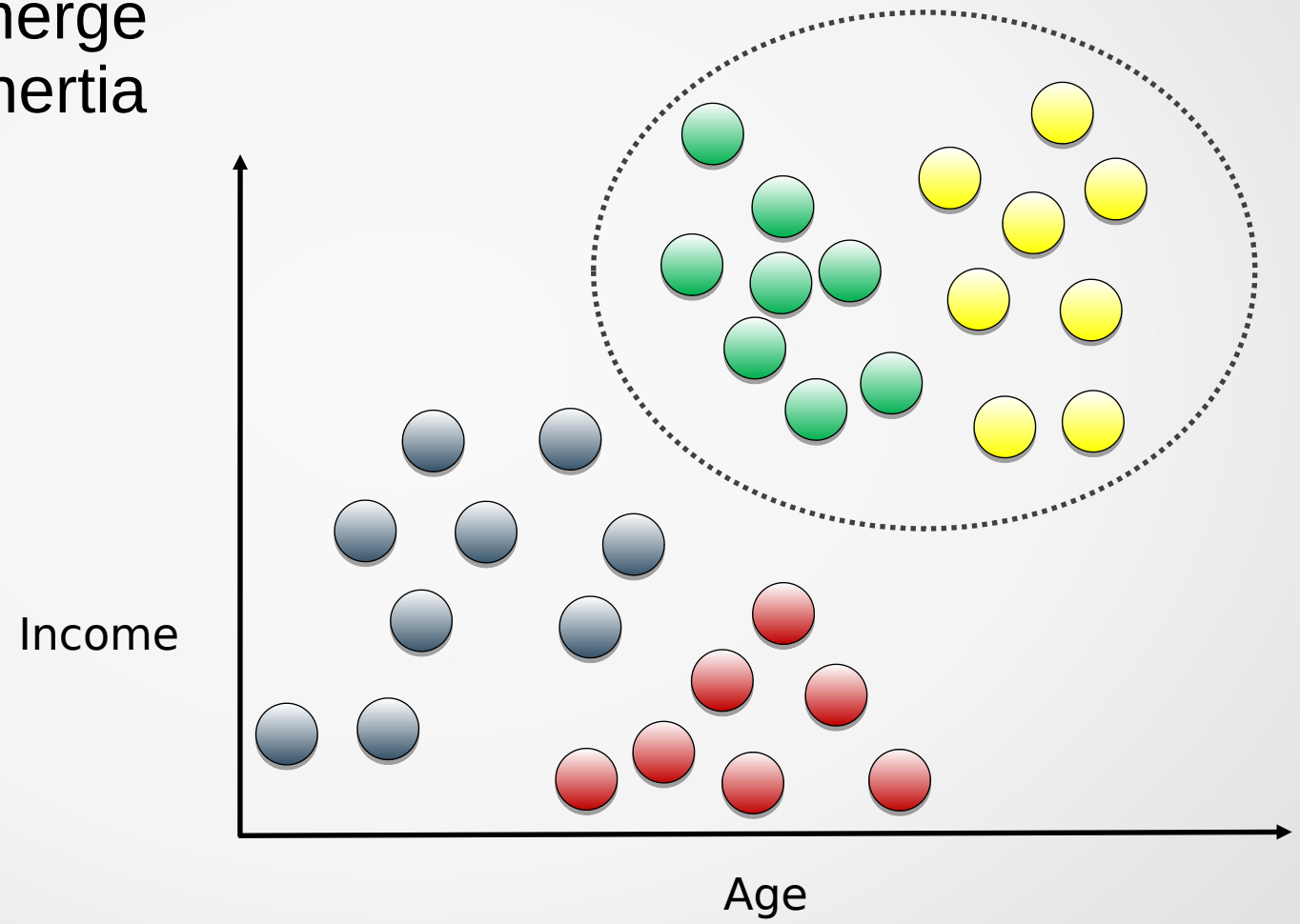
Hierarchical Linkage Types

Average linkage: average pairwise distance between clusters



Hierarchical Linkage Types

Ward linkage: merge based on best inertia



Agglomerative Clustering Syntax

- Import the class containing the clustering method

```
from sklearn.cluster import AgglomerativeCluastering
```

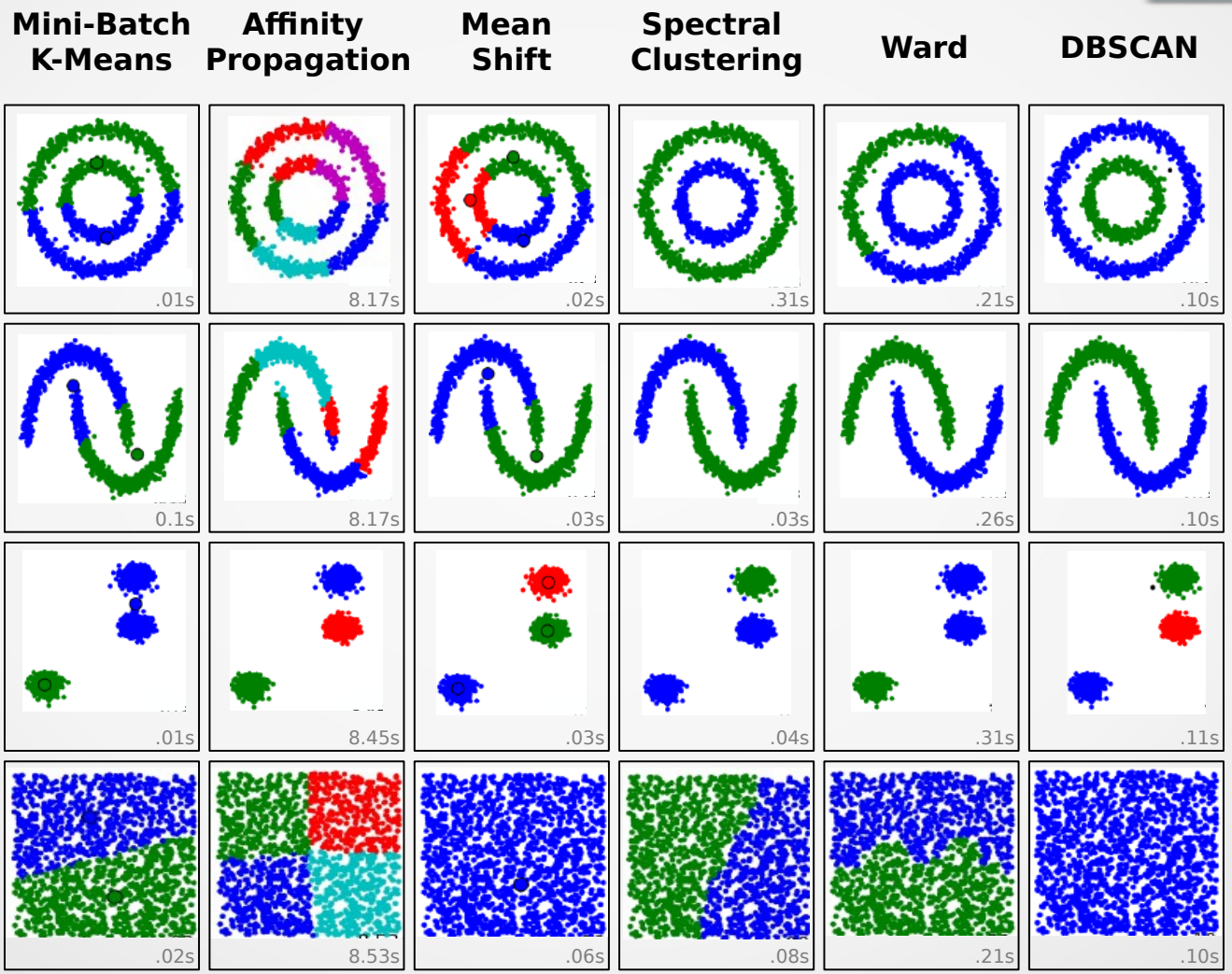
- Create an instance of the class

```
agg = AgglomerativeClustering(n_clusters=3,
                              affinity='euclidean',
                              linkcage='ward')
```

