Unsupervised Learning Algorithms

Contents

- Unsupervised learning
- Clustering
- K-Means

Types of Machine Learning

Supervised	data points have known outcome
Unsupervised	data points have unknown outcome

Types of Machine Learning

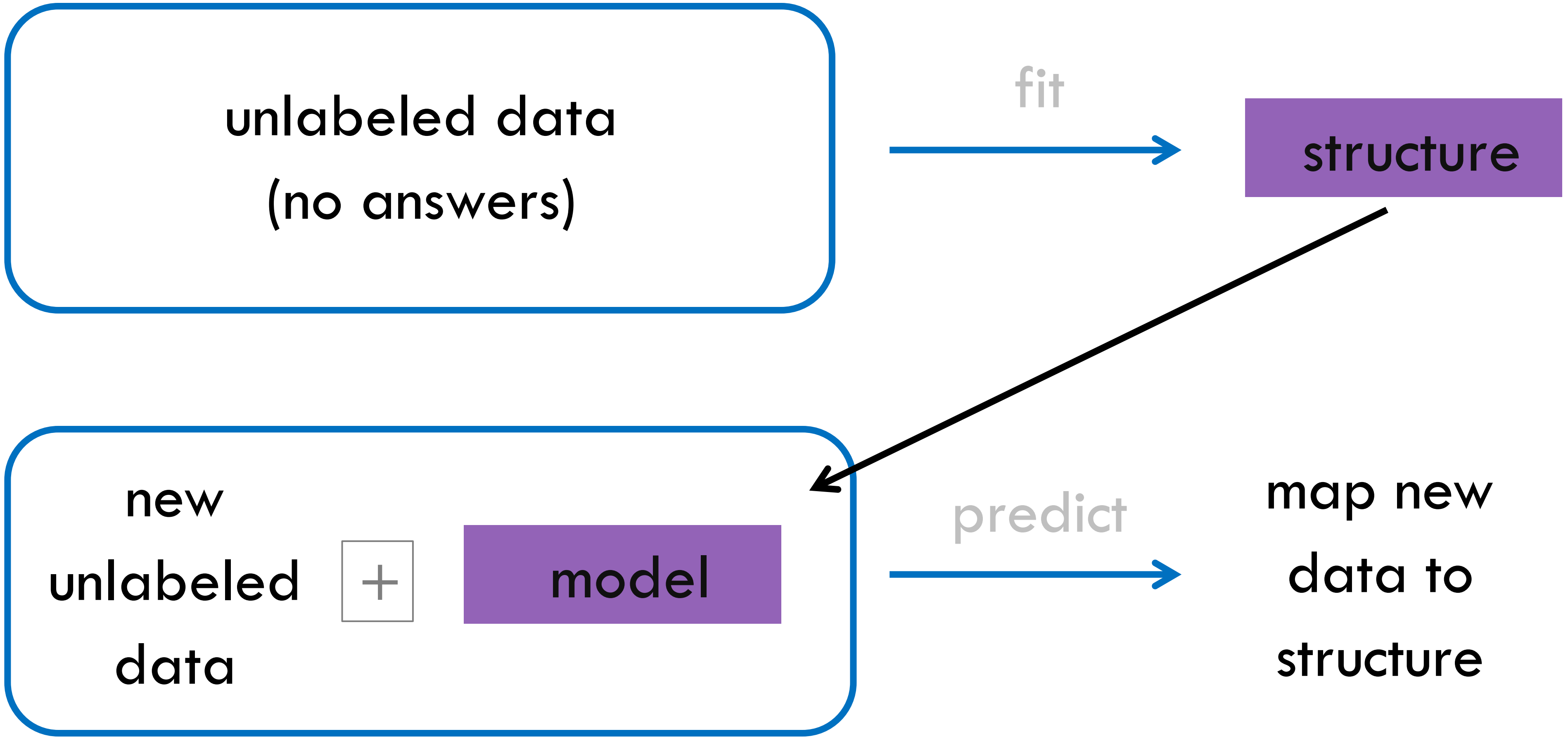
Supervised	data points have known outcome
Unsupervised	data points have unknown outcome

Types of Unsupervised Learning

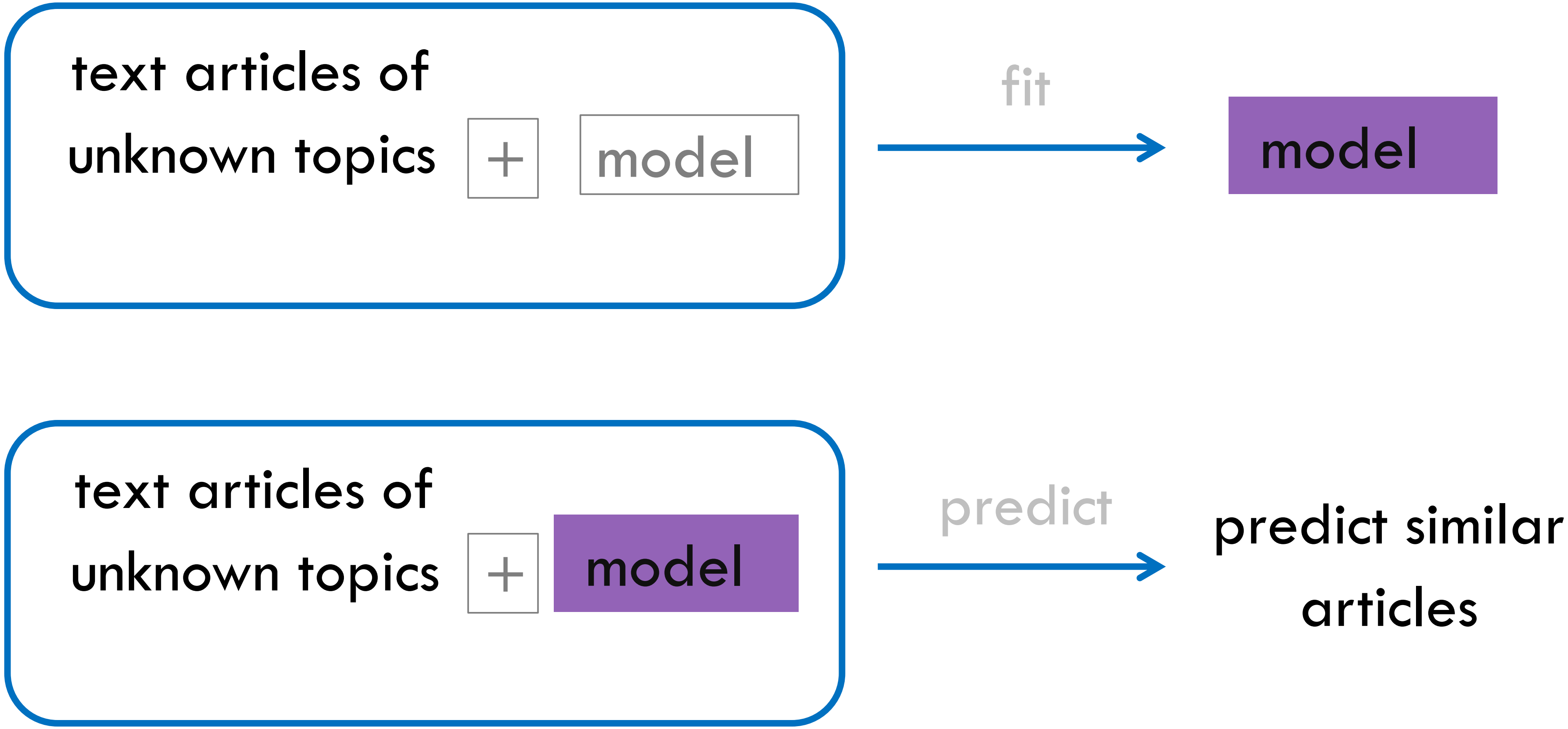
Clustering — identify unknown structure in data

Dimensionality Reduction — use structural characteristics to simplify data

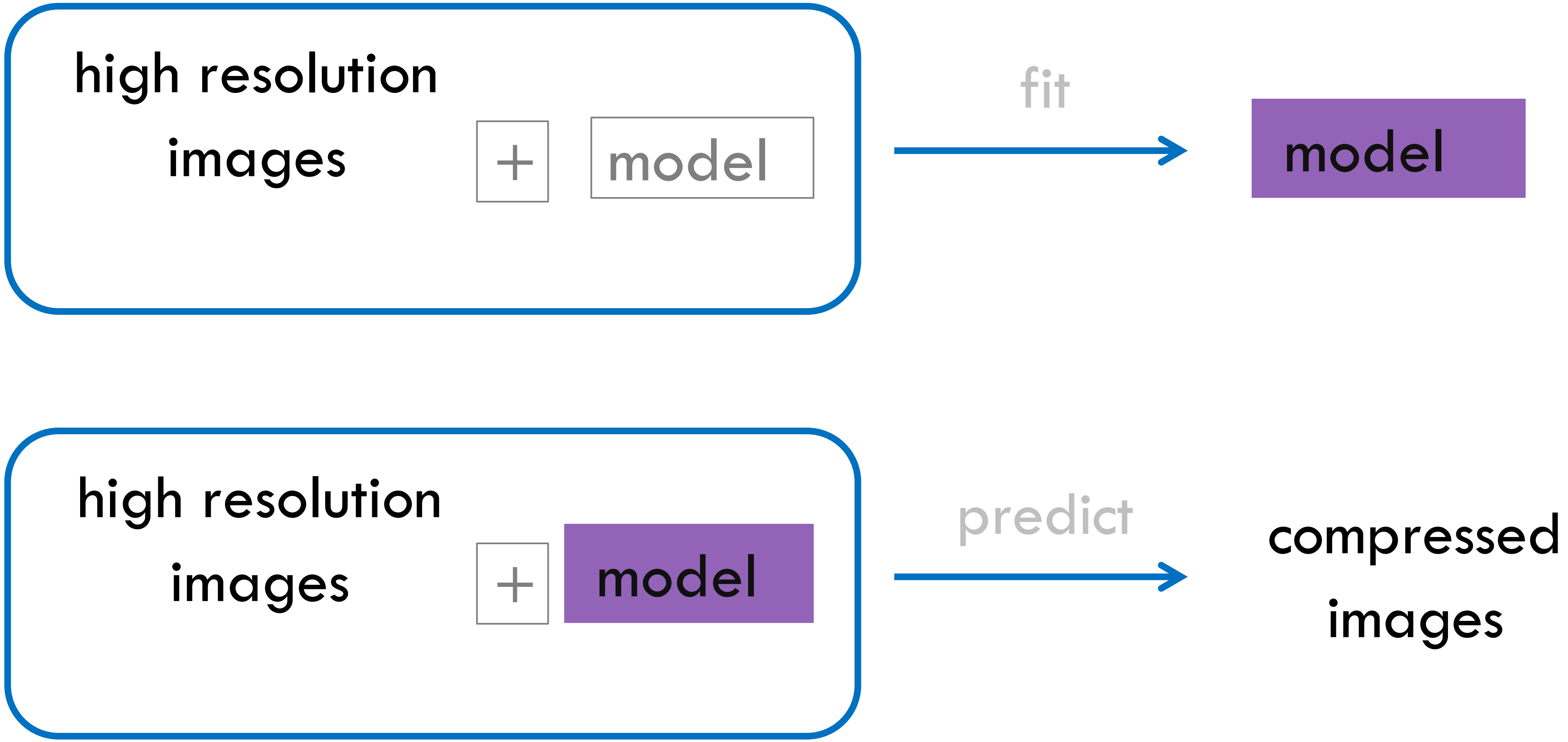
Unsupervised Learning Overview



Clustering: Finding Distinct Groups



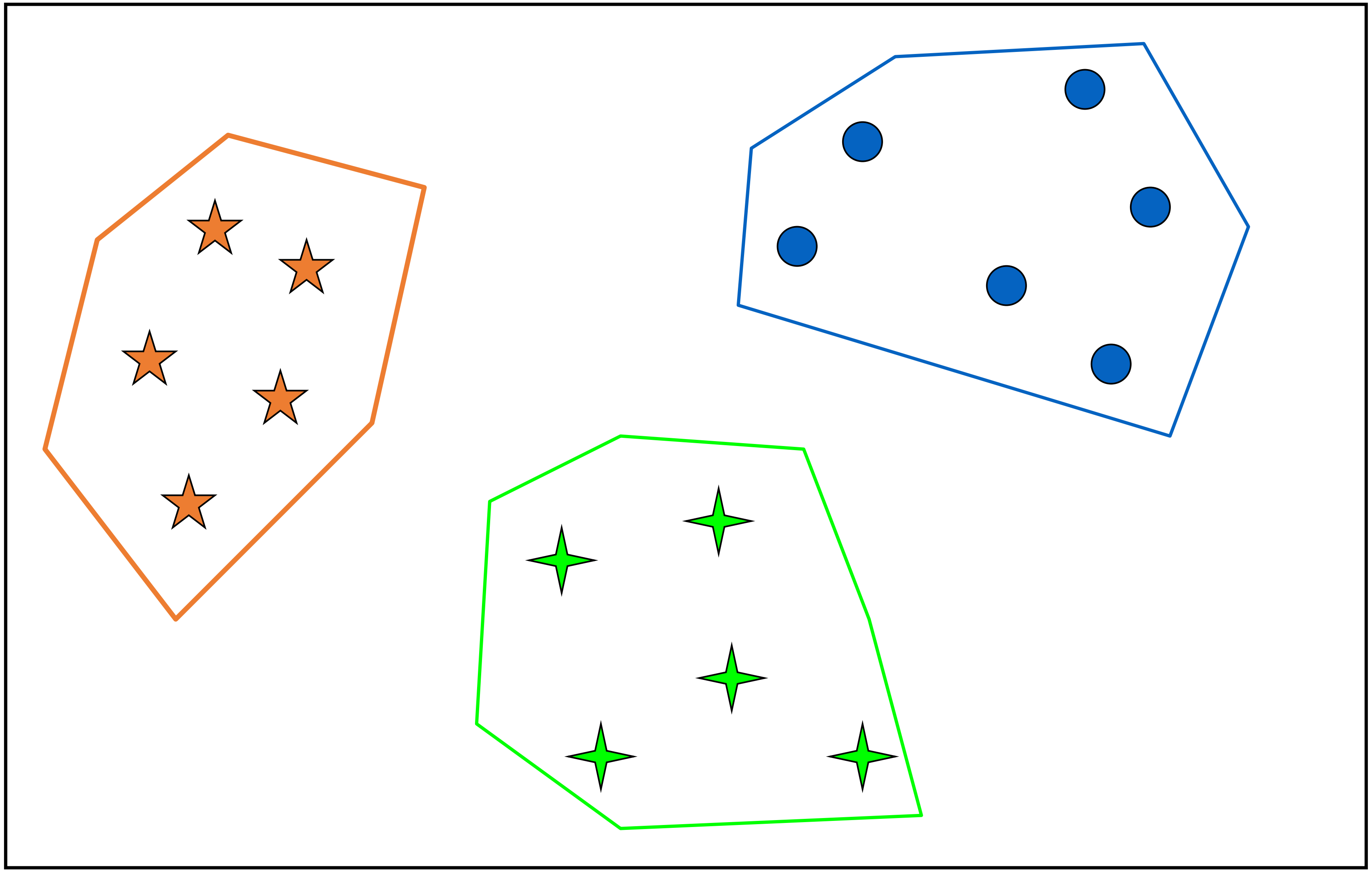
Dimensionality Reduction: Simplifying Structure



Cluster

- The process partitions a dataset into meaningful subclasses, called clusters.
- Clusters consist of data points that are similar to one another.
- These clusters should be treated as distinct groups.

