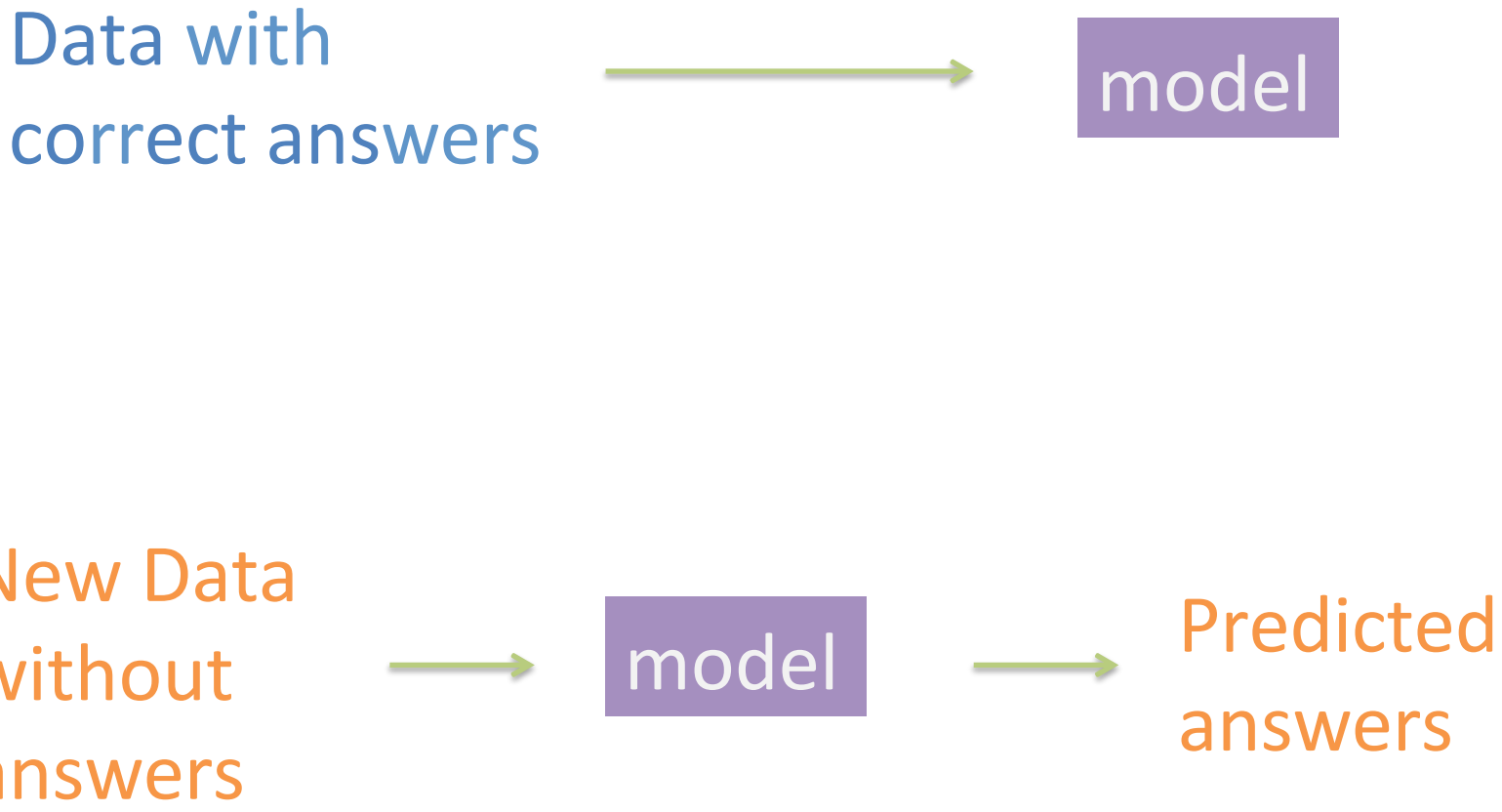
Unsupervised Learning

Supervised Learning



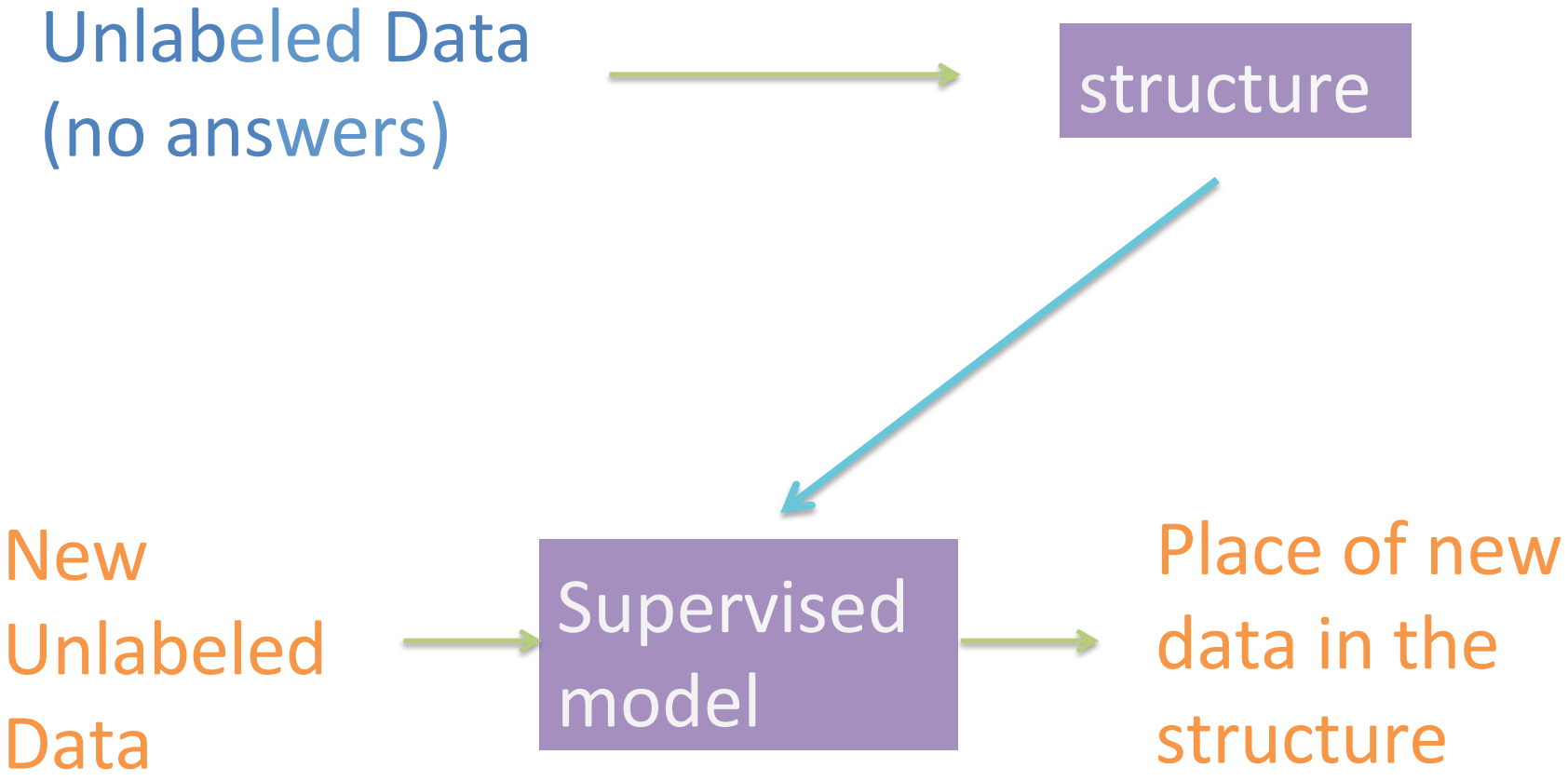
Unsupervised Learning

Unlabeled Data
(no answers)

structure

Making a map to better understand data

Unsupervised Learning



Classification: model with category labels

Color, shape, weight, sweetness, sourness for a bunch of apples, bananas & peaches

model

Color, shape, weight, sweetness, sourness (without fruit type)

model

Predict apple, banana or peach

Clustering: Finding separate groups in data

Color, shape, weight, sweetness, sourness for a bunch of alien fruits

Identify different fruits

(still don't know what each group is)

Finding units with similar behavior
(friend groups, similar products, etc.)