- Fit the instance on the data and then predict clusters for new data

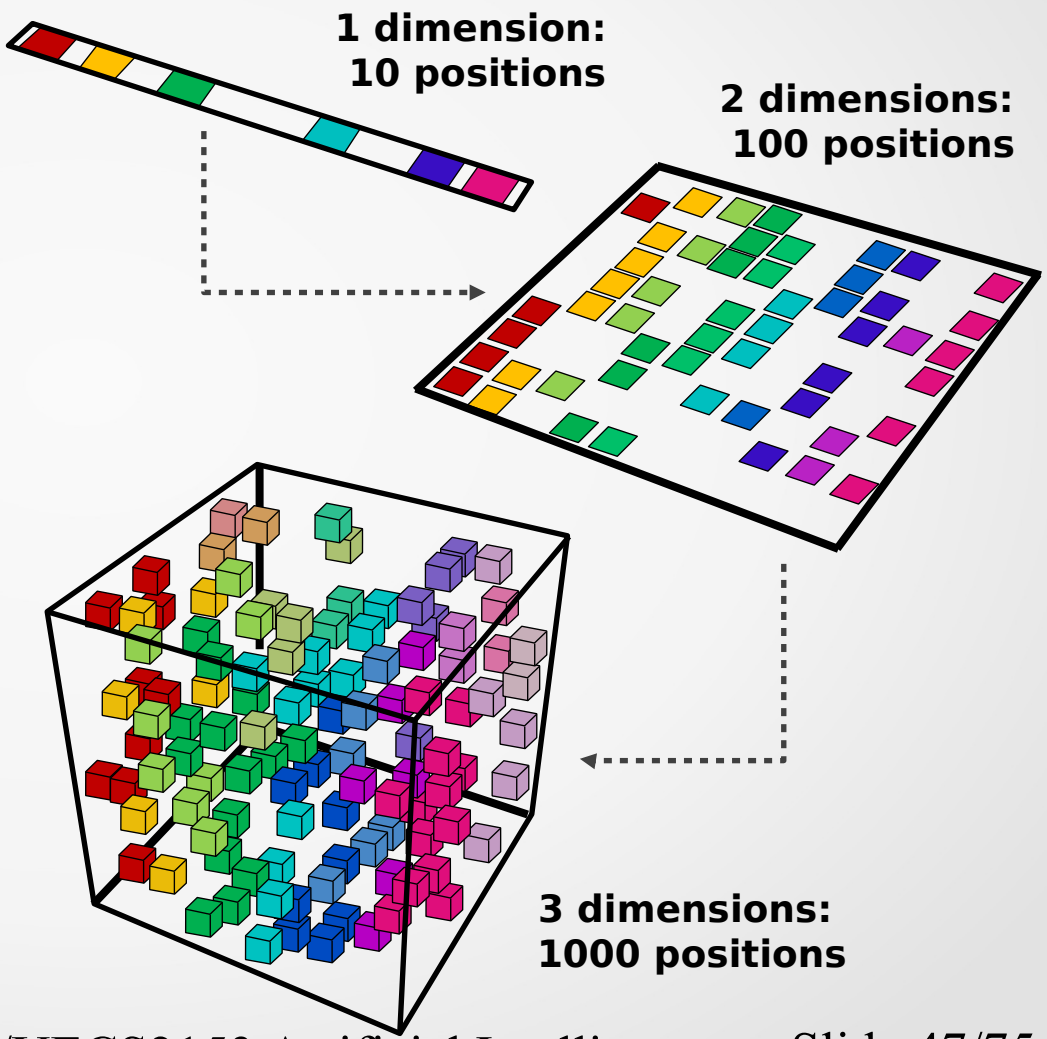
```
agg = agg.fit(x1)
y_predict = agg.predict(x2)
```

Other Types of Clustering



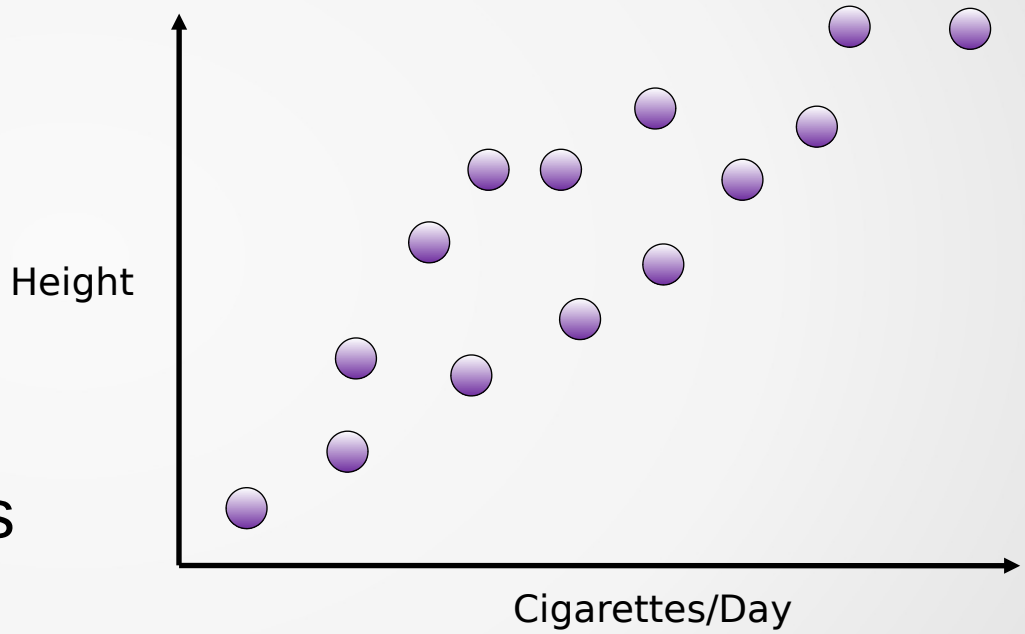
Curse of Dimensionality

- Theoretically, increasing features should improve performance
- In practice, too many features leads to worse performance
- Number of training examples required increases exponentially with dimensionality



Dimensionality Reduction

- Data can be represented by fewer dimensions (features)
- Reduce dimensionality by selecting subset (feature elimination)
- Combine with linear and non-linear transformations