Cluster



Unsupervised Learning Algorithms for Auditors

Detect hidden patterns and groupings in unlabeled audit data.

Clustering algorithms (e.g., K-Means) group similar transactions or vendors.

Anomaly detection helps identify outliers without predefined fraud labels.

Useful for exploring large datasets when risks are unknown.

Supports early risk identification and continuous monitoring strategies.

Clustering Techniques and Potential Applications in Insurance Auditing

Clustering Techniques	Potential Applications
K-Means Clustering	Grouping similar policyholder profiles based on demographics.
Hierarchical Clustering	Identifying clusters of claims based on patterns and severity.
DBSCAN	Detecting outliers in claim amounts for fraud detection.
Gaussian Mixture Models	Segmenting insurance products by customer risk profiles.

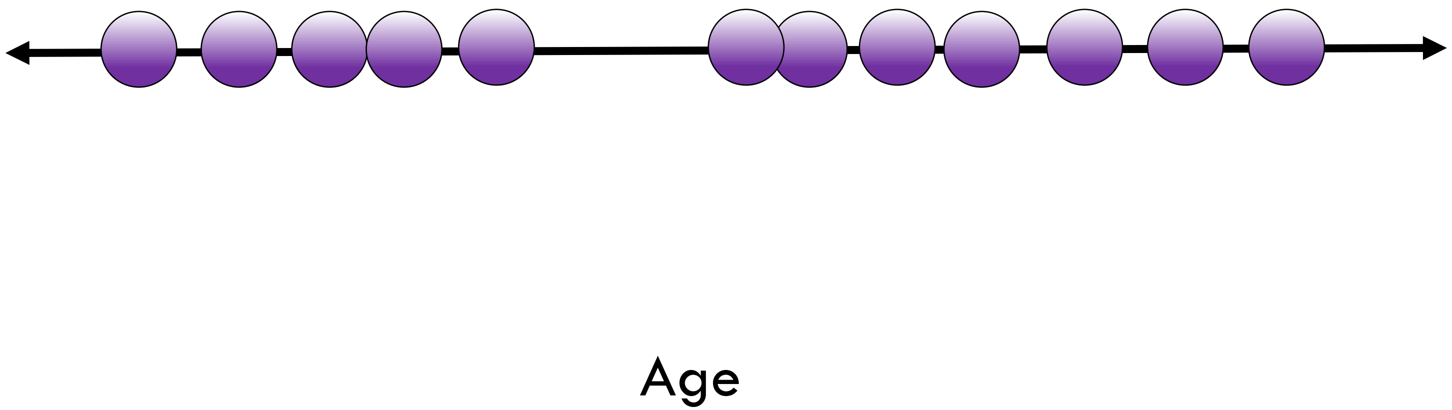
Supervised vs. Unsupervised Learning in Auditing

Feature	Supervised Learning	Unsupervised Learning
Data Requirement	Labeled data (e.g., fraud vs. non-fraud)	Unlabeled data
Common Algorithms	Logistic Regression, Decision Trees, KNN	K-Means, DBSCAN, Isolation Forest
Audit Use Cases	Fraud detection, risk classification	Outlier detection, vendor segmentation
Outcome	Predicts known categories	Discovers unknown patterns or groupings
Goal	Learn from historical labels to predict future	Explore data for anomalies or hidden structure

Introduction to Unsupervised Learning

Users of a web application:

One feature (age)

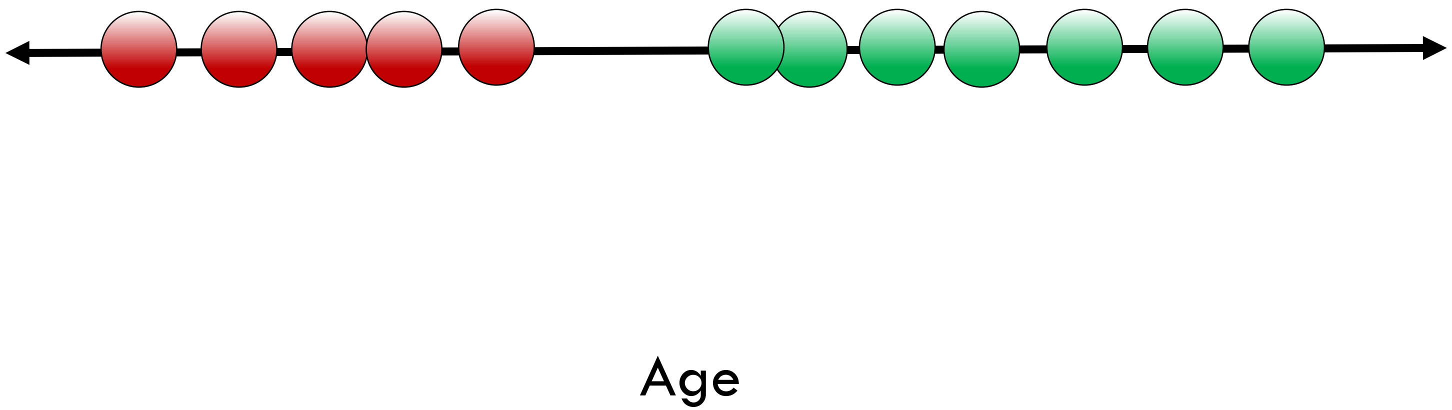


Introduction to Unsupervised Learning

Users of a web application:

One feature (age)

Two clusters

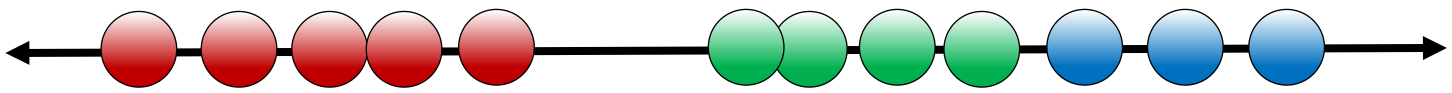


Introduction to Unsupervised Learning

Users of a web application:

One feature (age)

Three clusters



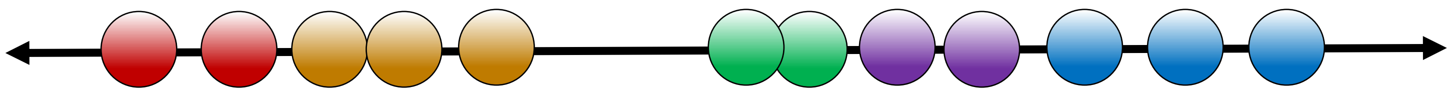
Age

Introduction to Unsupervised Learning

Users of a web application:

One feature (age)

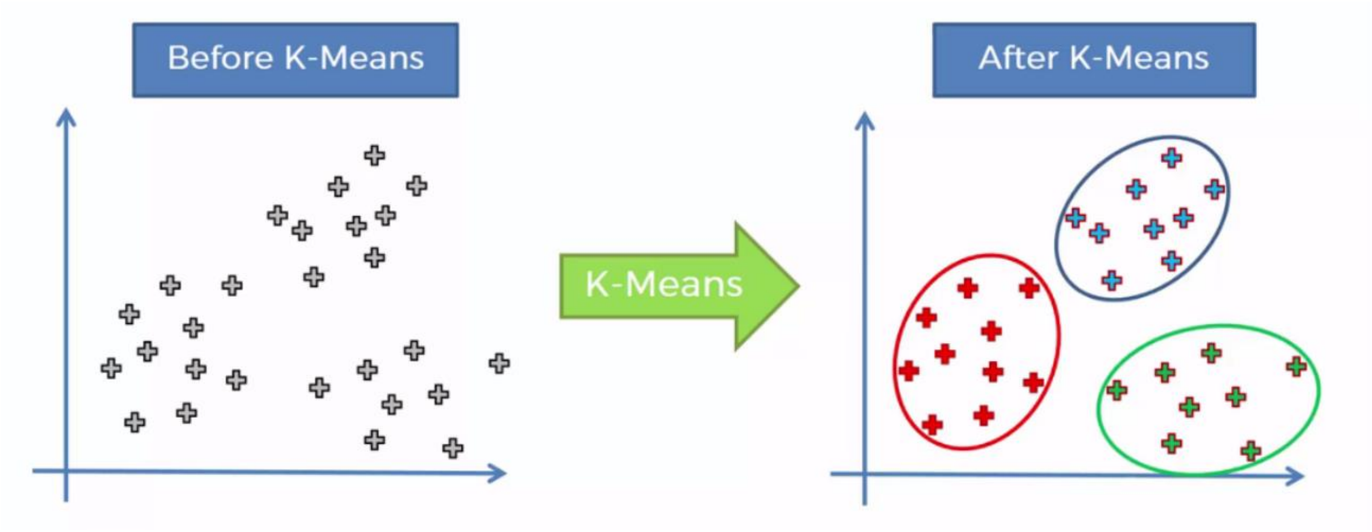
Five clusters



Age

K-Means

- K-means clustering is unsupervised learning for unlabeled data.
- It identifies groups within the data.
- 'k' signifies the number of groups.



K-Means

- **Initialization** - Start by randomly selecting 'k' initial centroids (cluster centers) from the dataset.

- **Assignment** - Assign each data point to the nearest centroid based on a distance metric (usually Euclidean distance).

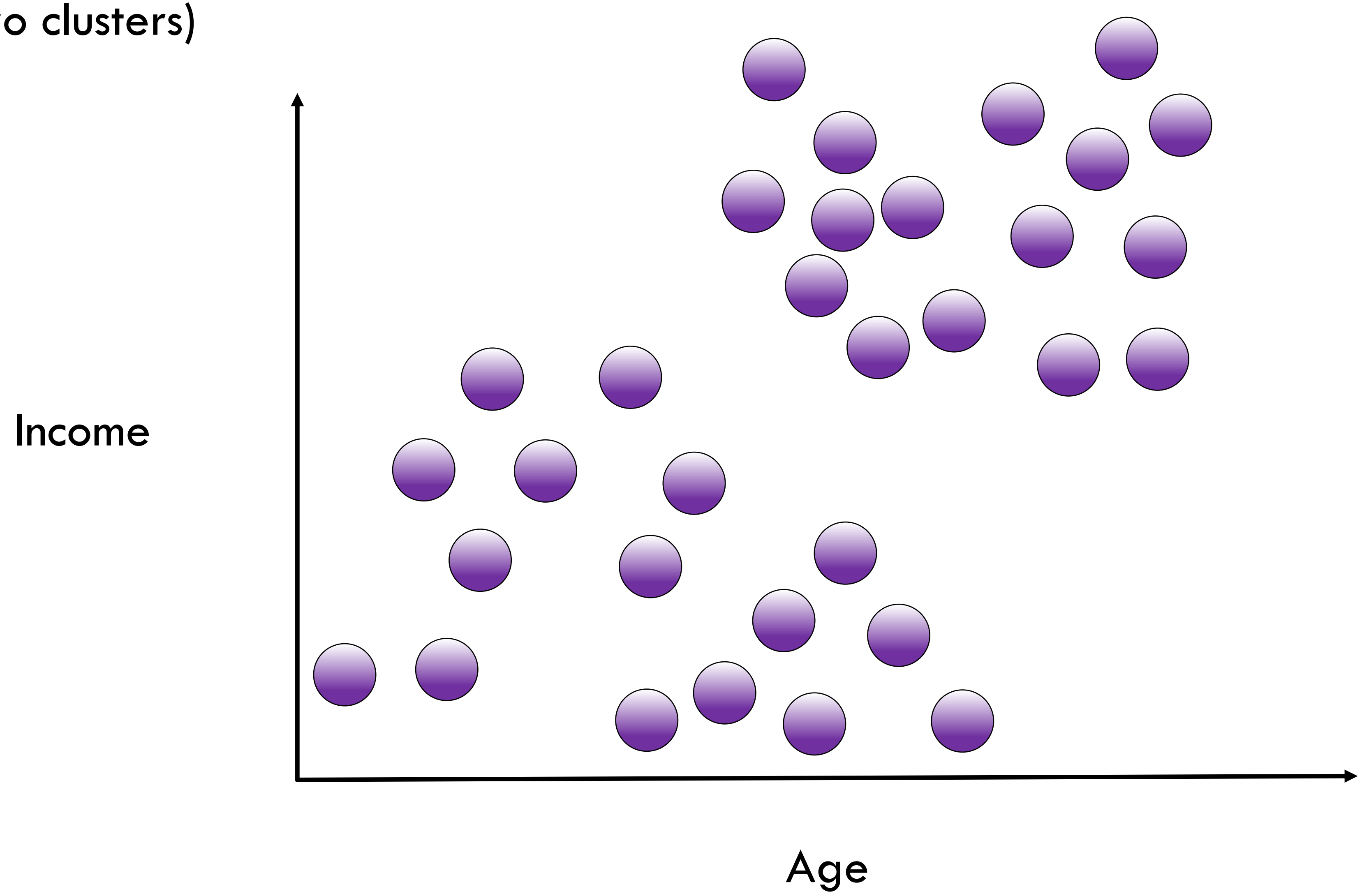
- **Update centroids** - Recalculate the centroids of the clusters by taking the mean of all data points assigned to each cluster.

- **Repeat** - Iterate steps 2 and 3 until convergence, meaning the centroids no longer change significantly or a predefined number of iterations is reached.

- **Convergence** - When centroids stabilize or after a set number of iterations, the algorithm stops, and the final clusters are formed.