Market segmentation

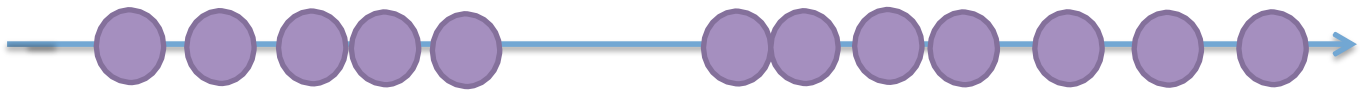
Understanding a complex system
(like purchases or friendships or flows)

Finding meaningful categories for your items

Fewer classes for classification by grouping
(change the resolution of the problem)

Users of a web app

1 Feature: Age



Age

Users of a web app

1 Feature: Age

2 clusters

Age

Users of a web app

1 Feature: Age

3 clusters

Age

Users of a web app

1 Feature: Age

5 clusters

Age

Users of a web app

1 Feature: Age

5 clusters

No “correct” answer

Not one way to map the structure

Age

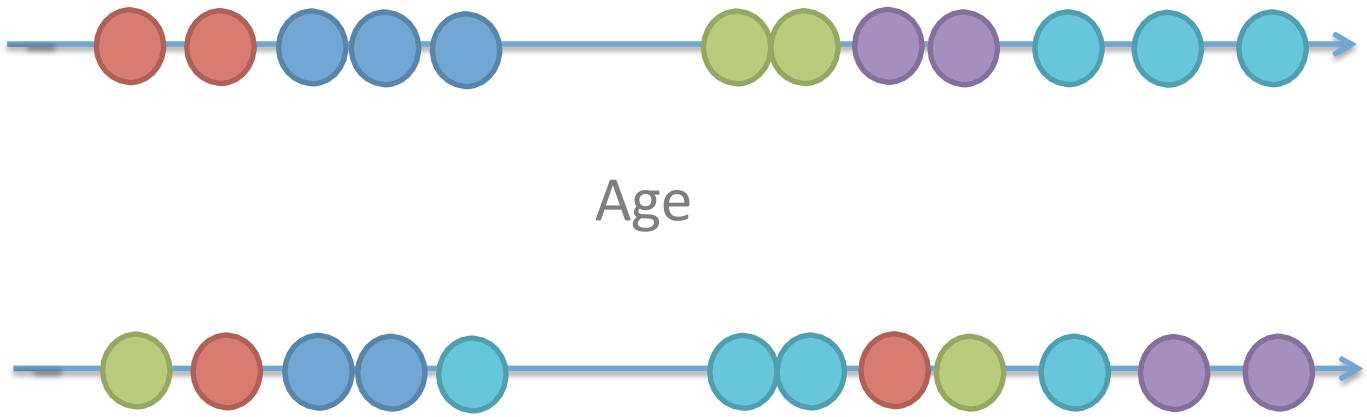
Users of a web app

1 Feature: Age

5 clusters

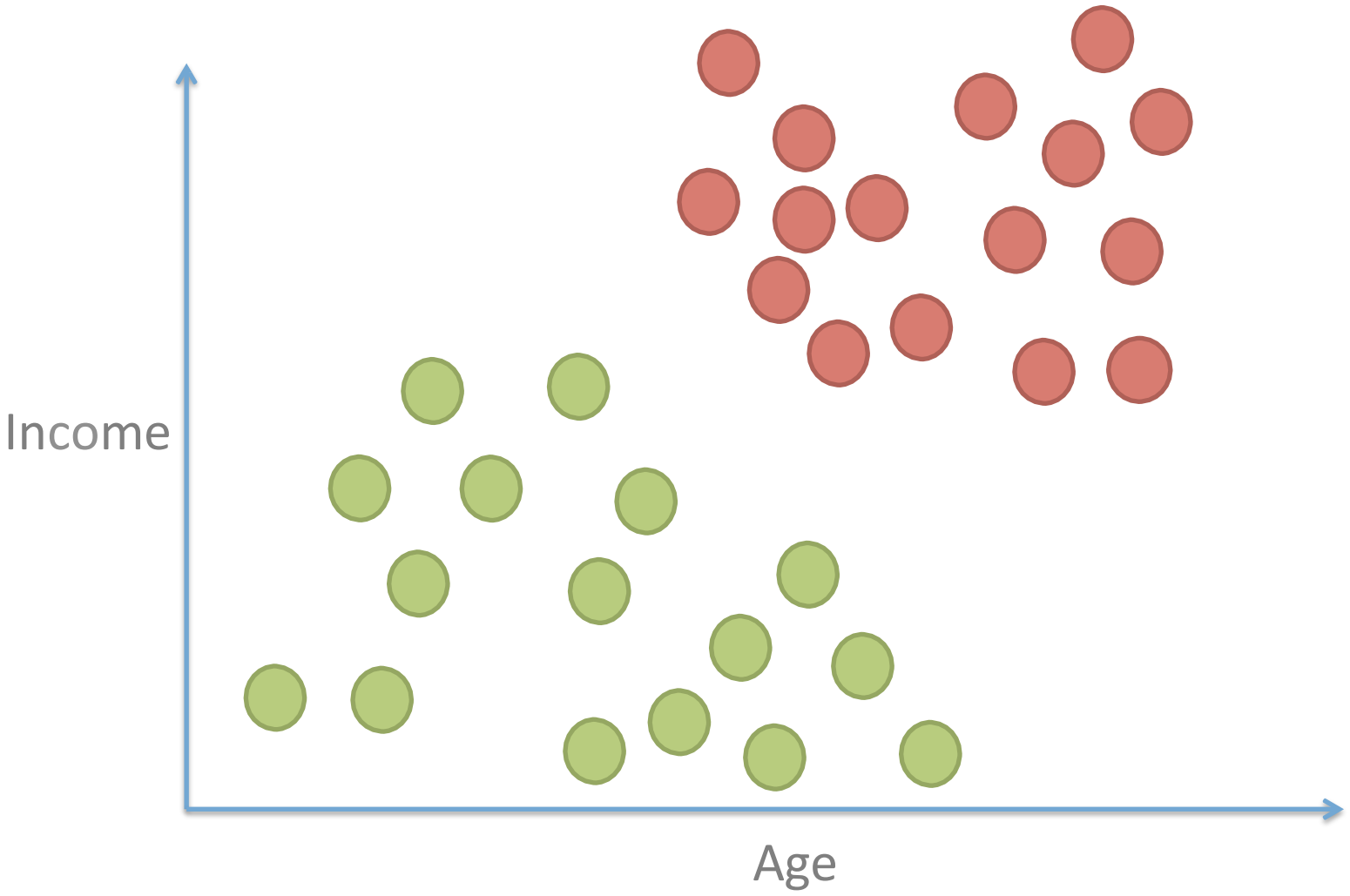
No “correct” answer

But there are better/worse maps within the same structure