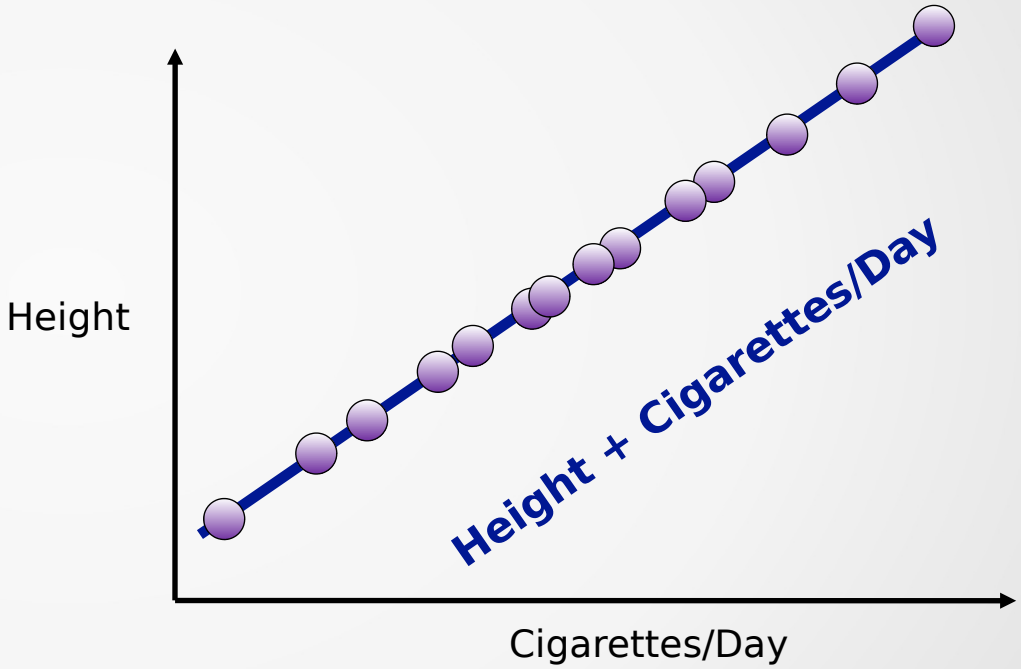


Dimensionality Reduction

- Two features: height and cigarettes per day
- Both features increase together (correlated)
- Can we reduce number of features to one?

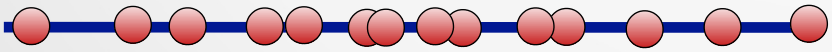
Dimensionality Reduction

- Two features: height and cigarettes per day
- Both features increase together (correlated)
- Can we reduce number of features to one?

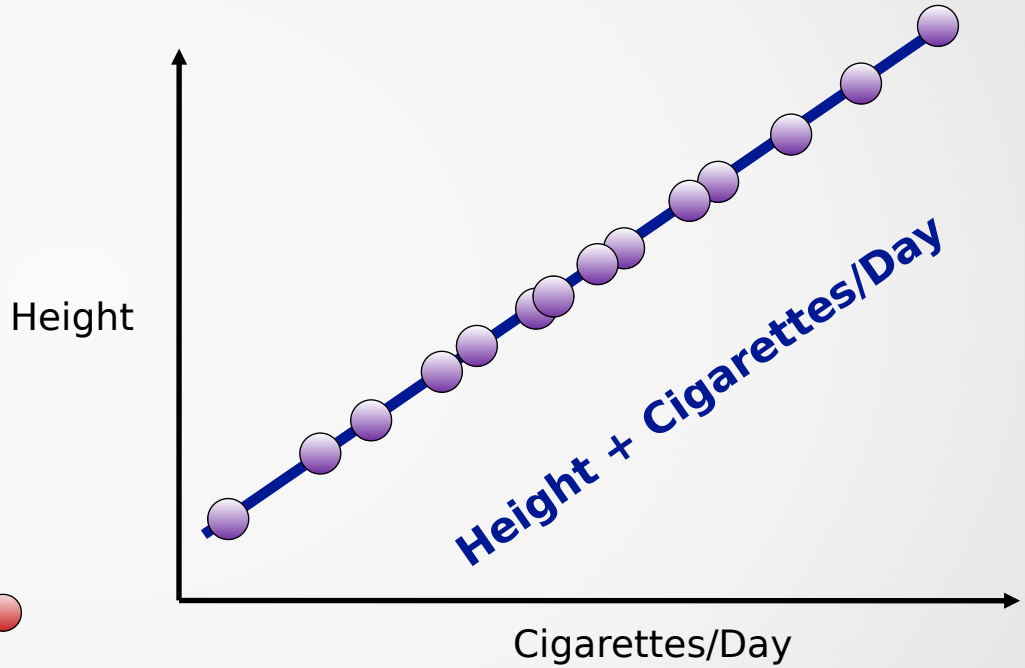


Dimensionality Reduction

- Create single feature that is combination of height and cigarettes
- This is Principal Component Analysis (PCA)



Height + Cigarettes/Day



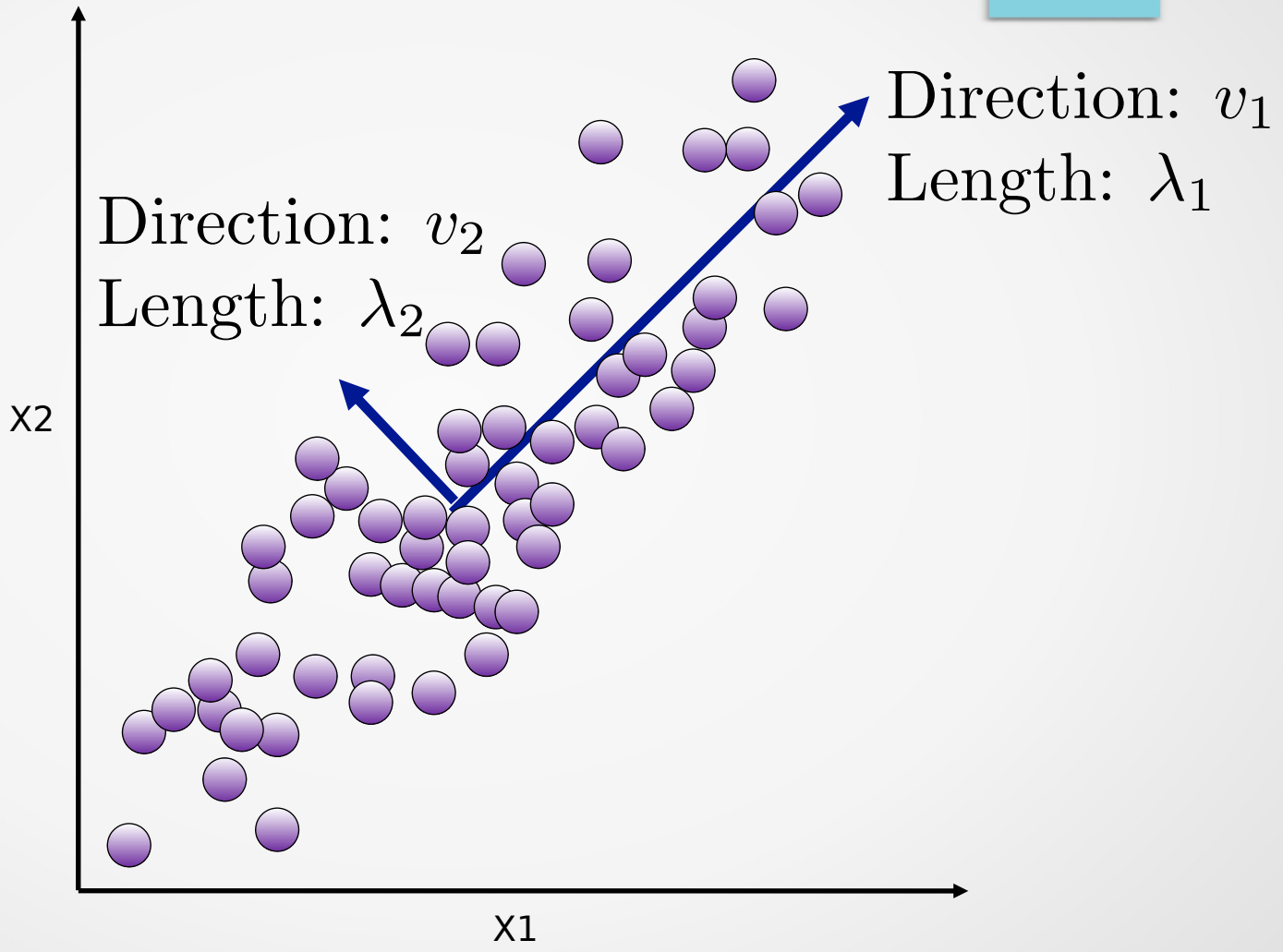
Dimensionality Reduction

Given a *N*-dimensional data set (*x*):-
find a *N x K* matrix (*U*)