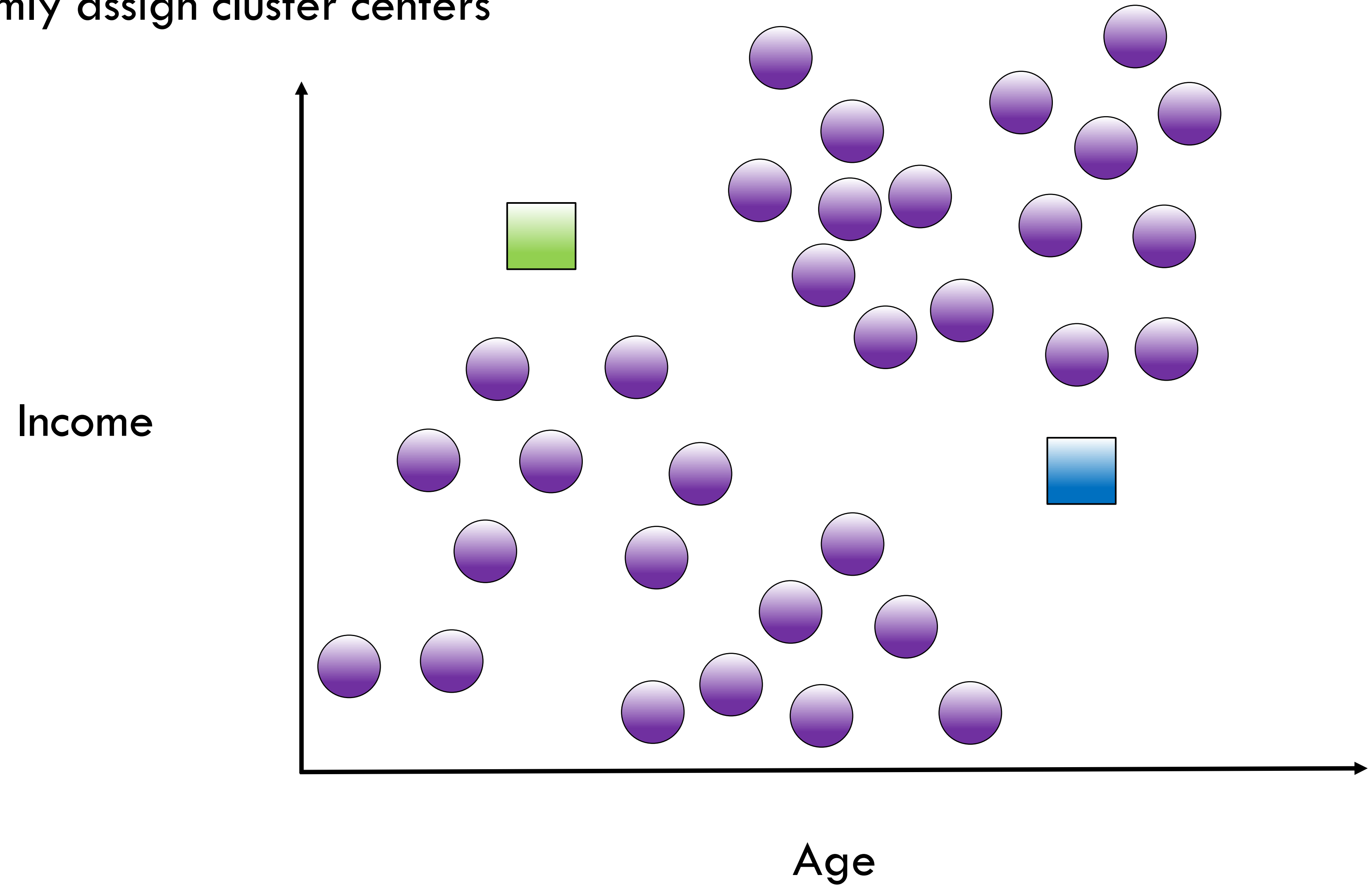
K-Means Algorithm

K = 2 (find two clusters)