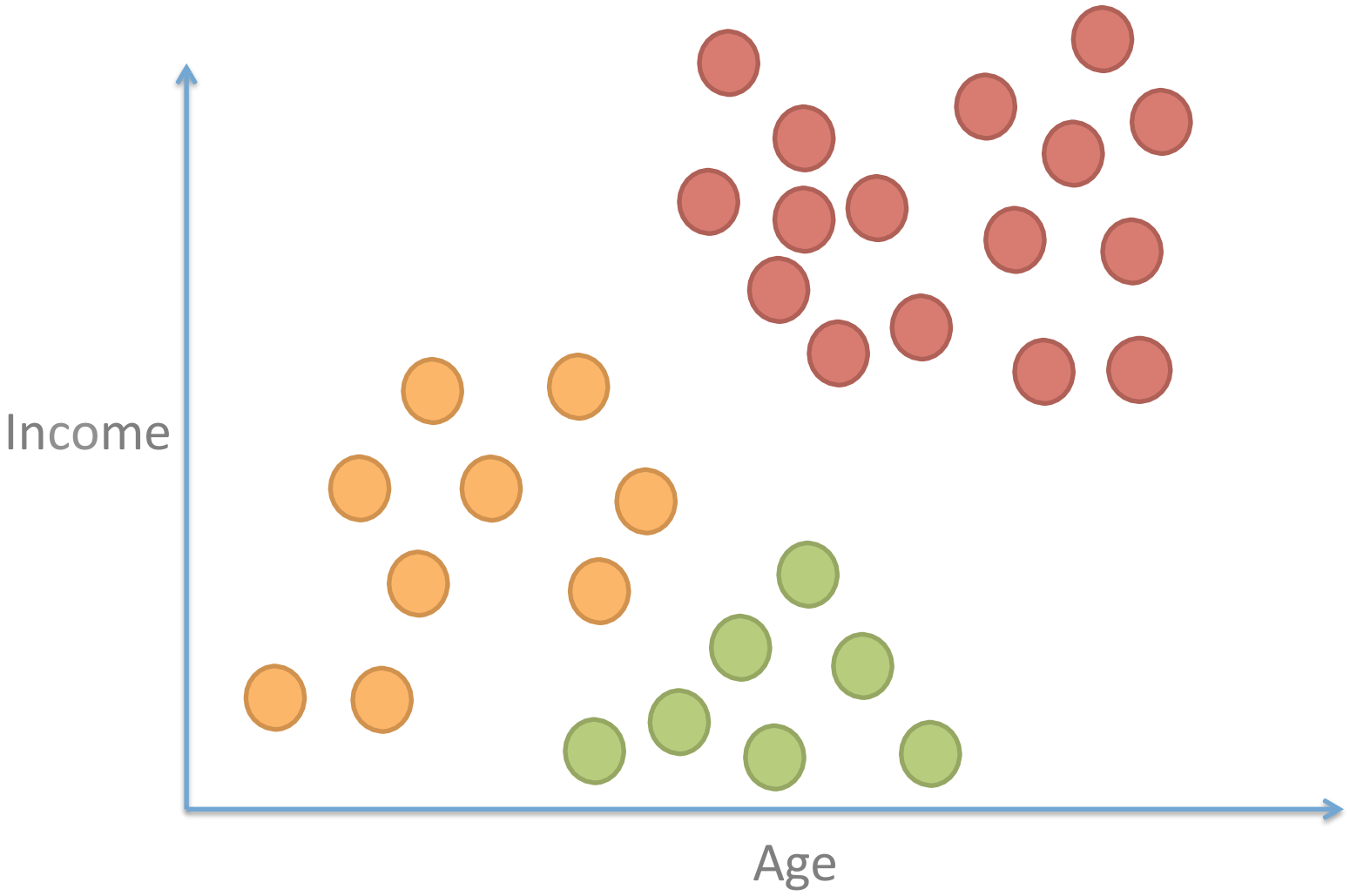
2 Features: Age, Income

2 clusters



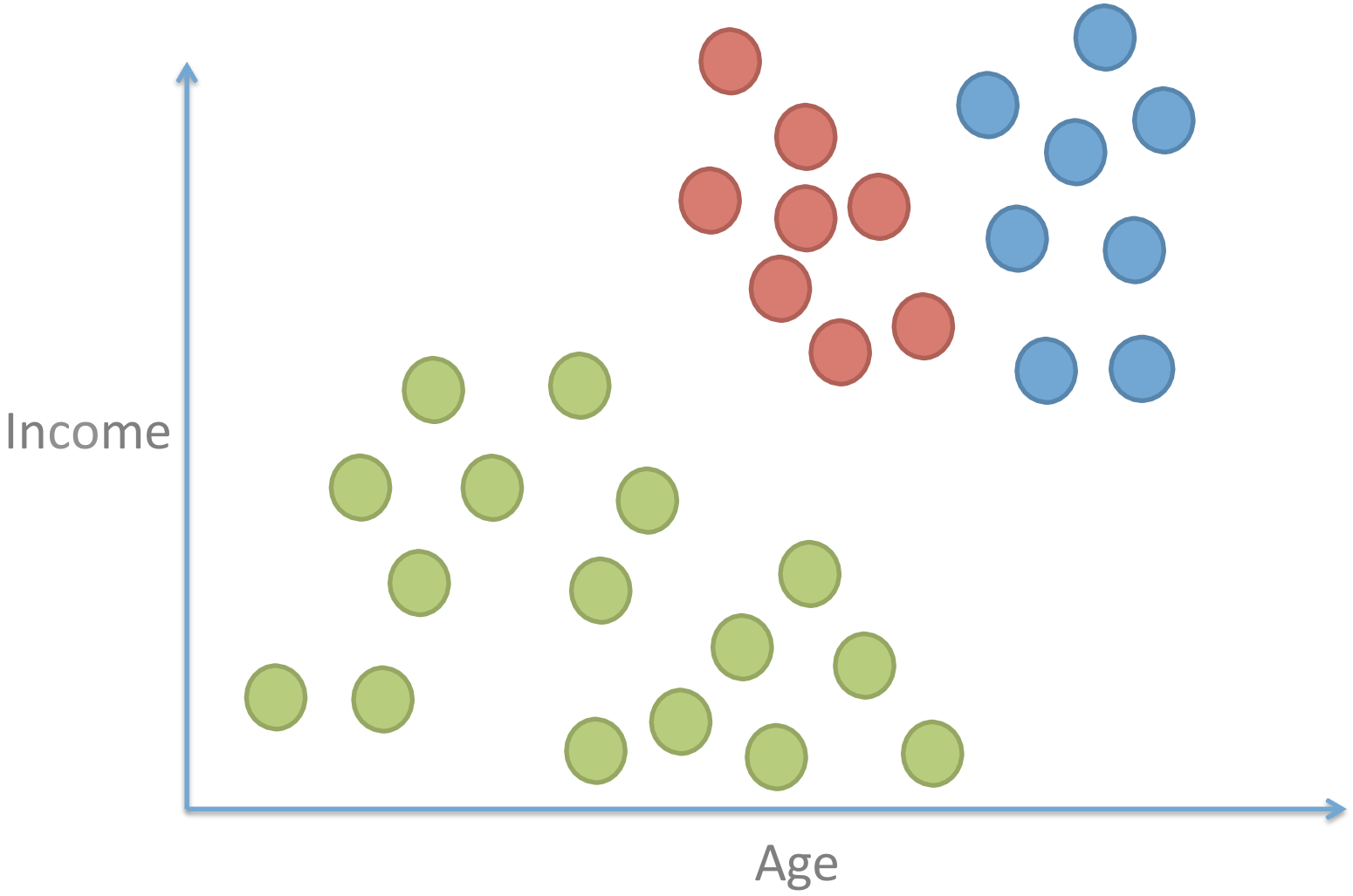
2 Features: Age, Income

3 clusters



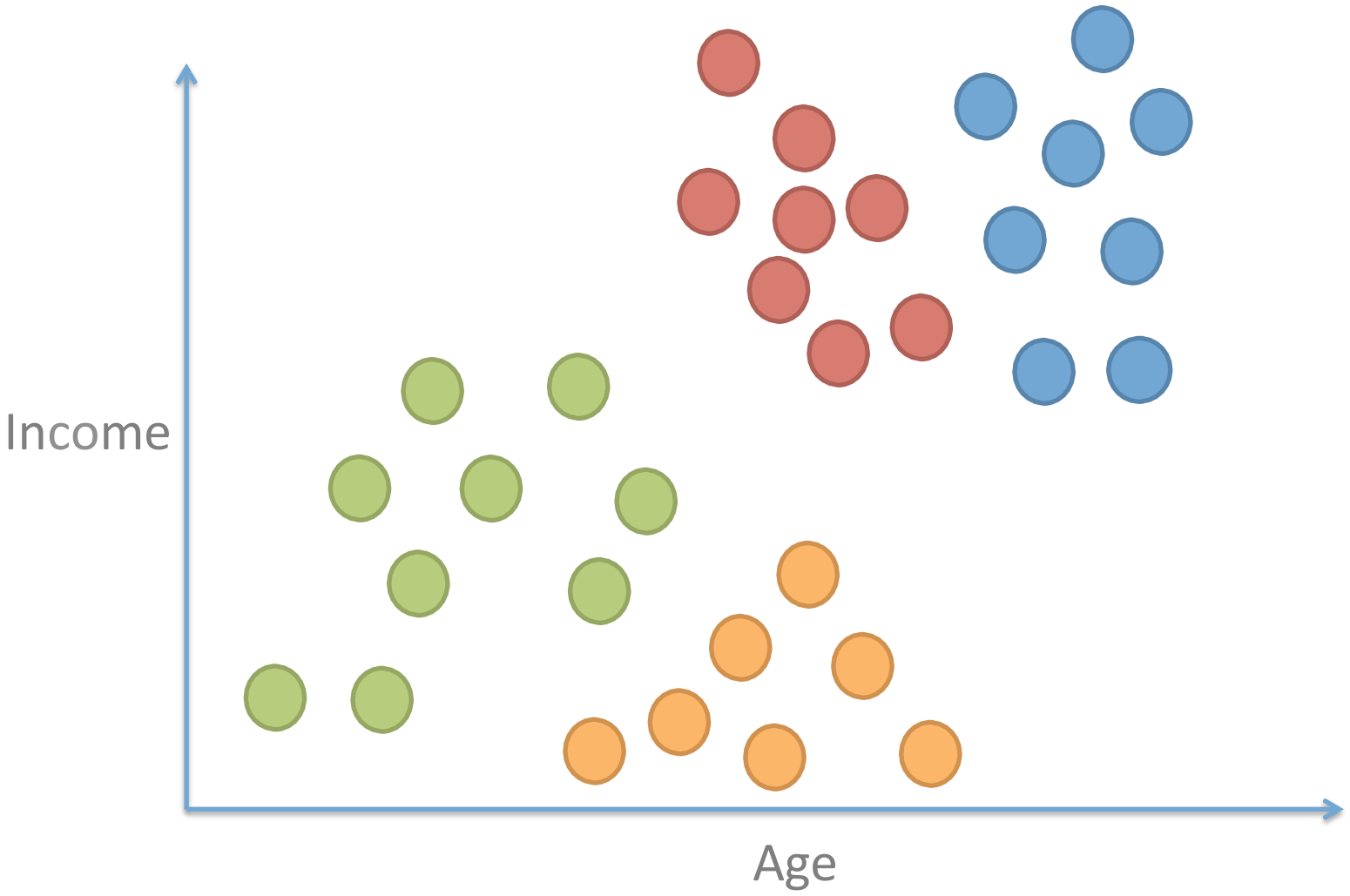
2 Features: Age, Income

3 clusters OR?