$y = U^T x$ where *y* has *K* dimensions and *K* < *N*

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \; U^T \; \rightarrow \; y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} \; (K < N)$$

Principle Component Analysis (PCA)



Single Value Decomposition (SVD)

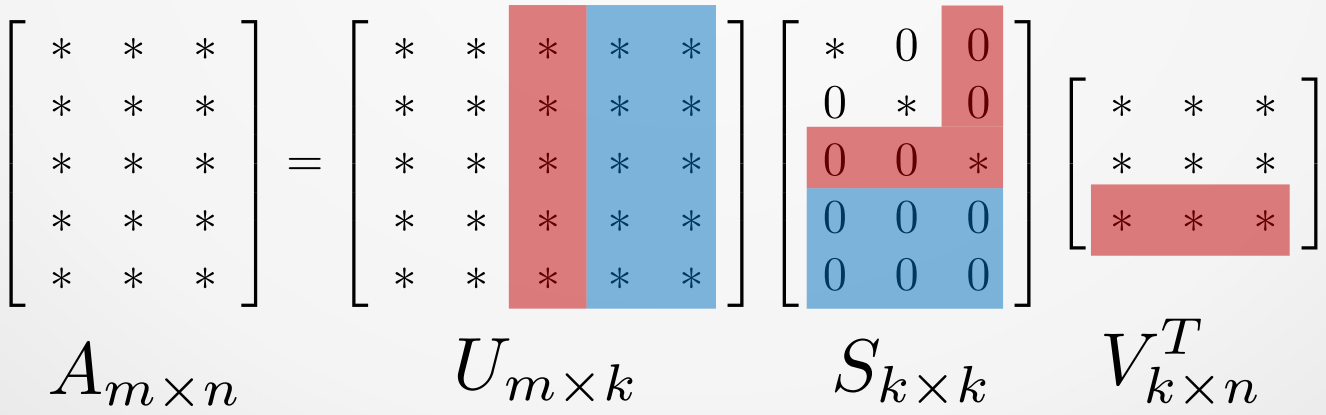
- SVD is a matrix factorization method normally used for PCA
- Does not require a square data set
- SVD is used by Scikit-learn for PCA

$$\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix} \begin{bmatrix} * & 0 & 0 \\ 0 & * & 0 \\ 0 & 0 & * \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}$$

$A_{m \times n}$ $U_{m \times m}$ $S_{m \times n}$ $V^T_{n \times n}$

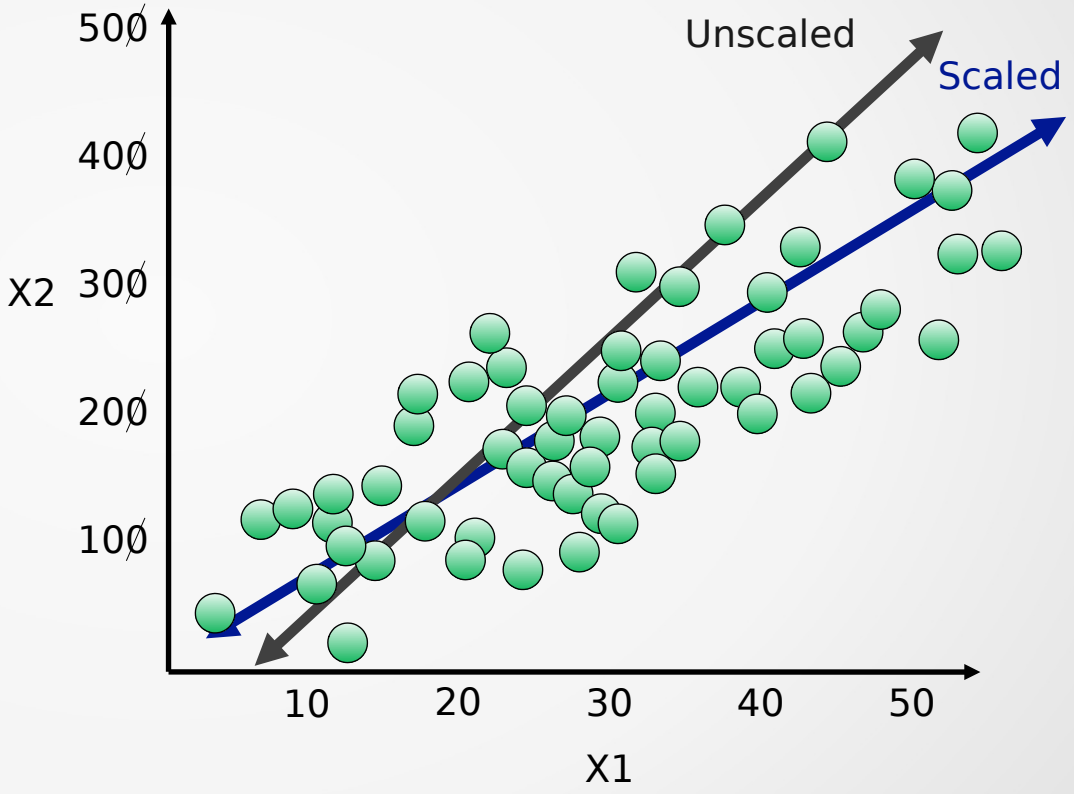
Truncated Single Value Decomposition

- How can SVD be used for dimensionality reduction?
- Principal components are calculated from US
- "Truncated SVD" used for dimensionality reduction $(n \rightarrow k)$



Reminder: Feature Scaling

- PCA and SVD seek to find the vectors that capture the most variance
- Variance is sensitive to axis scale
- Remember to scale your data!



PCA Syntax

- Import the class containing the dimensionality reduction method

```
from sklearn.decomposition import PCA
```

- Create an instance of the class

```
PCAinst = PCA(n_components=3, whiten=True)
```

- Fit the instance on the data and then transform the data

```
x_trans = PCAinst.fit_transform(x_train)
```

- Does not work with sparse matrices

Truncated SVD Syntax

- Import the class containing the dimensionality reduction method

```
from sklearn.decomposition import TruncatedSVD
```

- Create an instance of the class

```
SVD = TruncatedSVD(n_components=3)
```

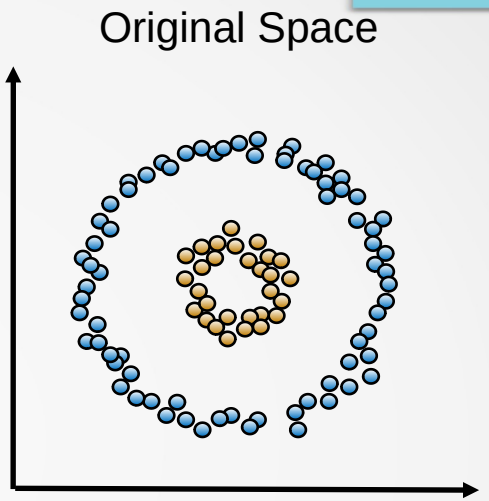
- Fit the instance on the data and then transform the data

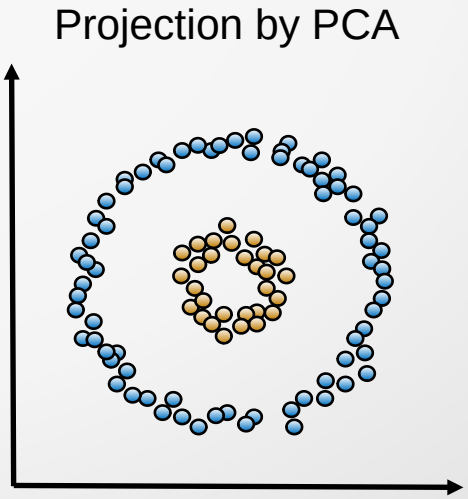
```
x_trans = SVD.fit_transform(x_train)
```