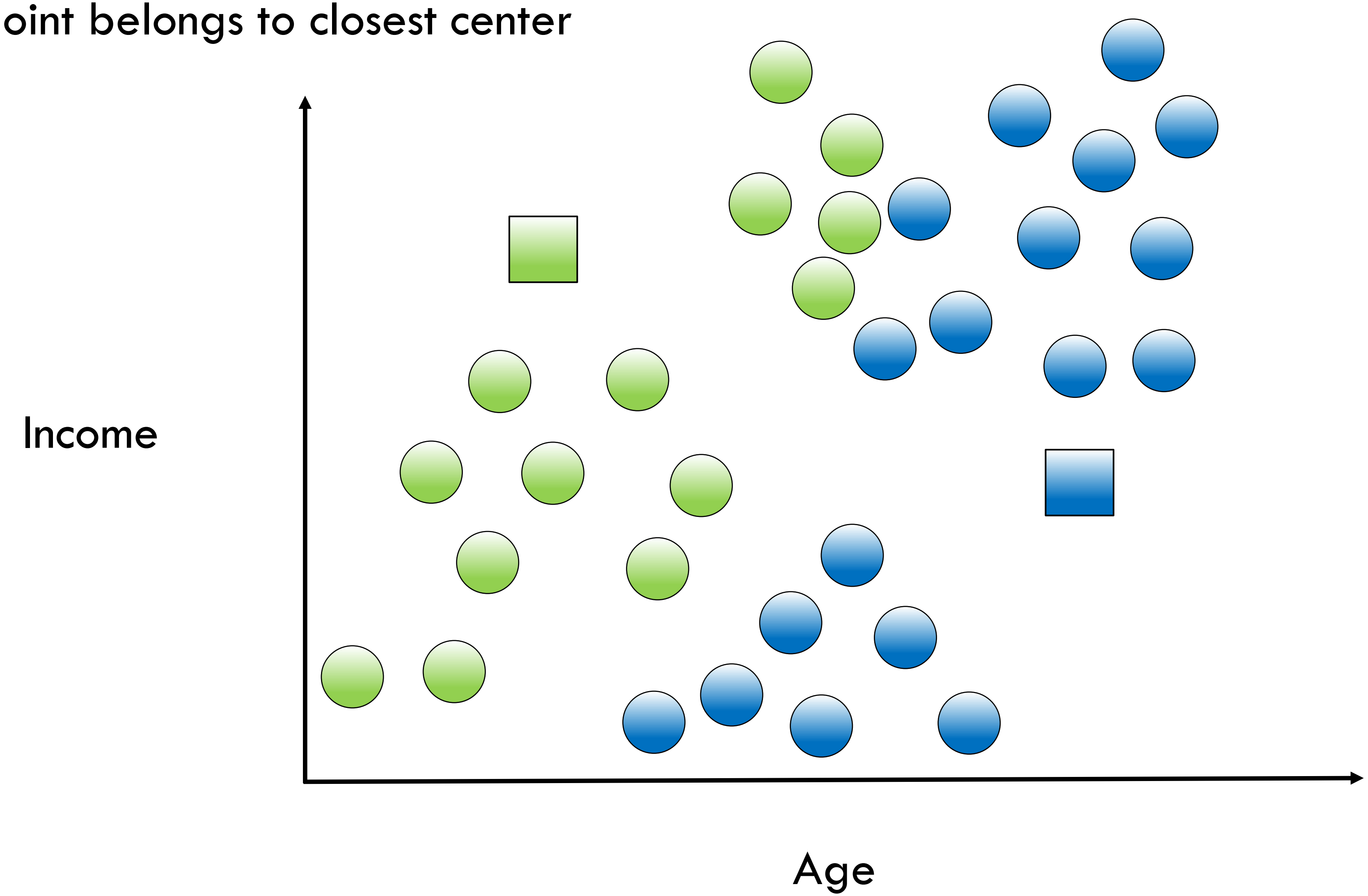
K-Means Algorithm

K = 2, Randomly assign cluster centers



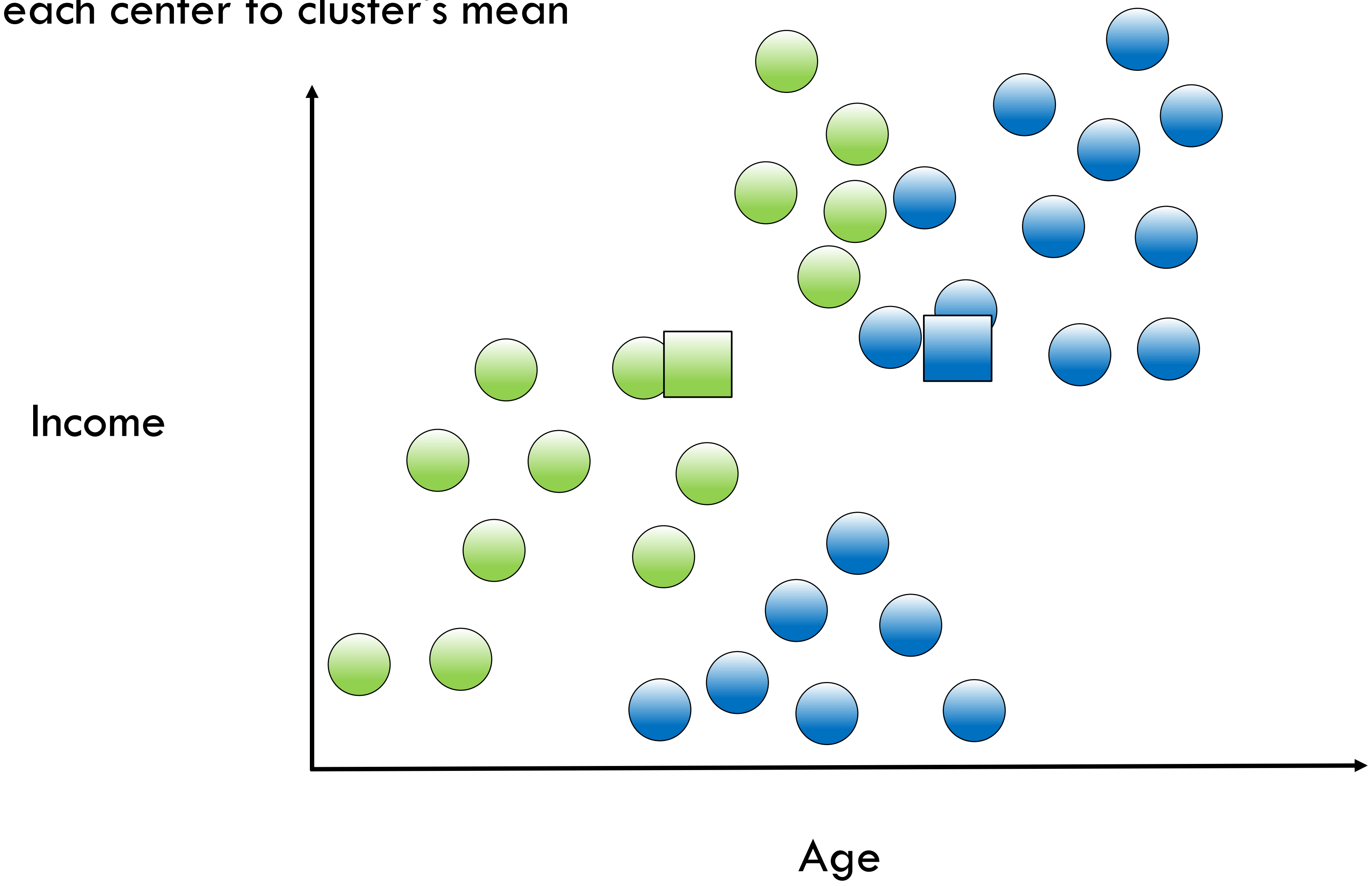
K-Means Algorithm

K = 2, Each point belongs to closest center



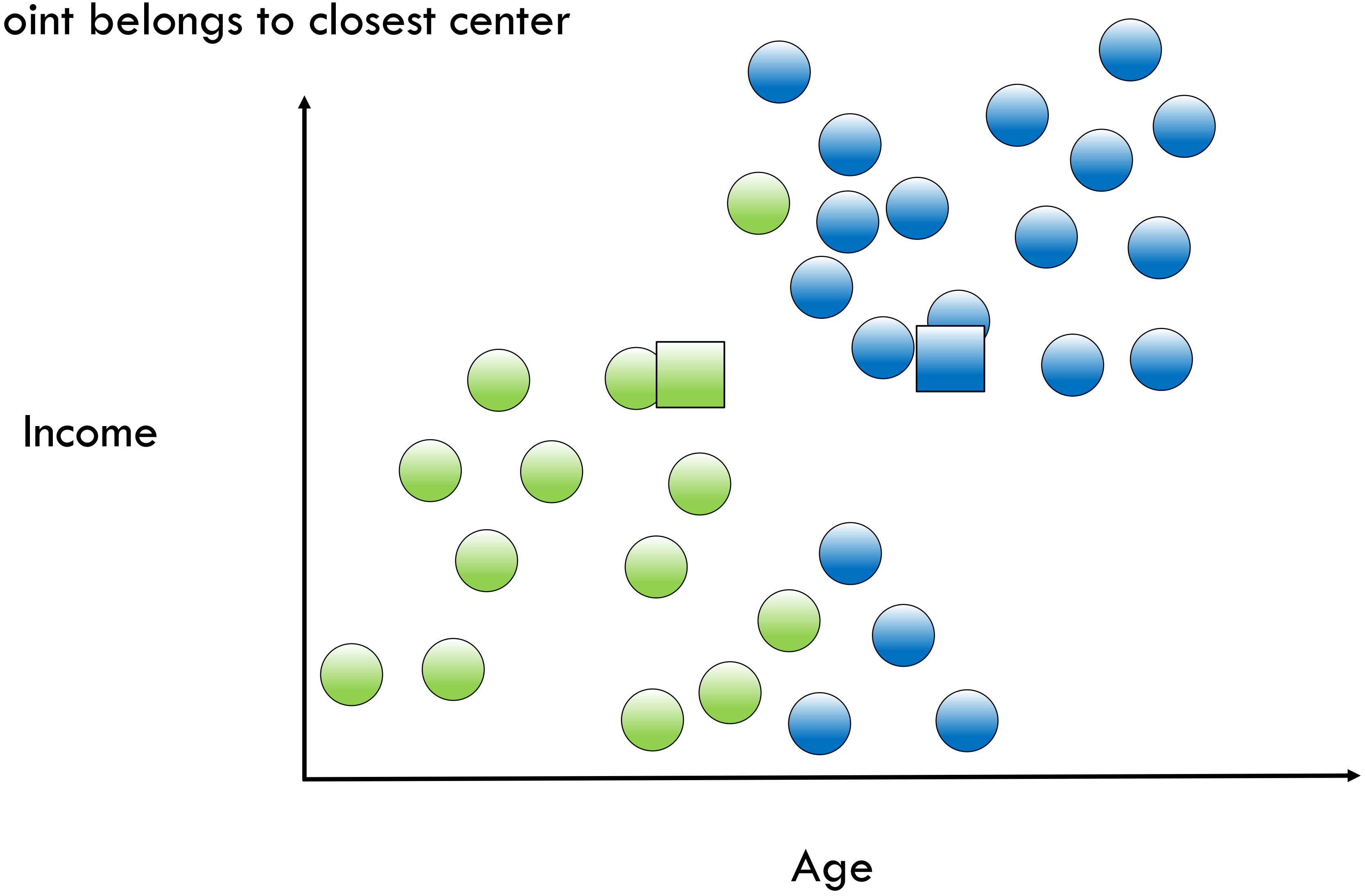
K-Means Algorithm

K = 2, Move each center to cluster's mean



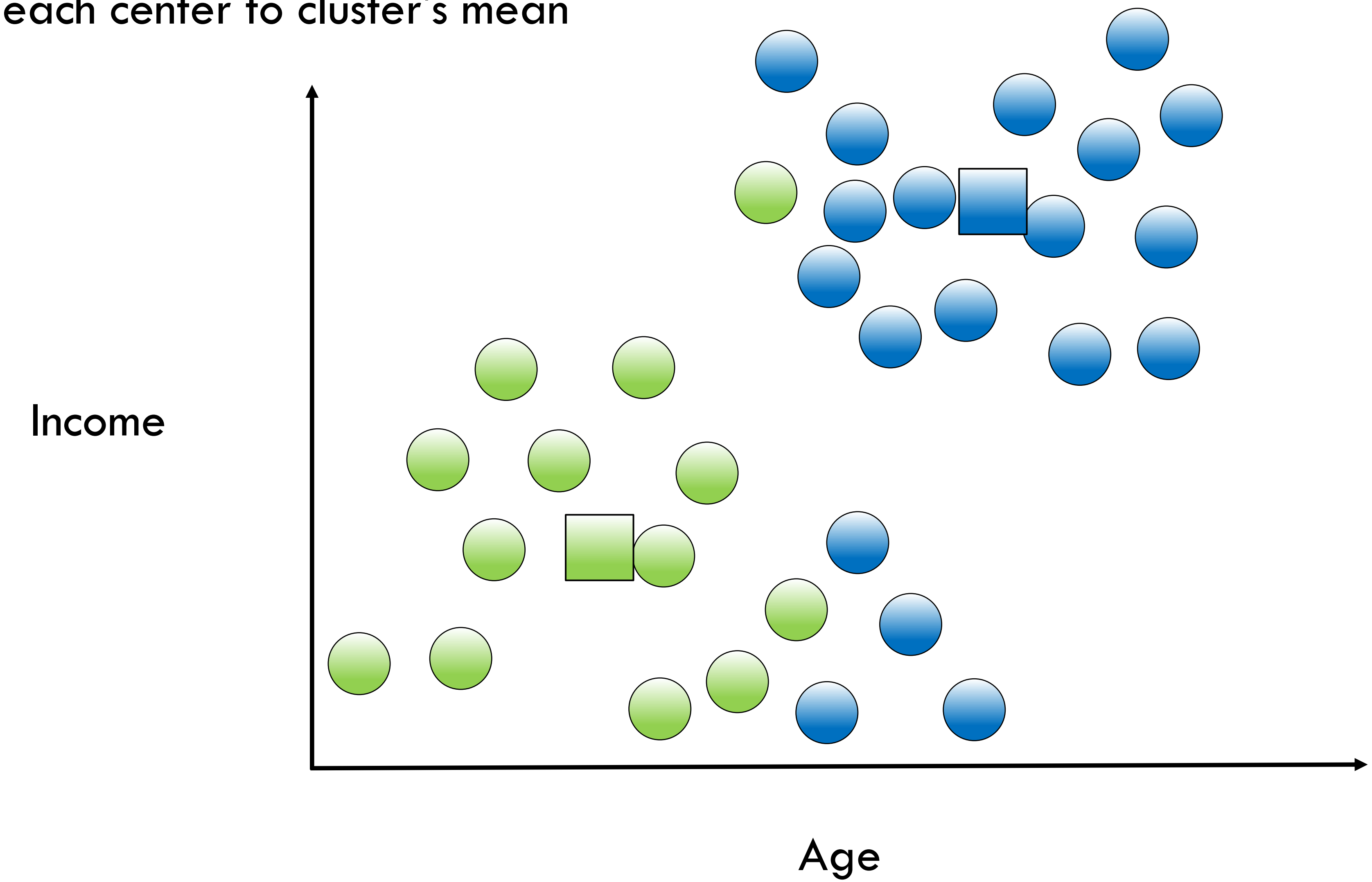
K-Means Algorithm

K = 2, Each point belongs to closest center



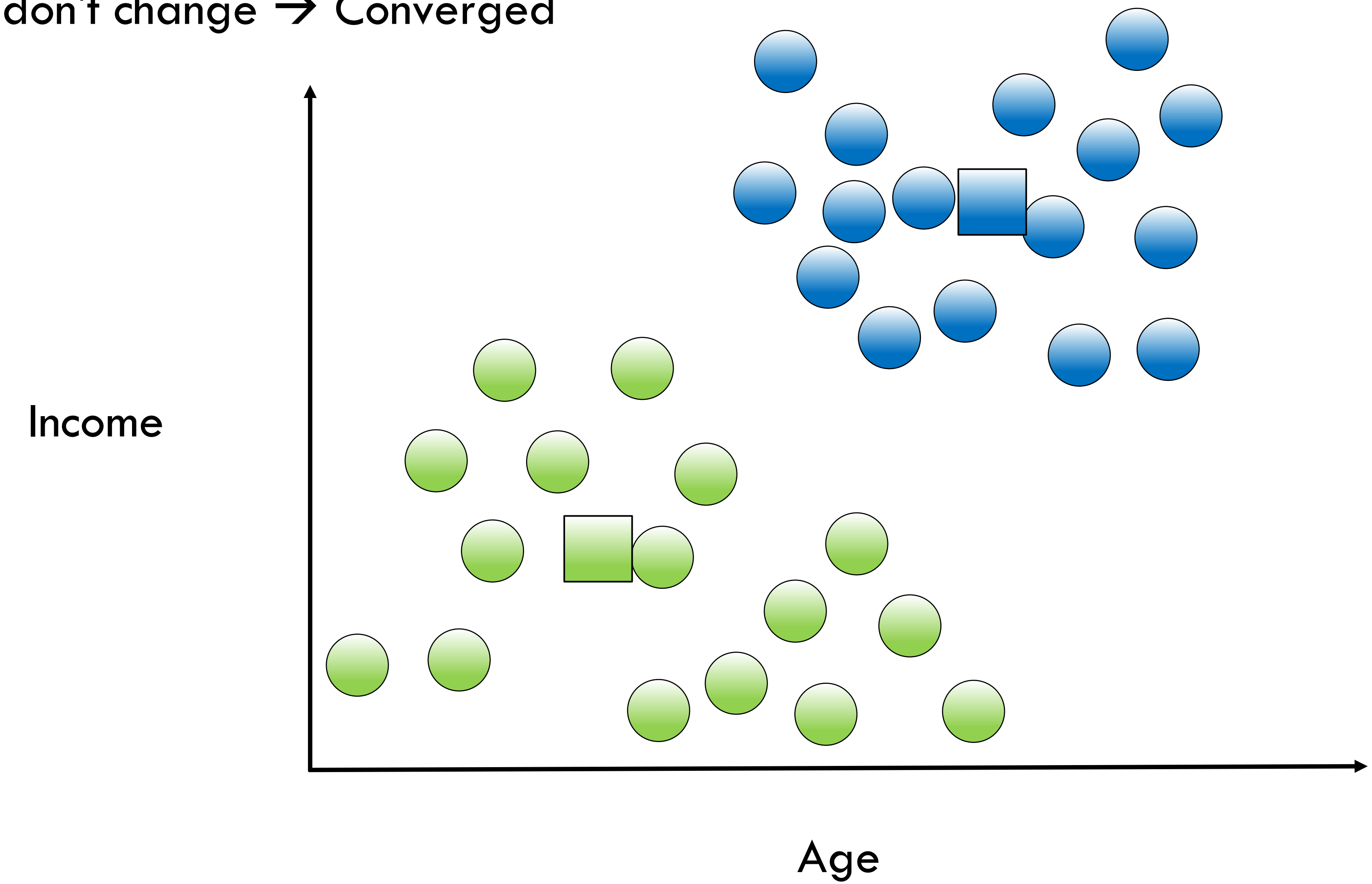
K-Means Algorithm

K = 2, Move each center to cluster's mean



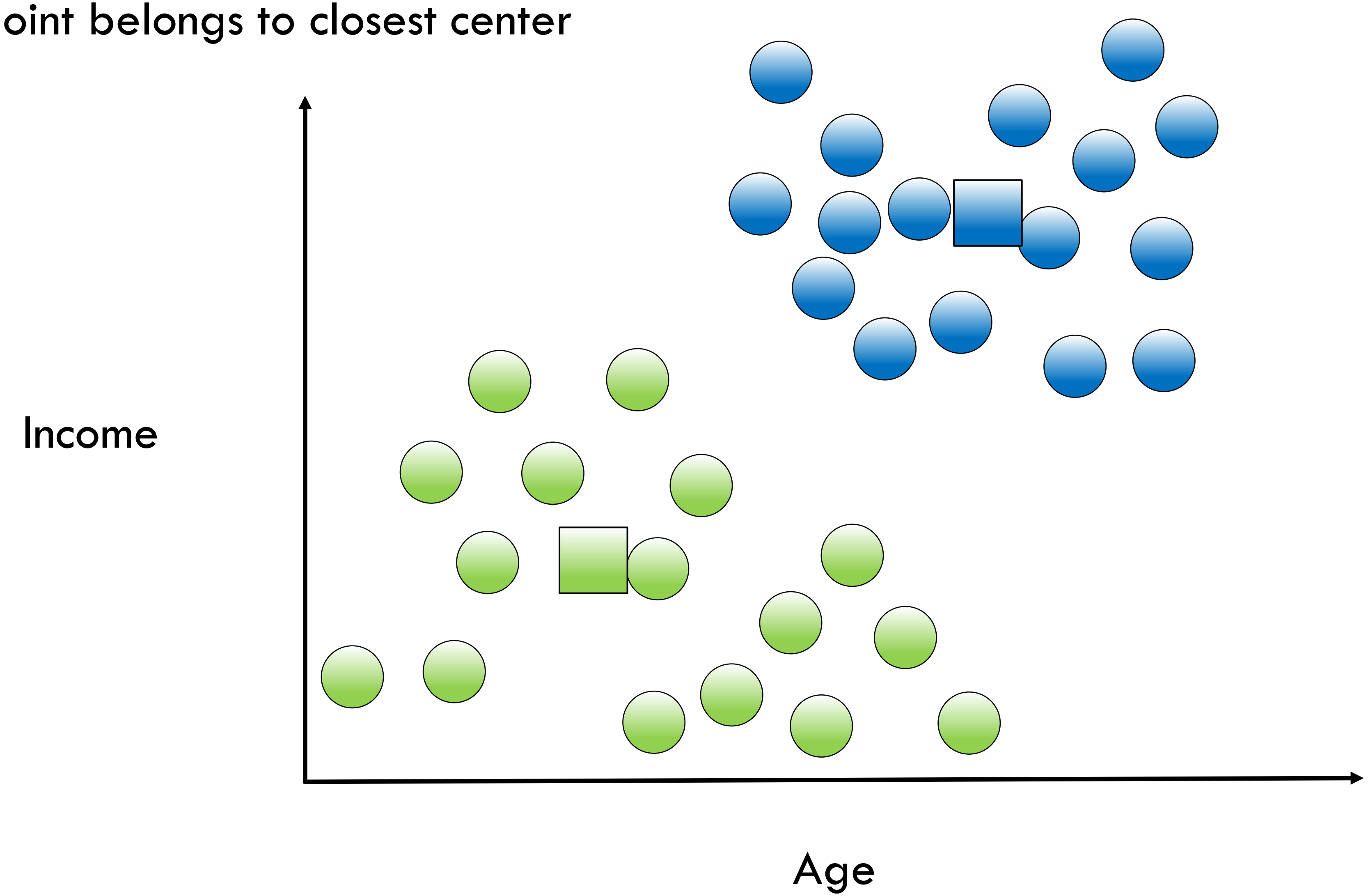
K-Means Algorithm

K = 2, Points don't change → Converged



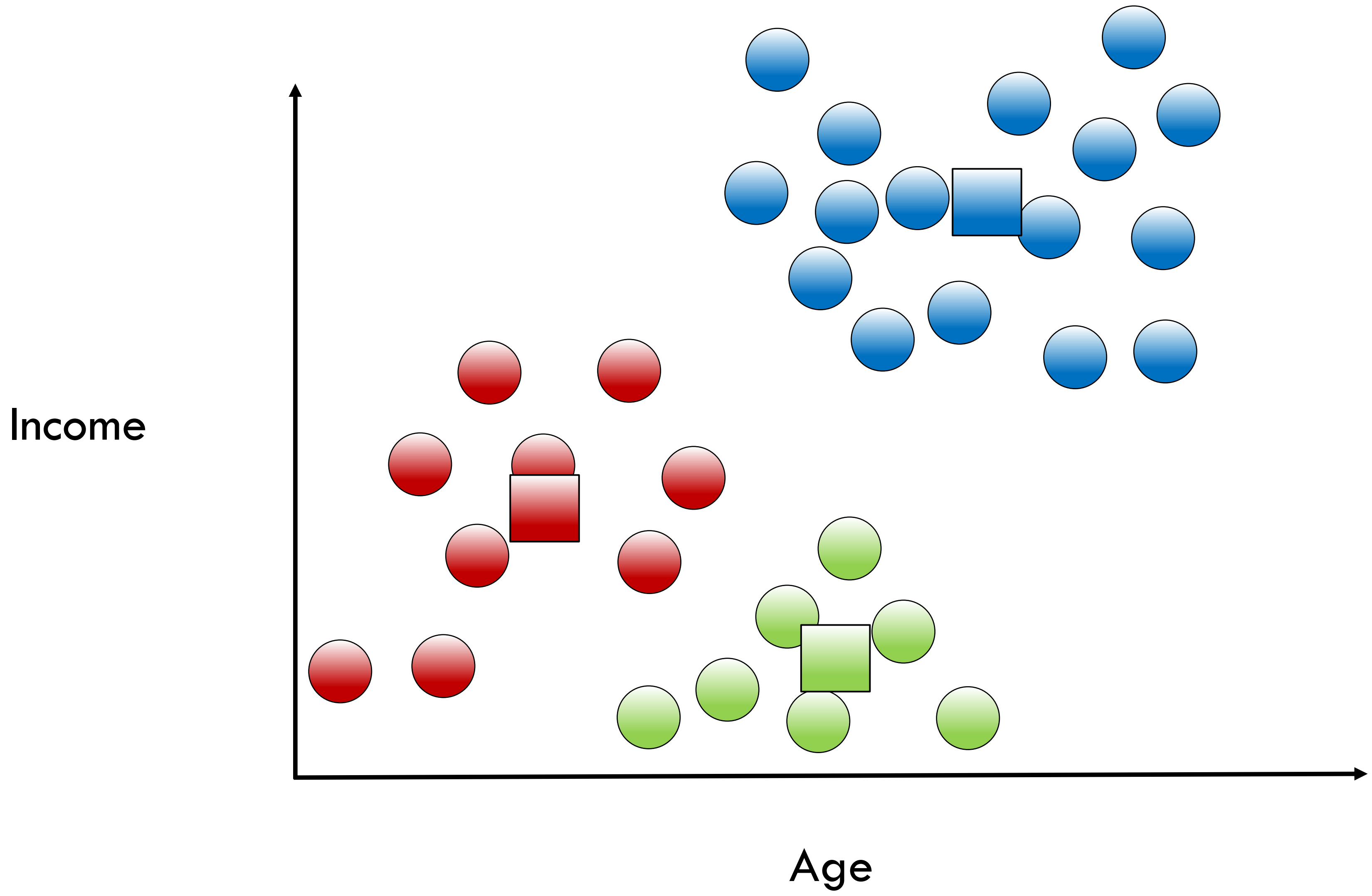
K-Means Algorithm