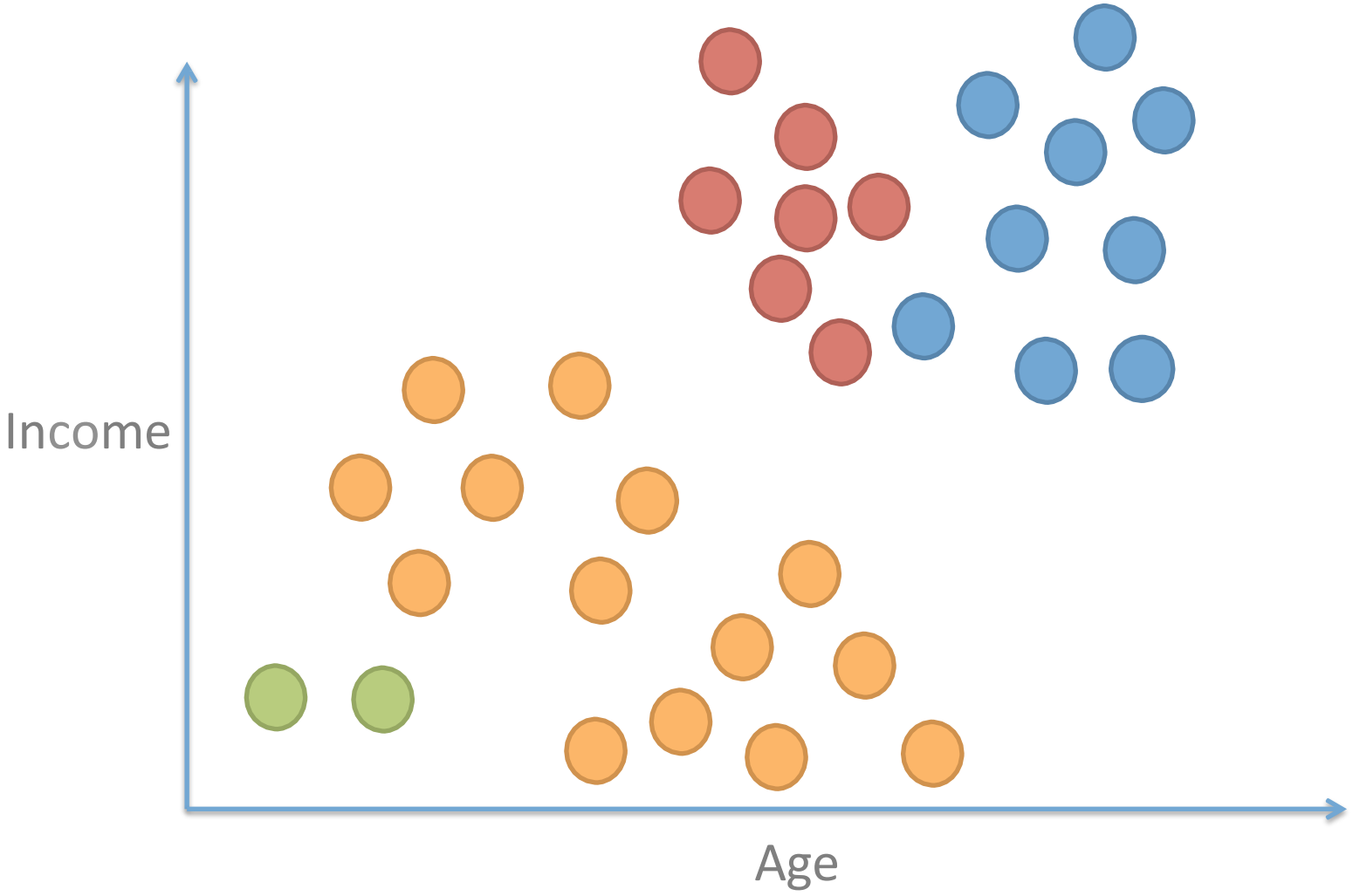
2 Features: Age, Income

4 clusters