- Works with sparse matrices, used for text data with Latent Smenatic Analysis (LSA)

Non-linear Data Features

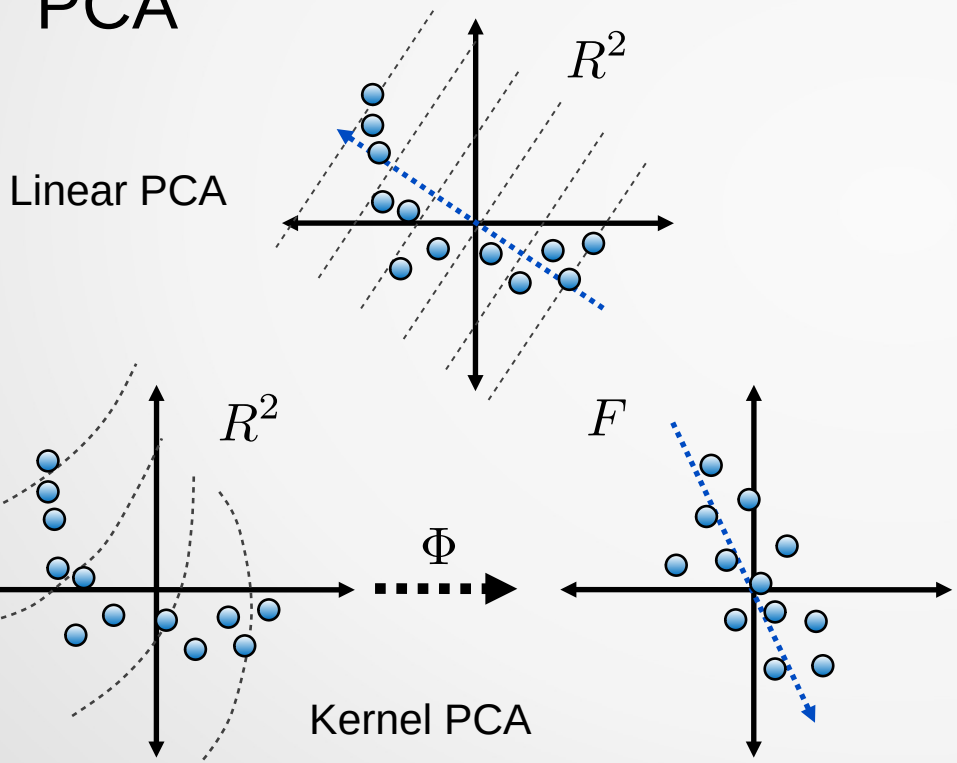
- Transformations calculated with PCA/SVD are linear
- Data can have non-linear features
- This can cause dimensionality reduction to fail

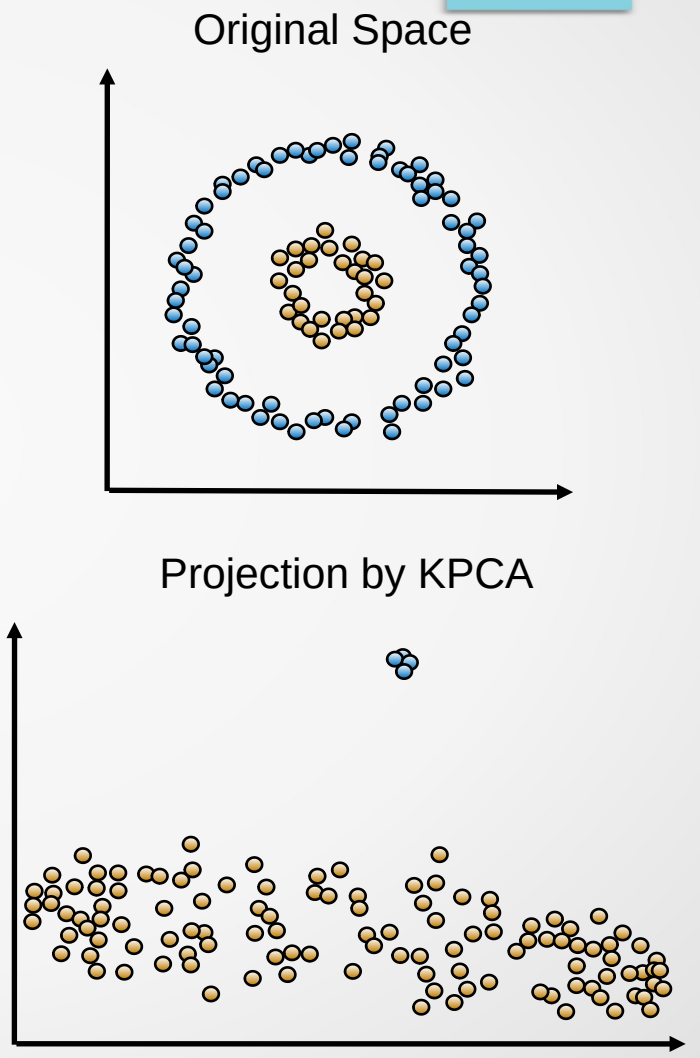




Kernel PCA

- **Solution:** kernels can be used to perform non-linear PCA





Kernel PCA Syntax

- Import the class containing the dimensionality reduction method

```
from sklearn.decomposition import KernelPCA
```

- Create an instance of the class

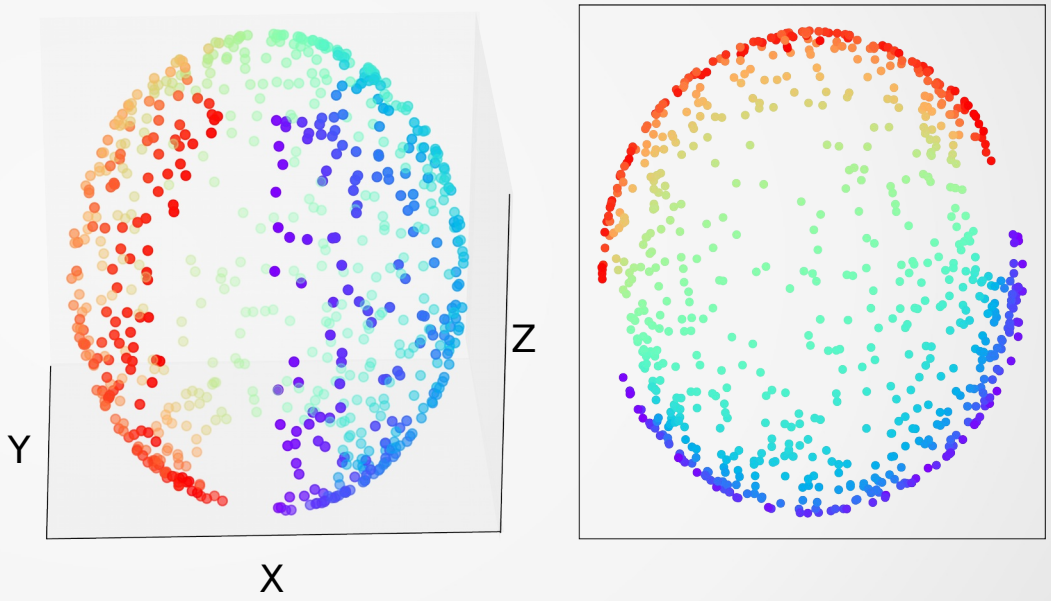
```
kPCA = KernelPCA(n_components=3, kernel='rbf', gamma=1.0)
```