K = 2, Each point belongs to closest center



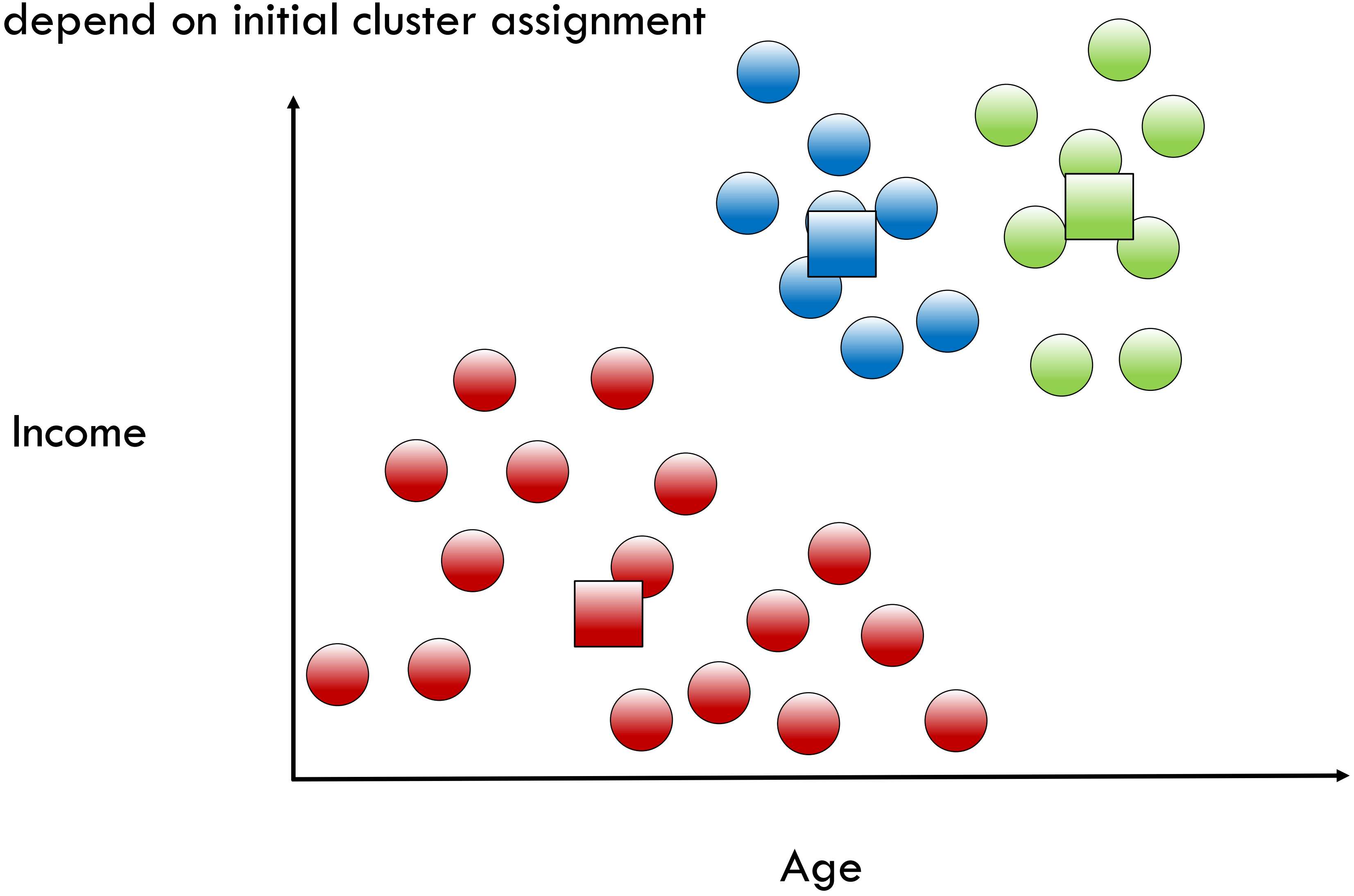
K-Means Algorithm

K = 3

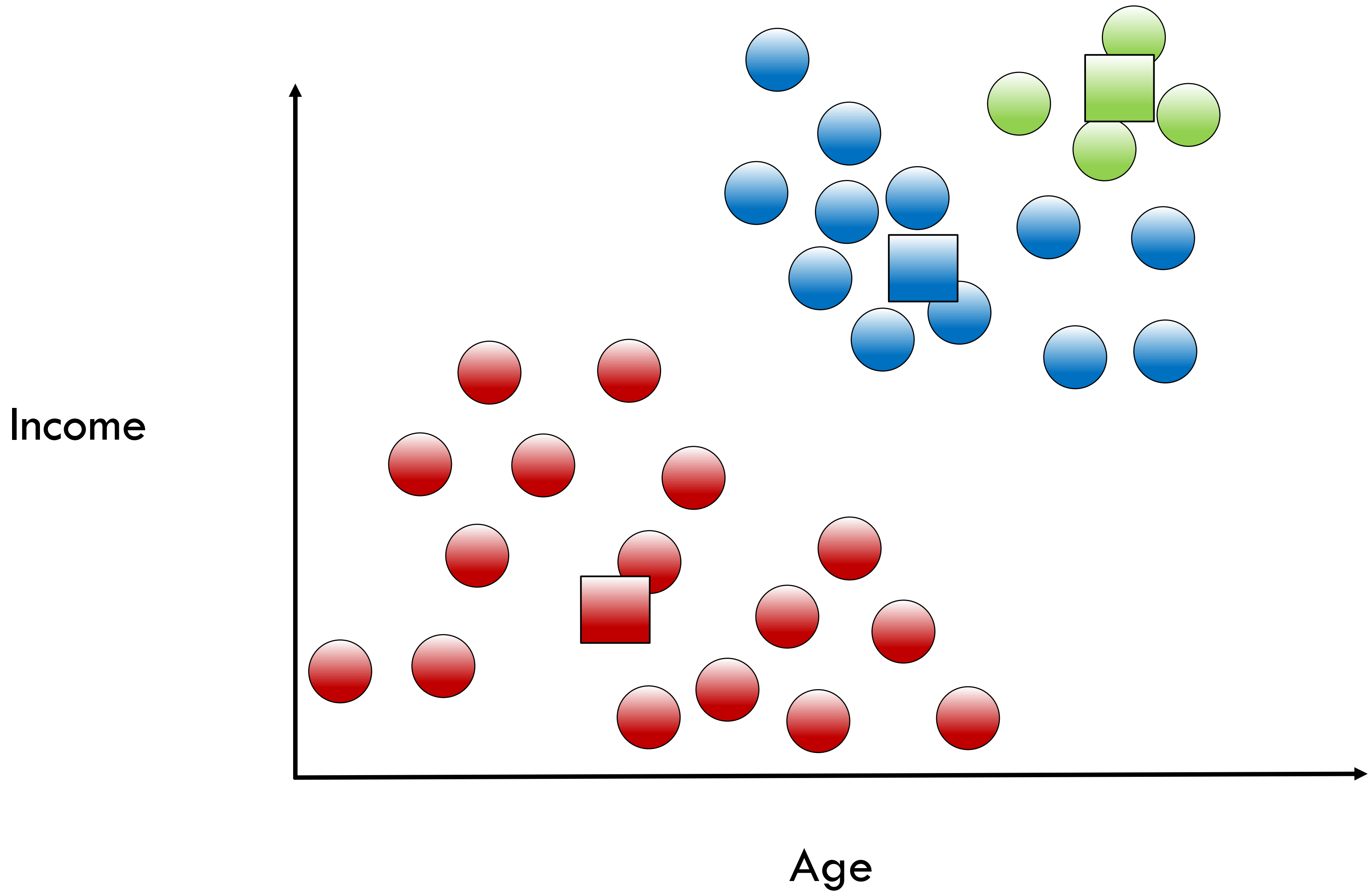


K-Means Algorithm

K = 3, Results depend on initial cluster assignment



Which Model is the Right One?



Which Model is the Right One?

- **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

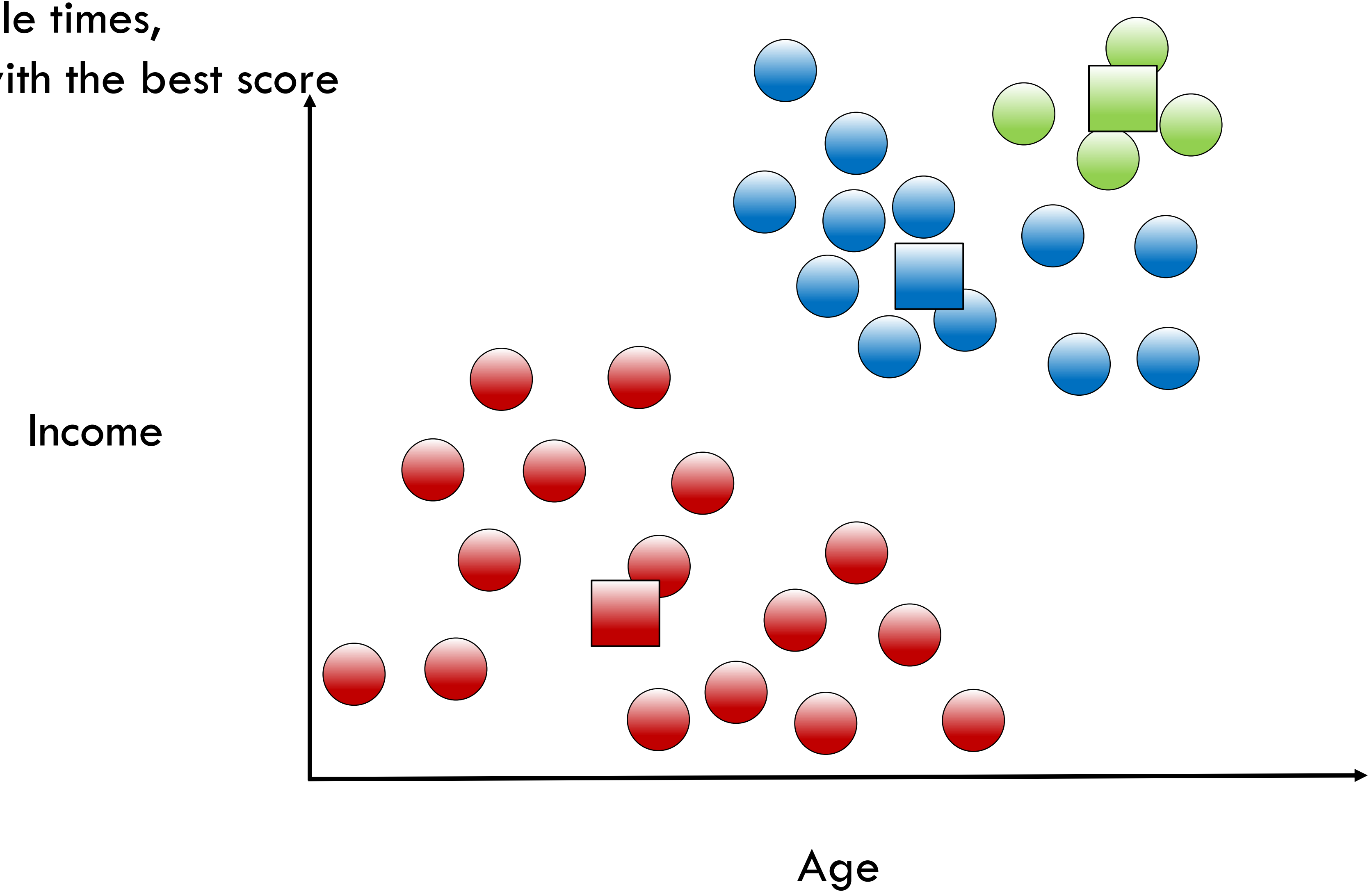
- Other metrics can also be used

Inertia

- Inertia measures how well a dataset was clustered by K-Means.
- It is calculated by measuring the distance between each data point and its centroid, squaring this distance, and summing these squares across one cluster.
- A good model is one with low inertia AND a low number of clusters (K).