- Fit the instance on the data and then transform the data

```
x_trans = kPCA.fit_transform(x_train)
```

Multi-Dimensional Scaling (MDS)

- Non-linear transformation
- Doesn't focus on maintaining overall variance
- Instead, maintains geometric distances between points



Multi-Dimensional Scaling Syntax

- Import the class containing the dimensionality reduction method

```
from sklearn.decomposition import MDS
```

- Create an instance of the class

```
mdsMod = MDS(n_components=2)
```

- Fit the instance on the data and then transform the data

```
x_trans = mdsMod.fit_transform(x_train)
```

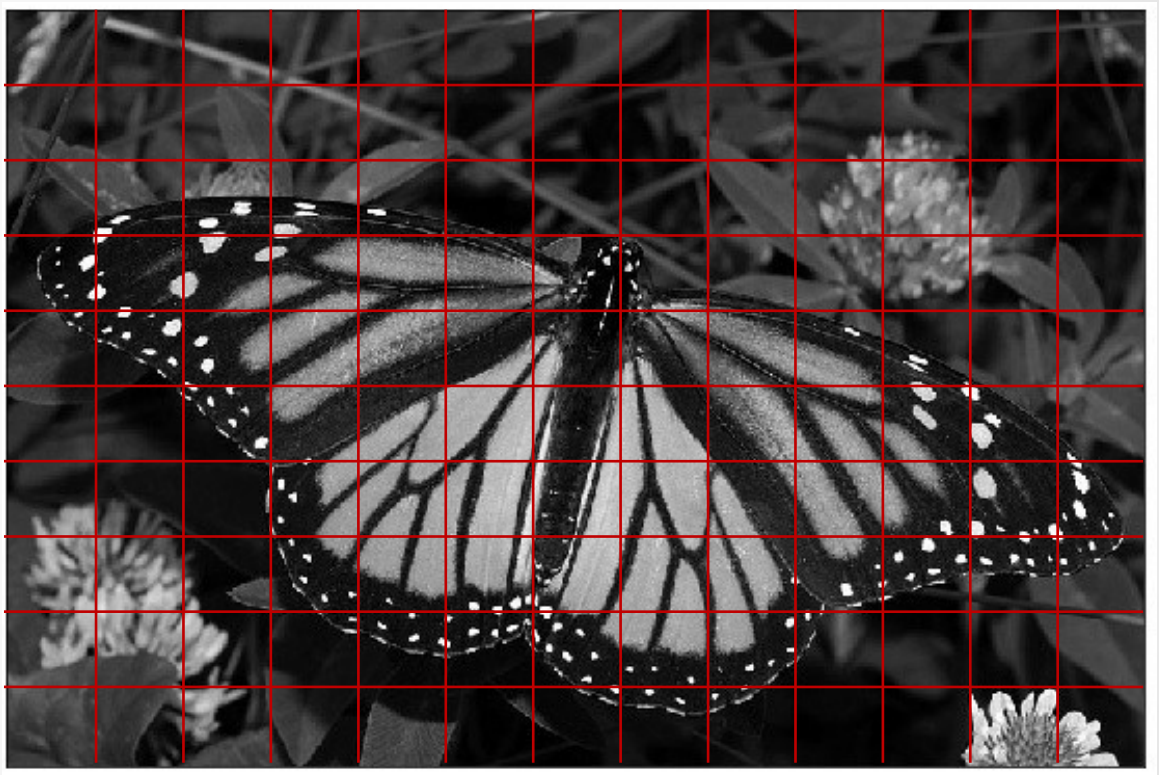
- Many other manifold dimensionality methods exist: Isomap, TSNE etc.

Dimensionality Reduction Application

- Frequently used for high dimensionality data
- Natural language processing (NLP) – many word combinations
- Image-based data sets – pixels are features

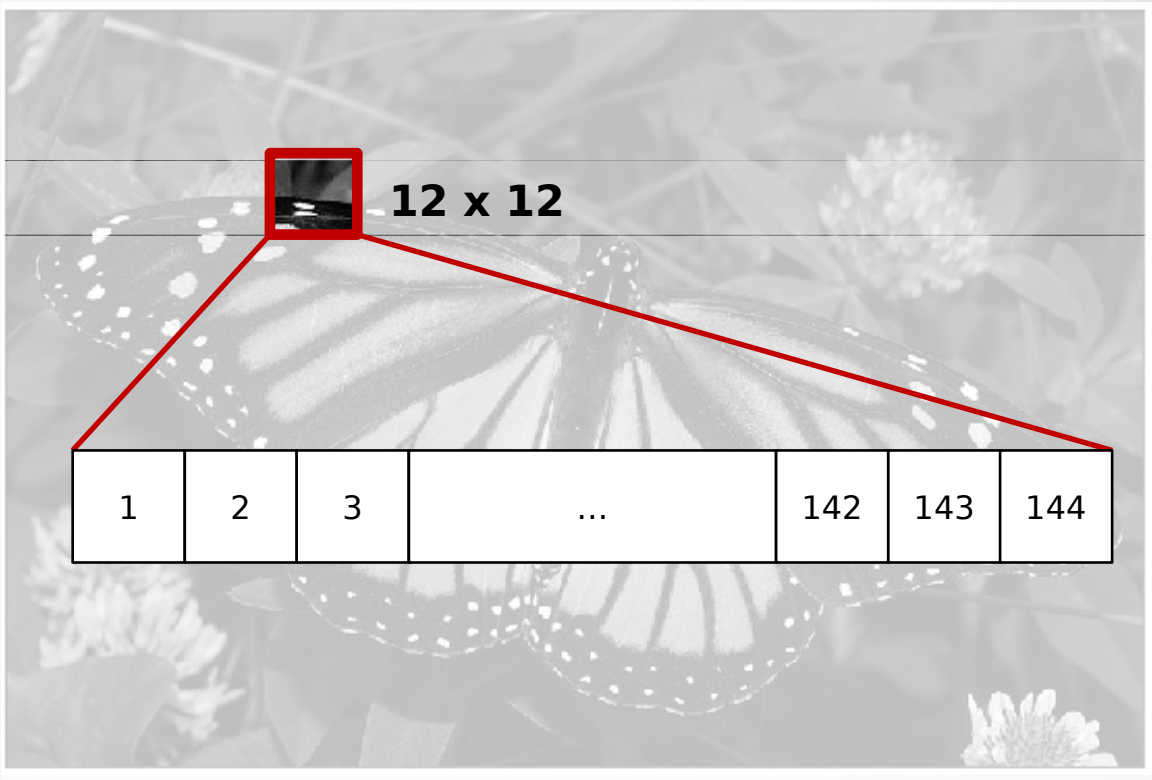
Dimensionality Reduction Application

- Divide image into 12 x 12 pixel sections



Dimensionality Reduction Application

- Divide image into 12 x 12 pixel sections
- Flatten section to create row of data with 144 features