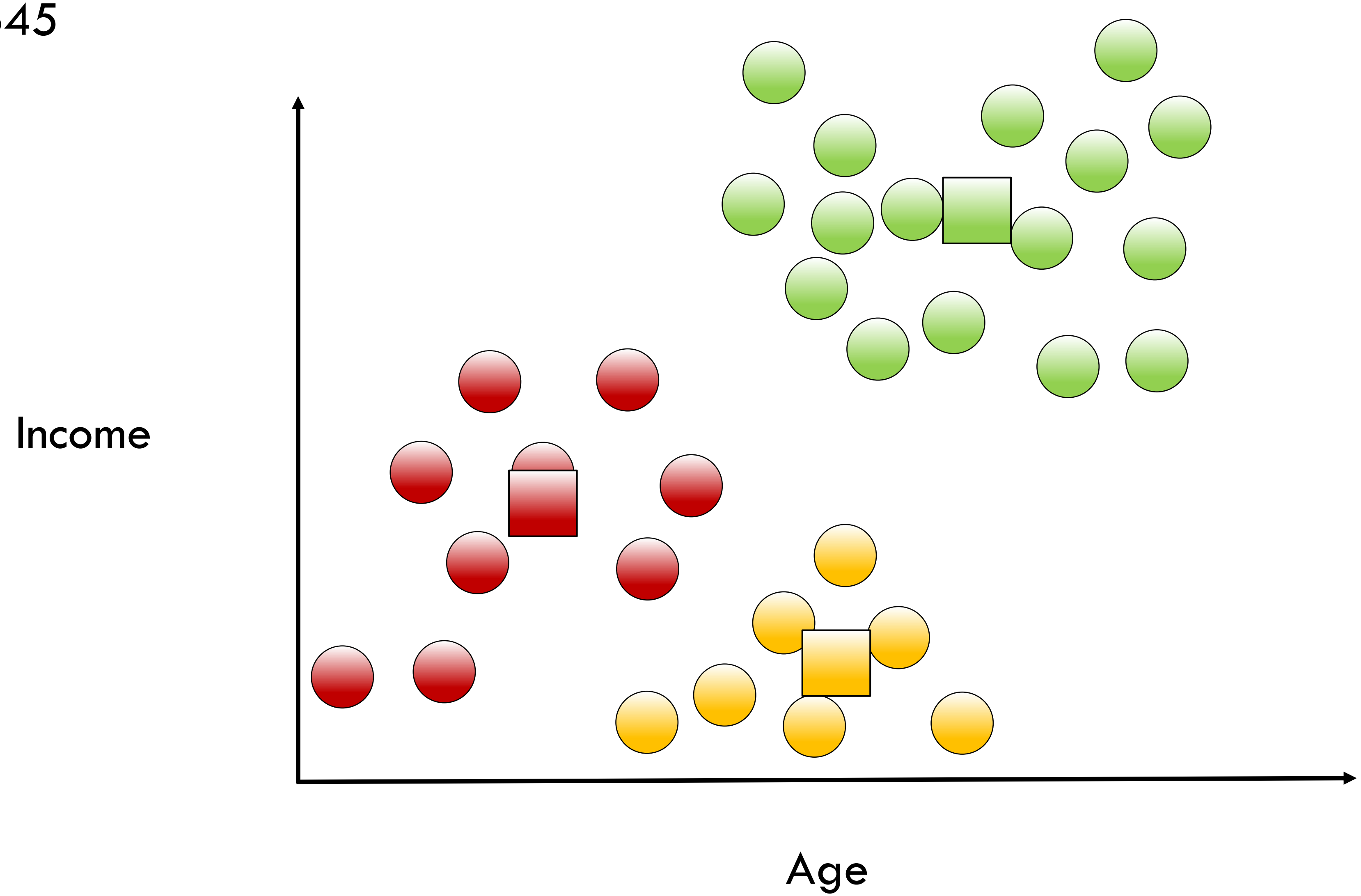
Which Model is the Right One?

Initiate multiple times,
take model with the best score



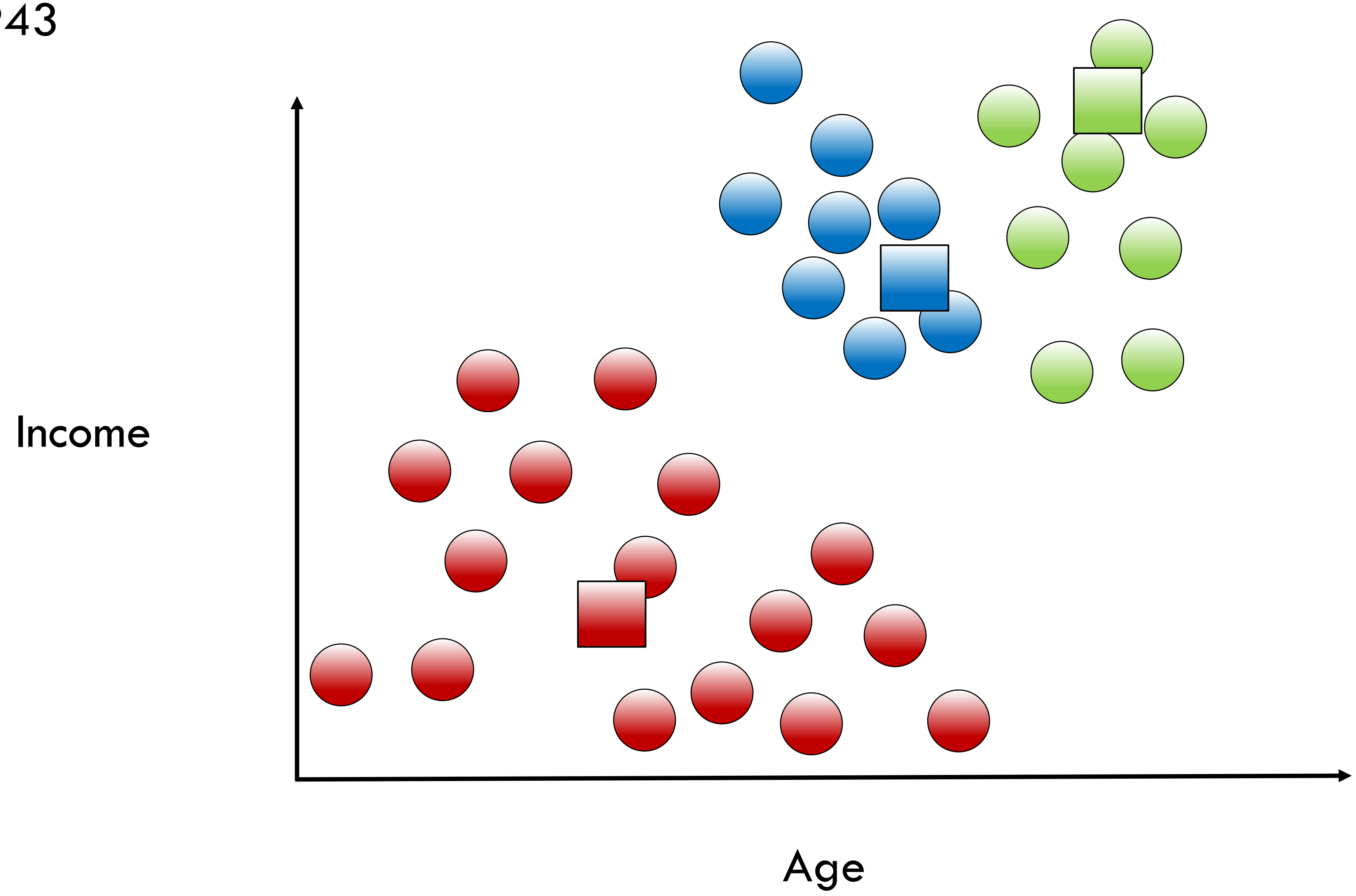
Which Model is the Right One?

Inertia = 12.645



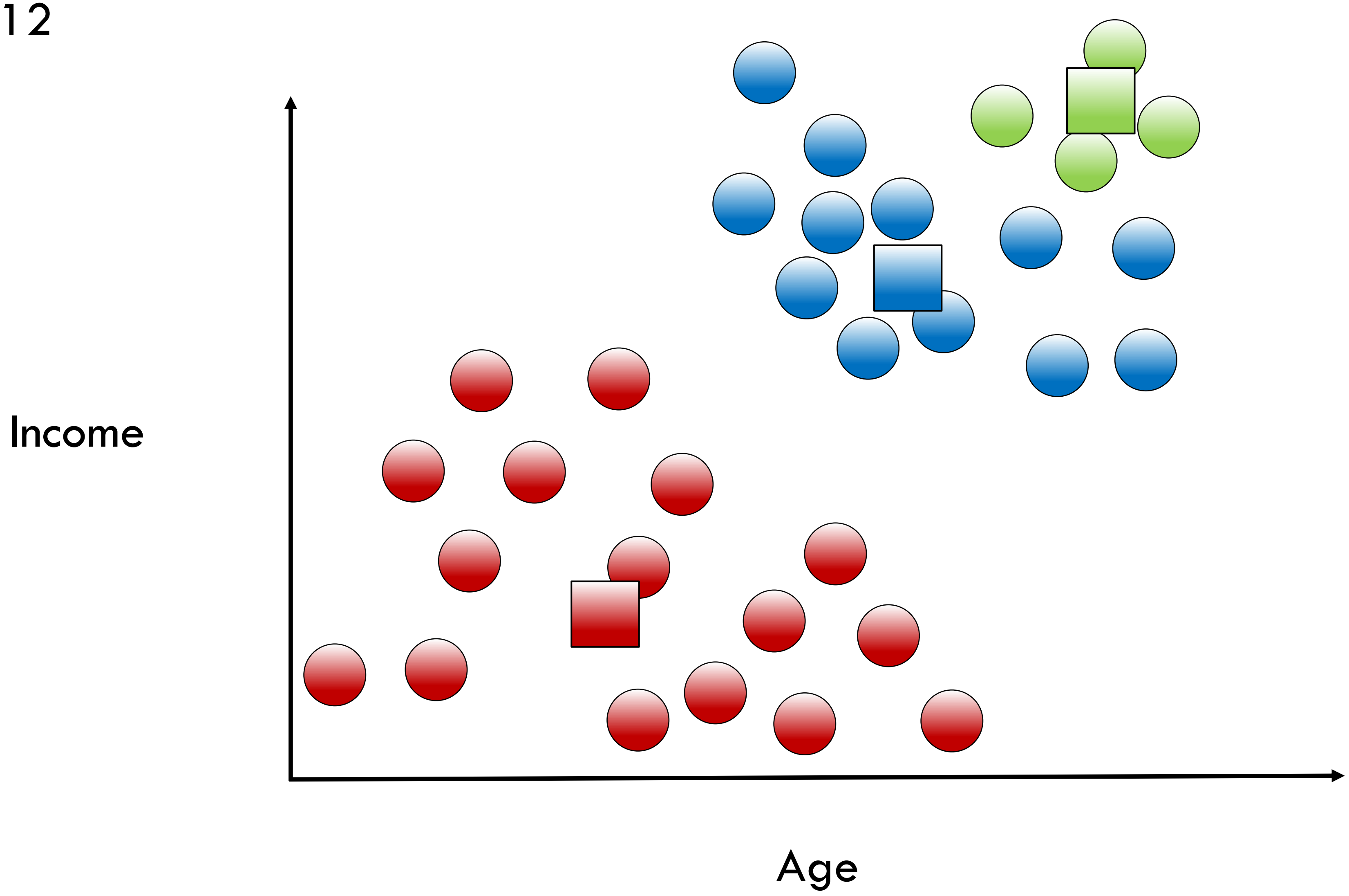
Which Model is the Right One?

Inertia = 12.943

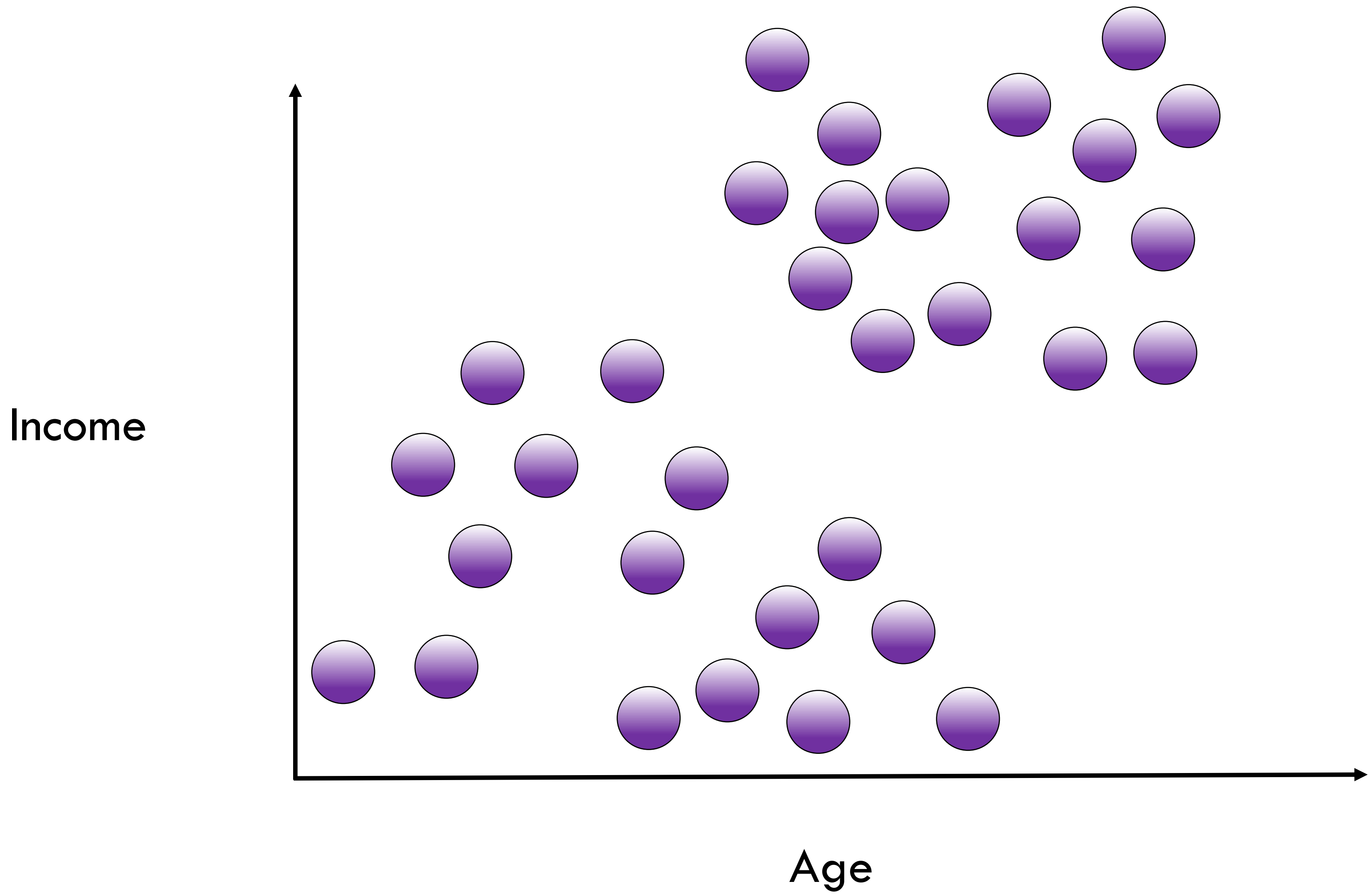


Which Model is the Right One?