Dimensionality Reduction Application

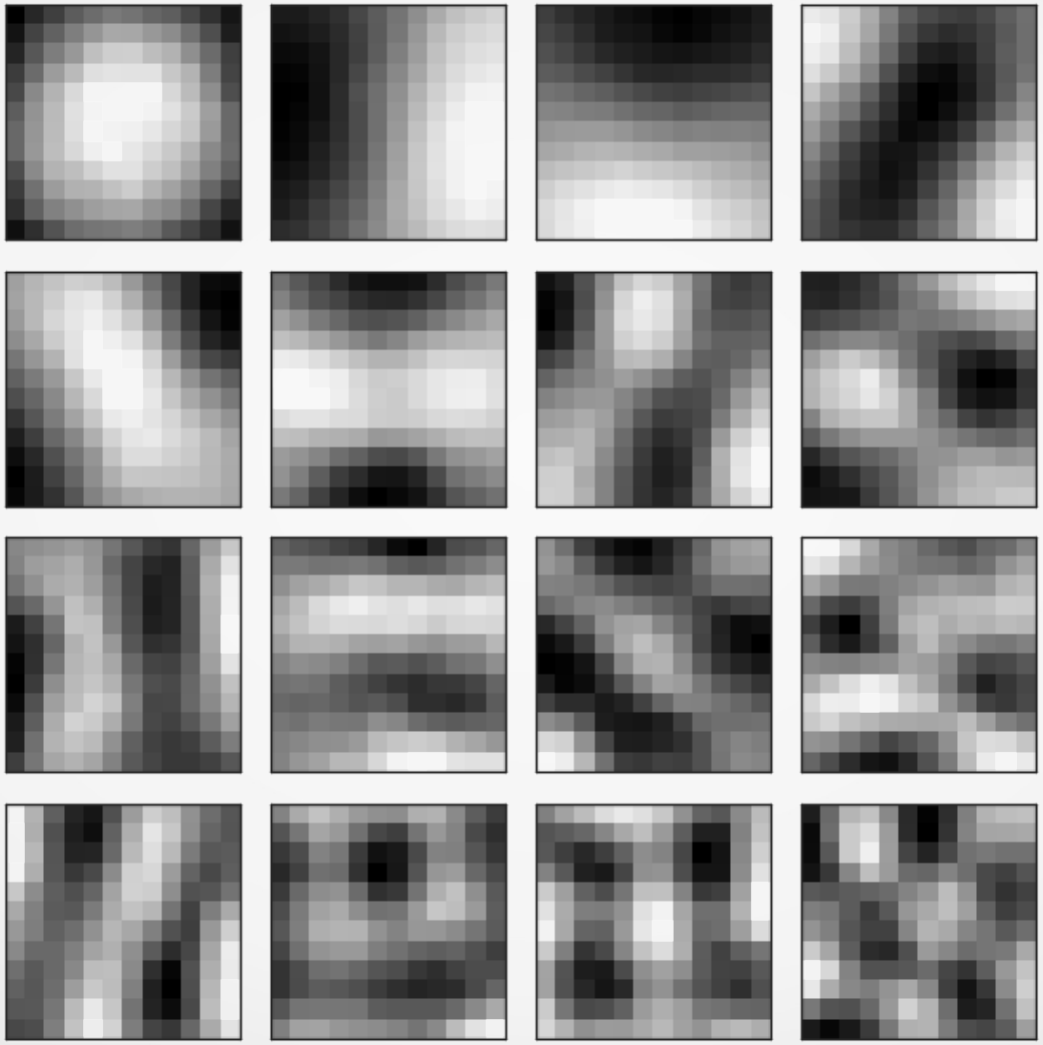
- Divide image into 12 x 12 pixel sections
- Flatten section to create row of data with 144 features
- Perform PCA on all data points

1	2	3	...	142	143	144
1	2	3	...	142	143	144
1	2	3	...	142	143	144
1	2	3	...	142	143	144
1	2	3	...	142	143	144
1	2	3	...	142	143	144

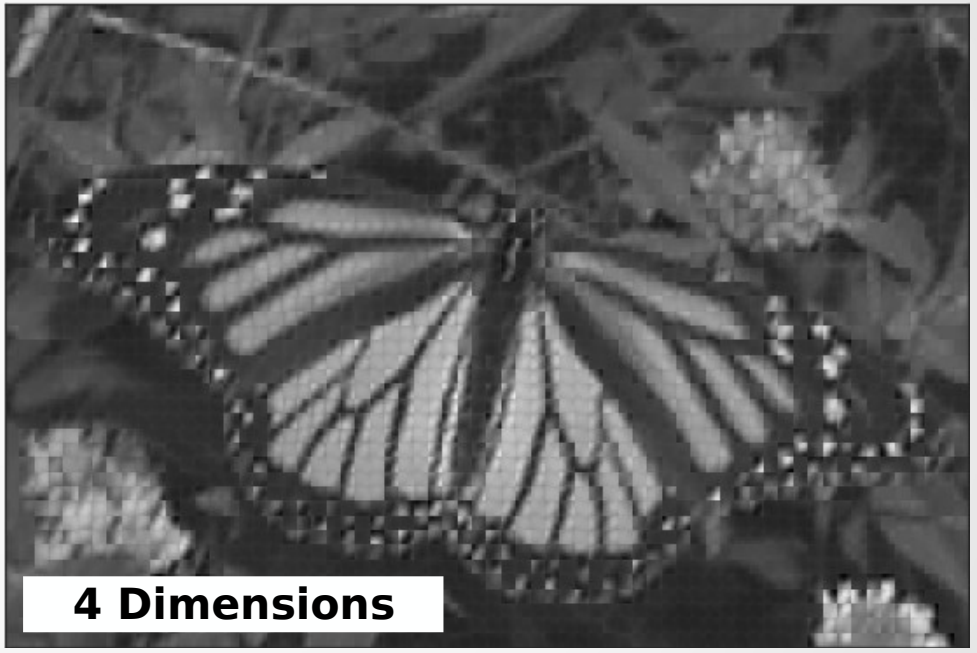
PCA Compression

PCA Compression

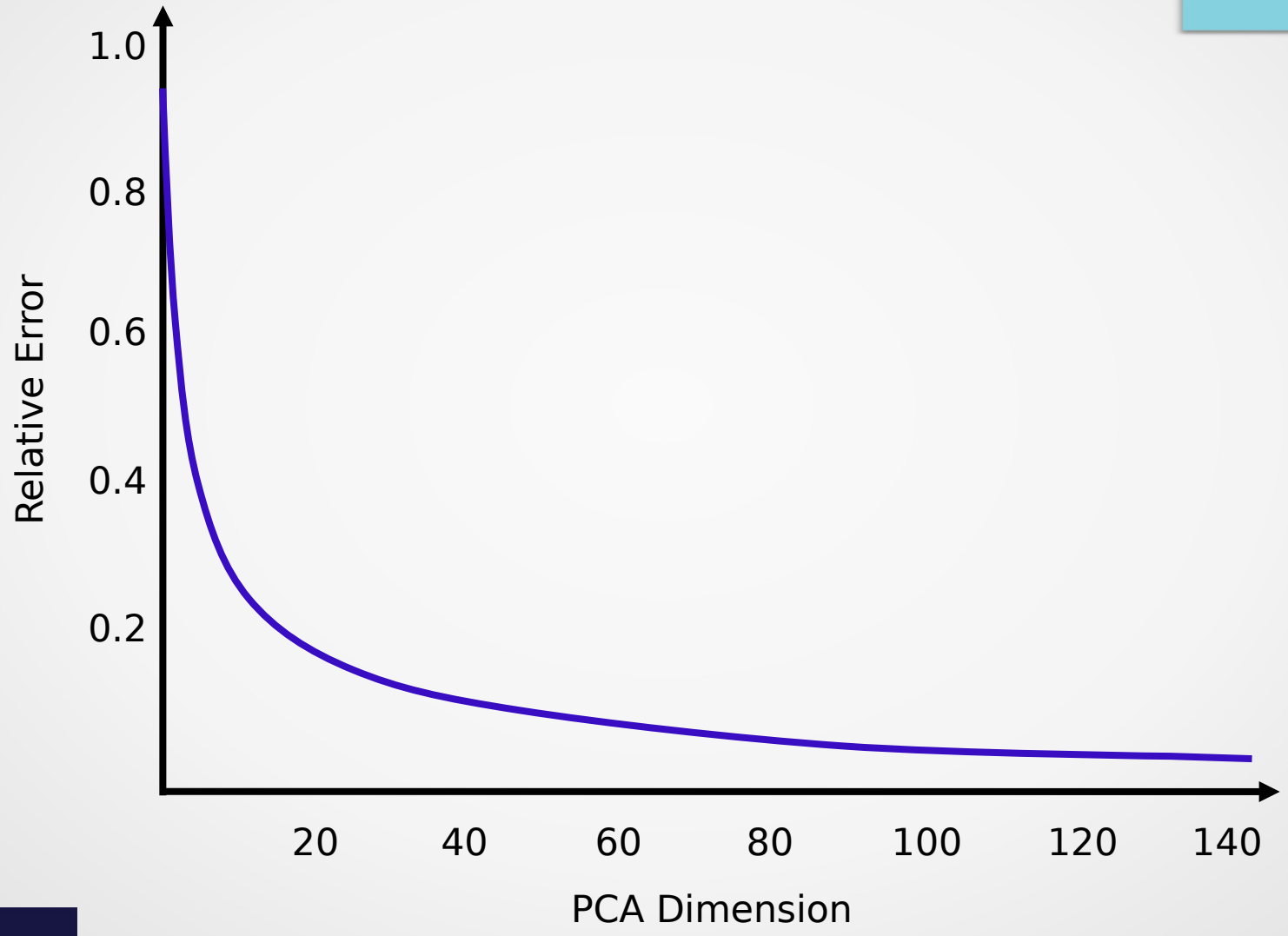
Sixteen Most Important Eigenvectors



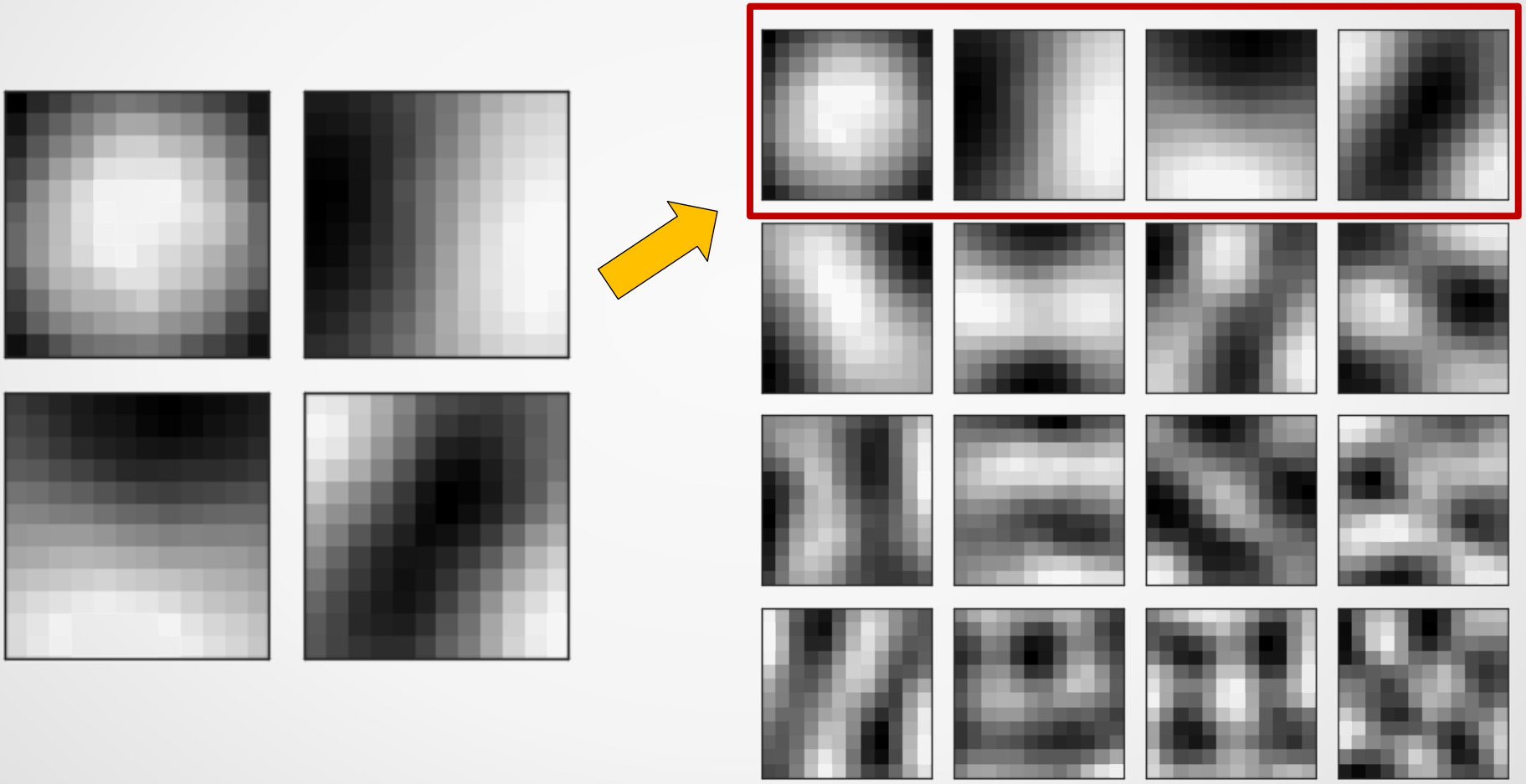
PCA Compression



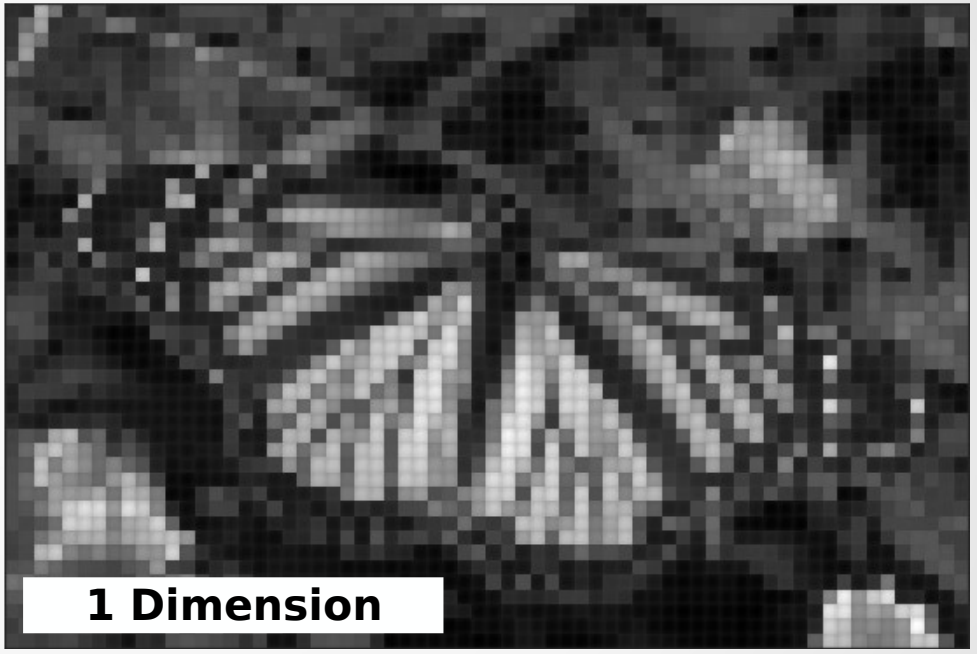
L2 Error and PCA Dimension



Four Most Important Eigenvectors



PCA Compression



End of Lecture

Many thanks to Intel Software for providing a variety of resources for this lecture series