Inertia = 13.112

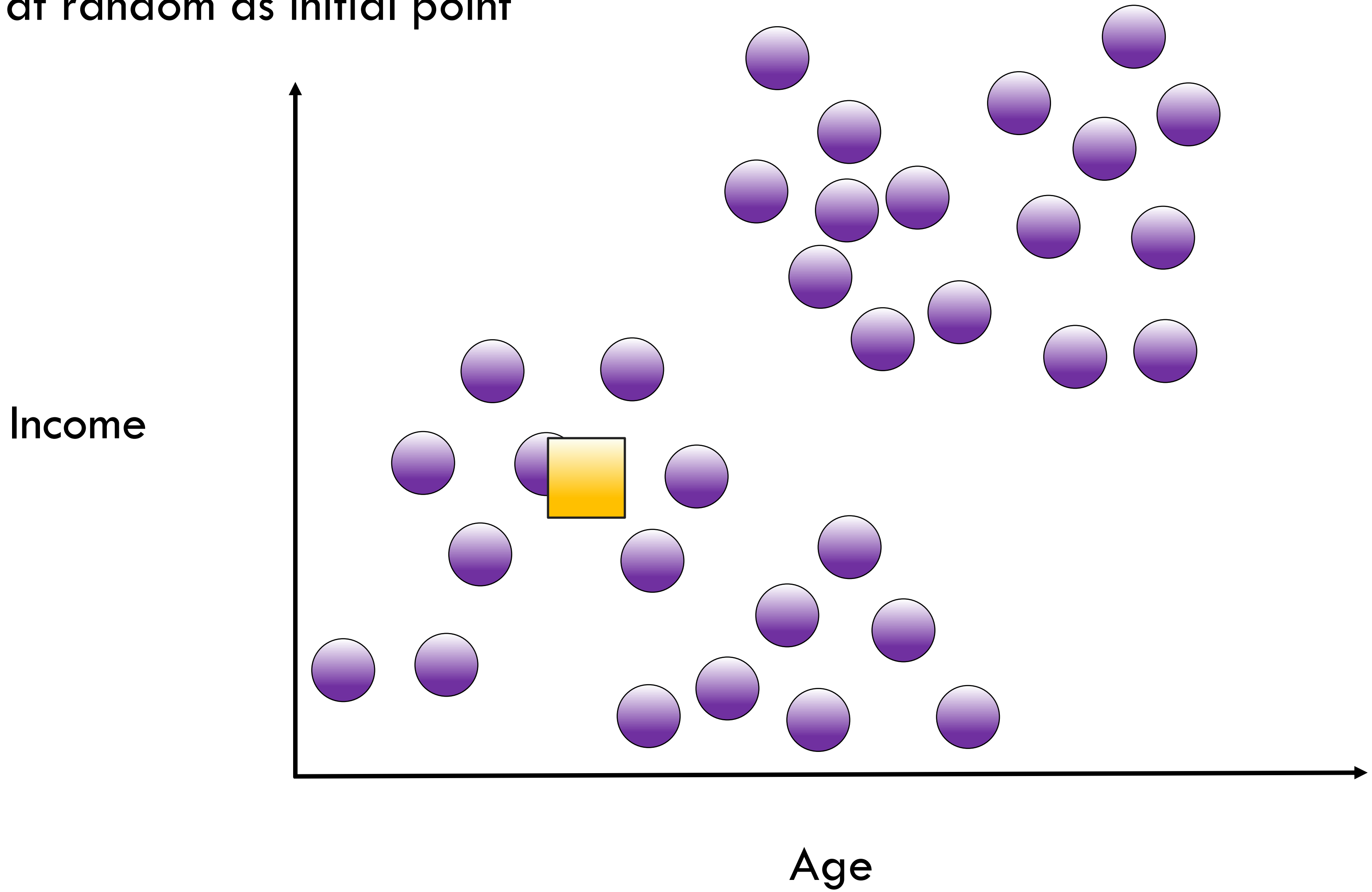


Smarter Initialization of K-Means Clusters



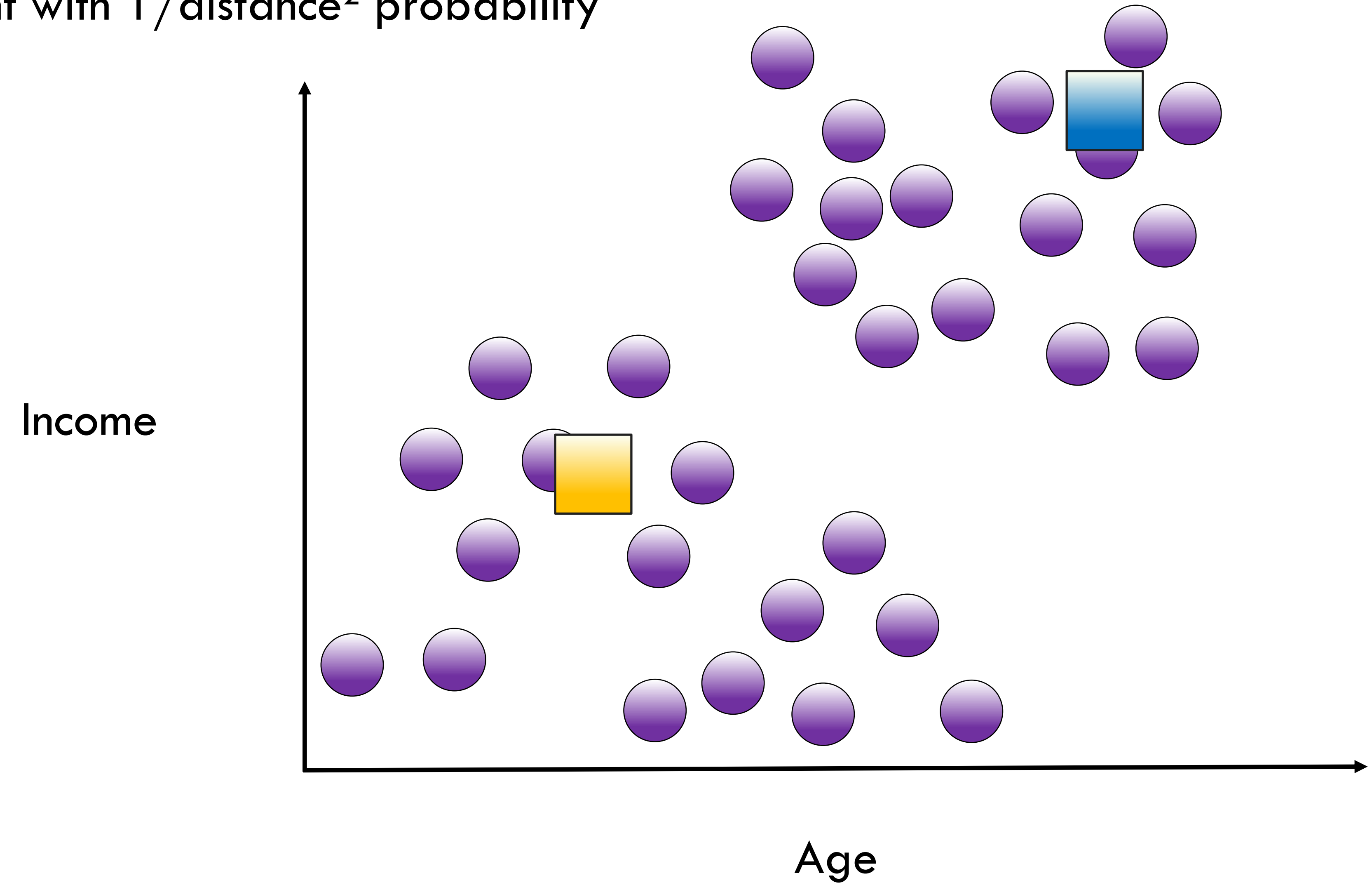
Smarter Initialization of K-Means Clusters

Pick one point at random as initial point



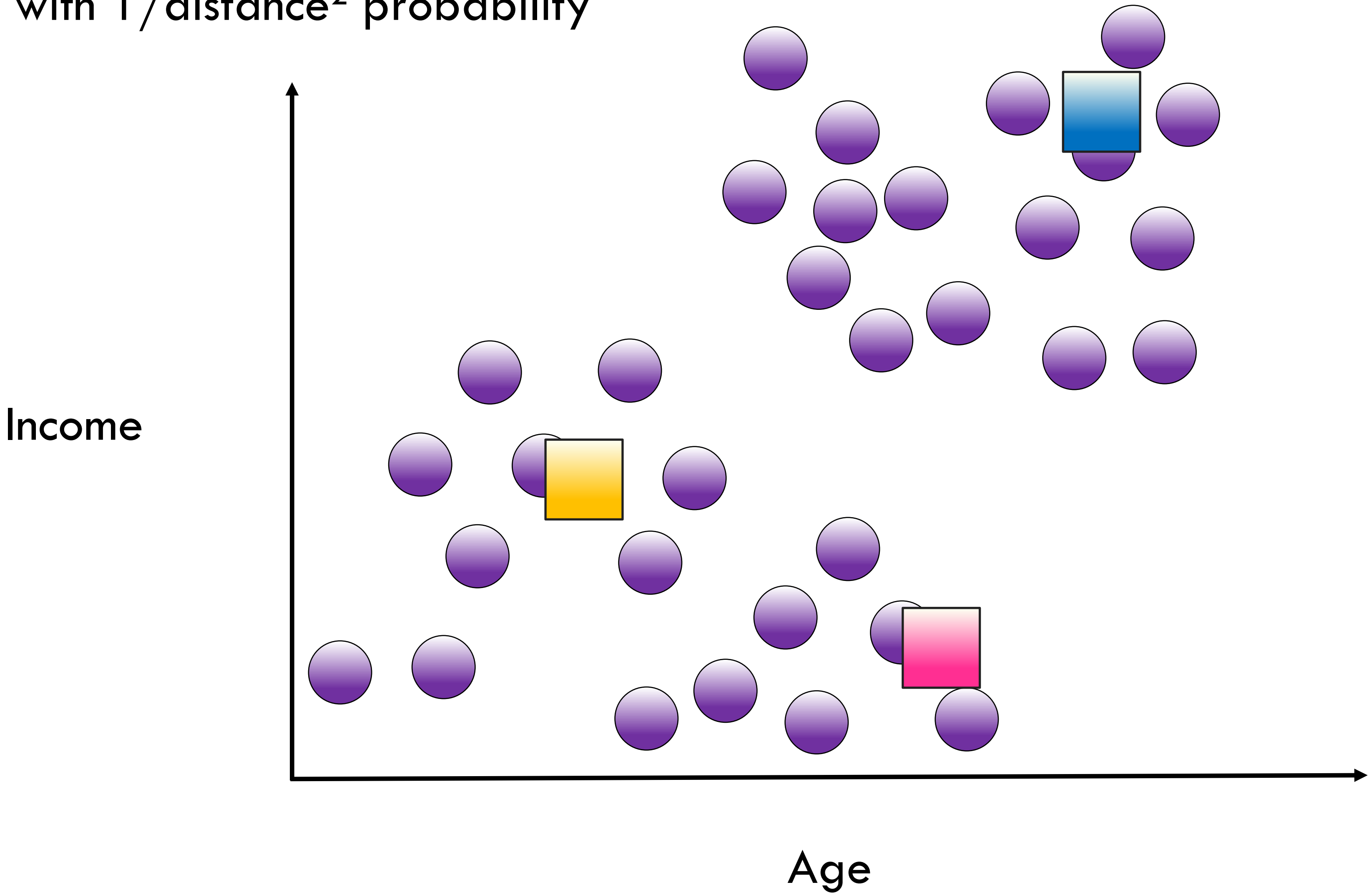
Smarter Initialization of K-Means Clusters

Pick next point with $1/\text{distance}^2$ probability



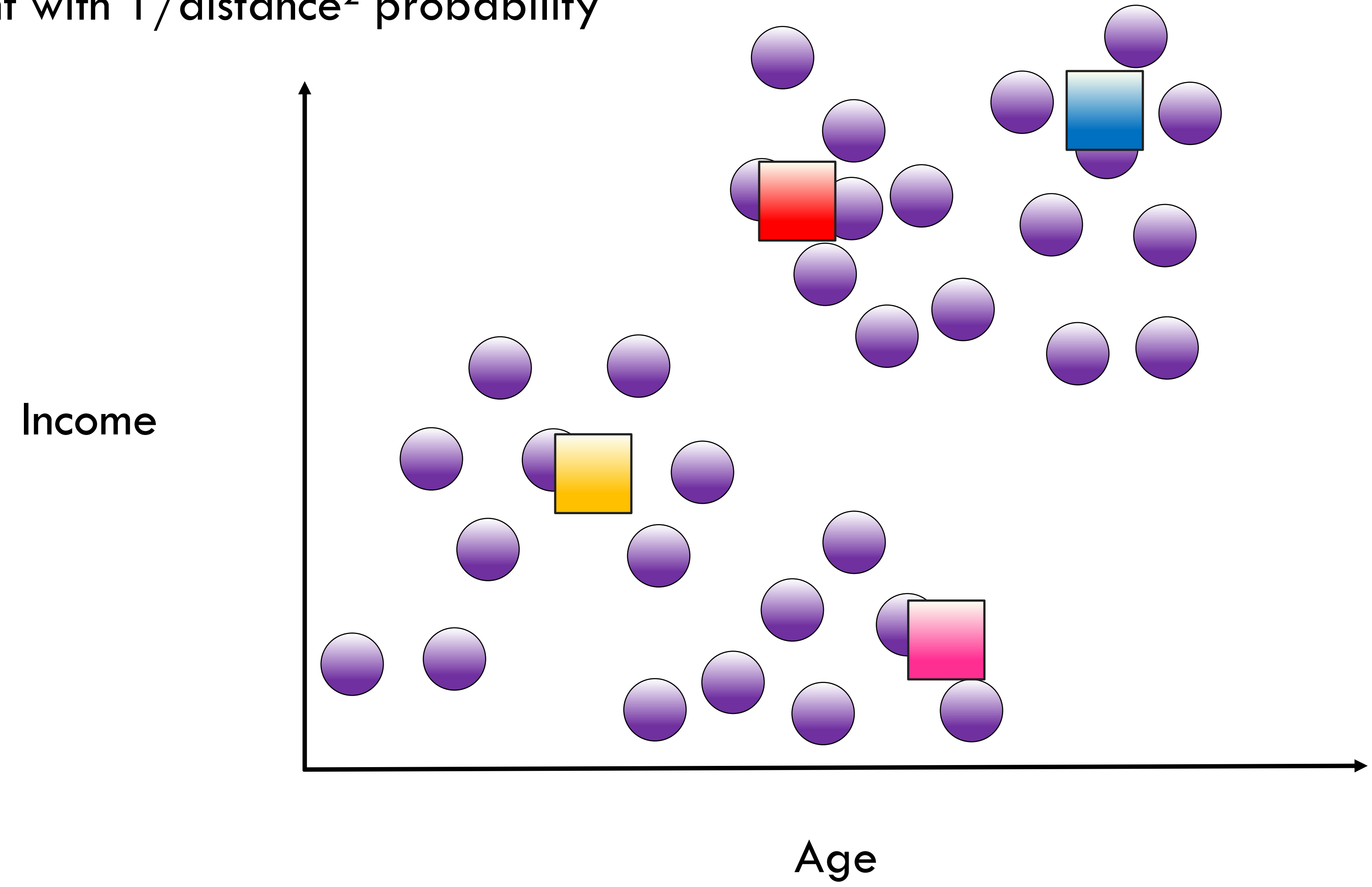
Smarter Initialization of K-Means Clusters

Pick next point with 1/distance^2 probability



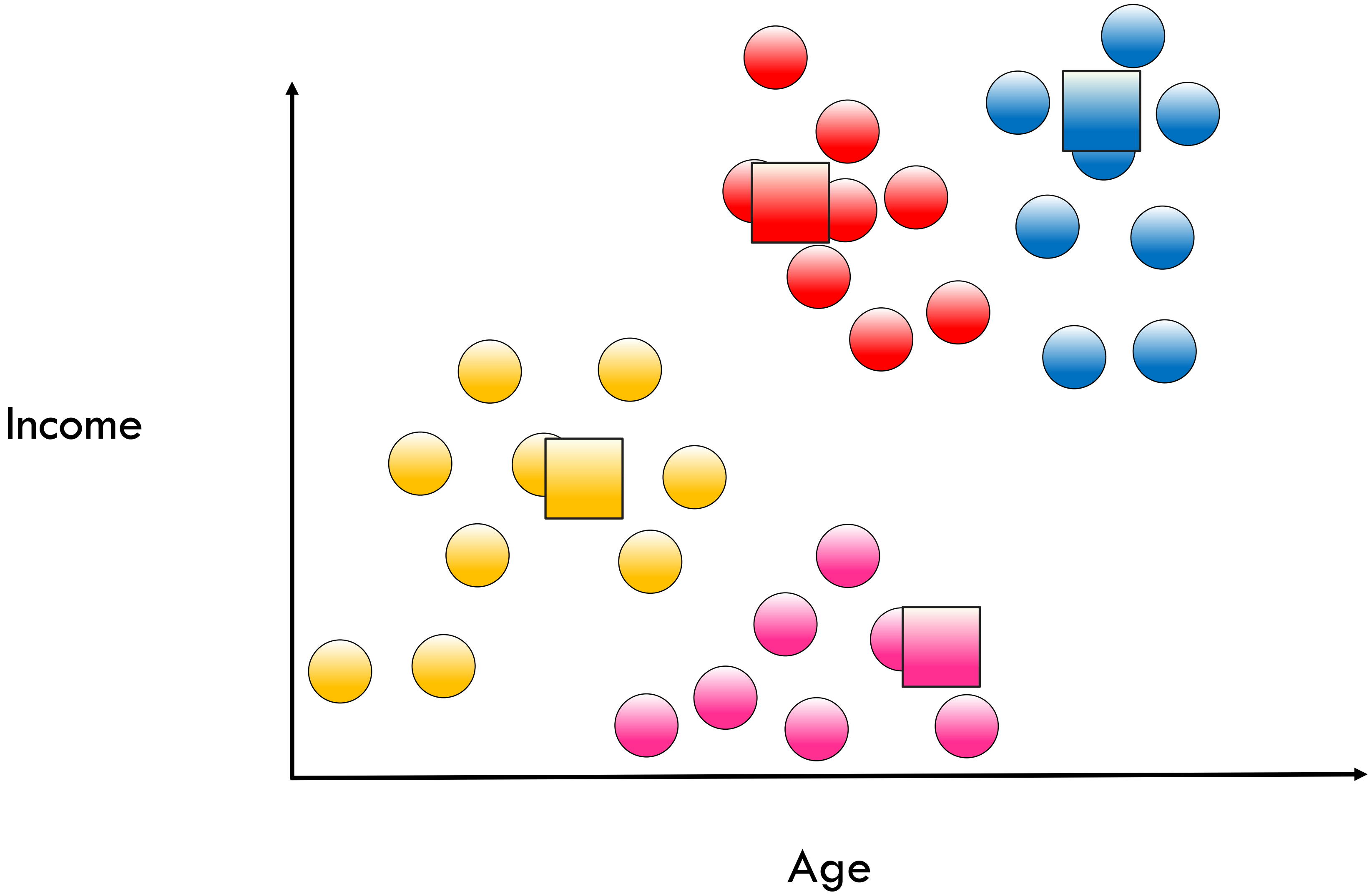
Smarter Initialization of K-Means Clusters

Pick next point with $1/\text{distance}^2$ probability



Smarter Initialization of K-Means Clusters

Assign clusters



Choosing the Right Number of Clusters

Choosing the Right Number of Clusters

- Sometimes the question has a K

Choosing the Right Number of Clusters

- Sometimes the question has a K

Choosing the Right Number of Clusters

- Sometimes the question has a K
- Clustering similar jobs on 4 CPU cores (K=4)

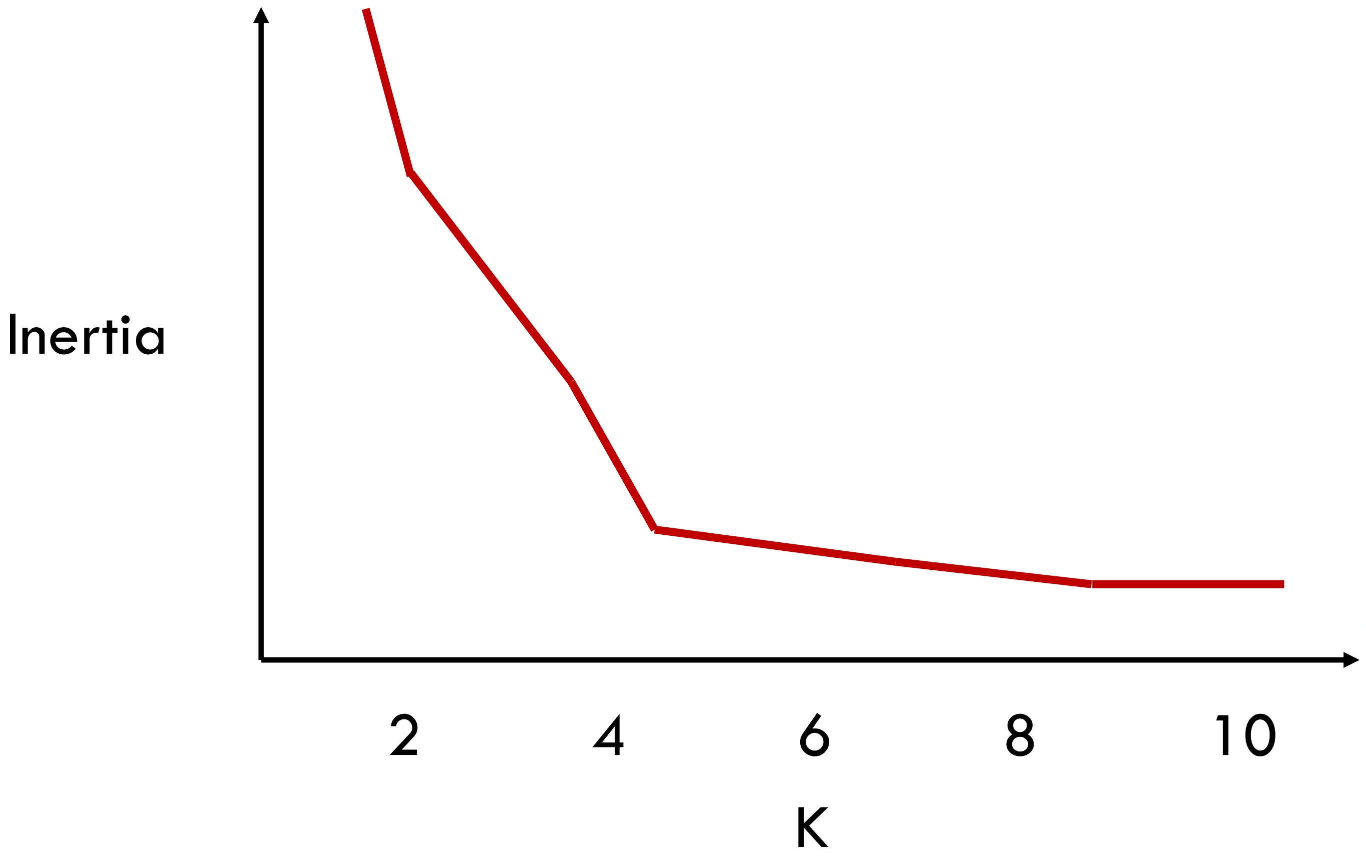
Choosing the Right Number of Clusters

- 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)

Choosing the 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)

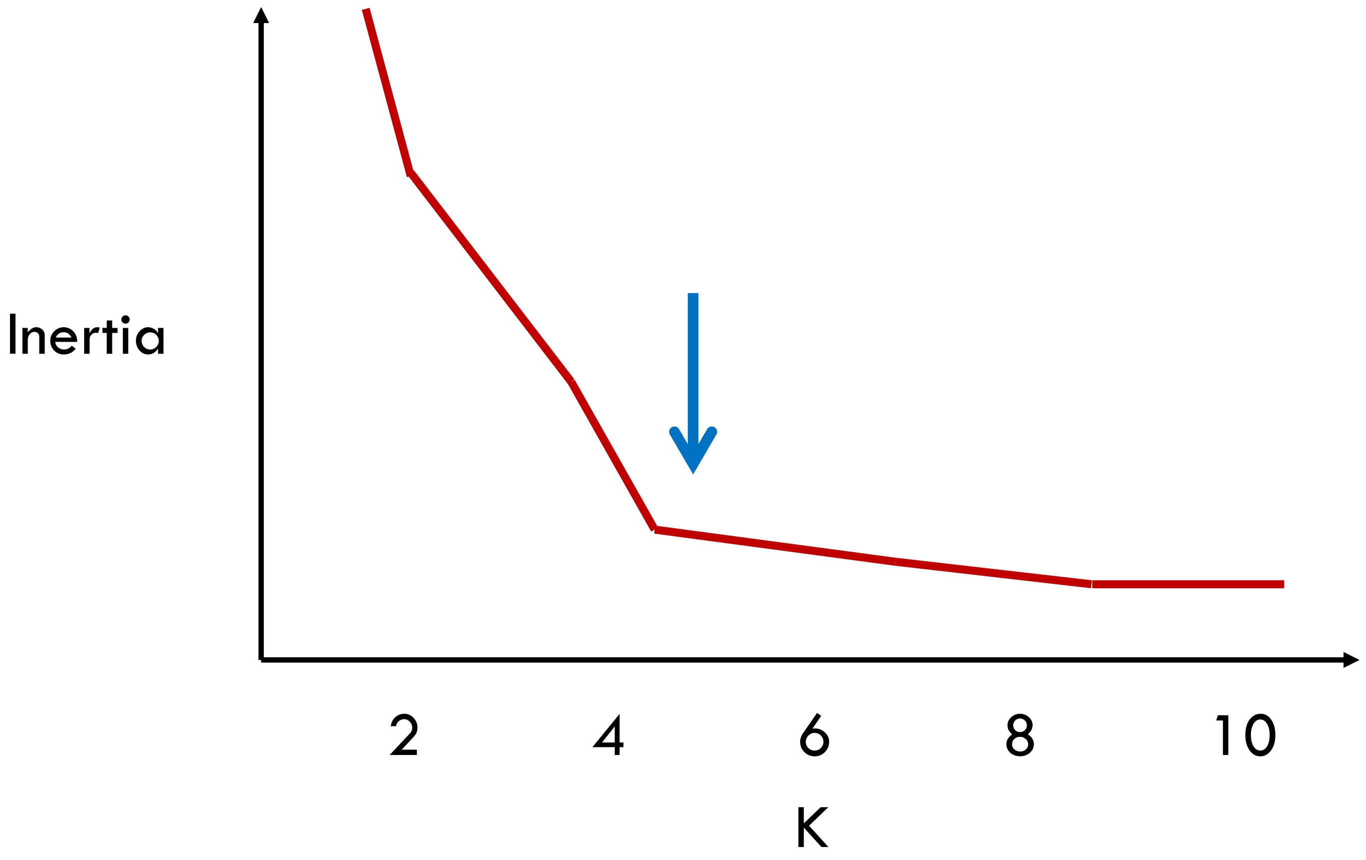
Choosing the Right Number of Clusters



- Inertia measures distance of point to cluster

r

Choosing the Right Number of Clusters



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

K-Means: The Syntax

Import the class containing the clustering method

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

K-Means: The 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='k-means++')
```

K-Means: The 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='k-means++')
```

final number of clusters

K-Means: The 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='k-means++')
```

kmeans++
cluster
initiation

K-Means: The 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='k-means++')
```

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

```
kmeans = kmeans.fit(X1)
```

K-Means: The 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='k-means++')
```

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.

Pros and Cons of the K-means Algorithm

Pros	Cons
Simple and easy to implement	Sensitive to initial centroid selection
Fast and computationally efficient	May converge to local optima
Scalable to large datasets	Requires predefined number of clusters (k)
Versatile and widely used	Performs poorly with non-linear data
Effective for spherical clusters	Sensitivity to outliers

Summary