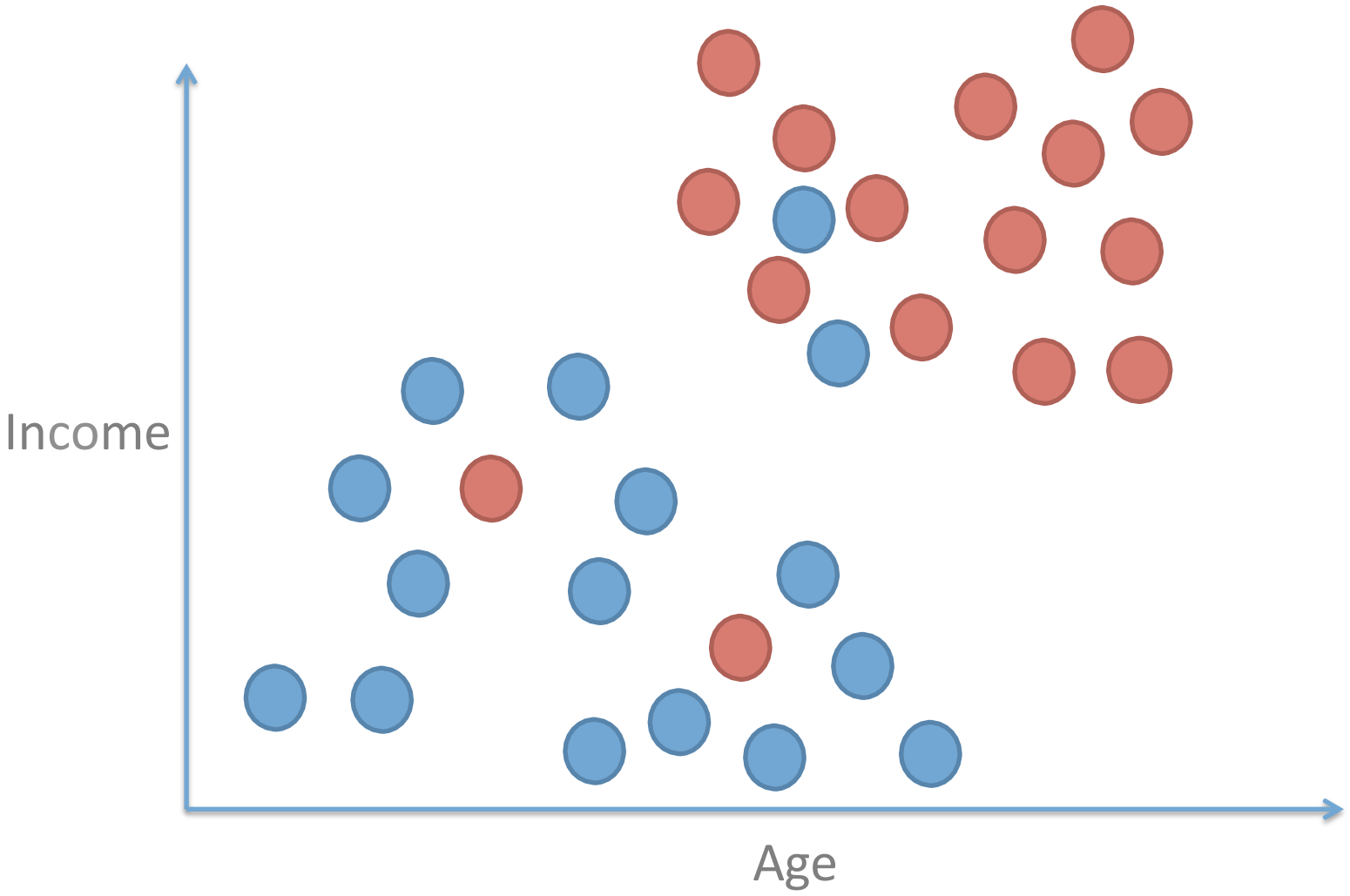


2 Features: Age, Income

4 clusters OR?

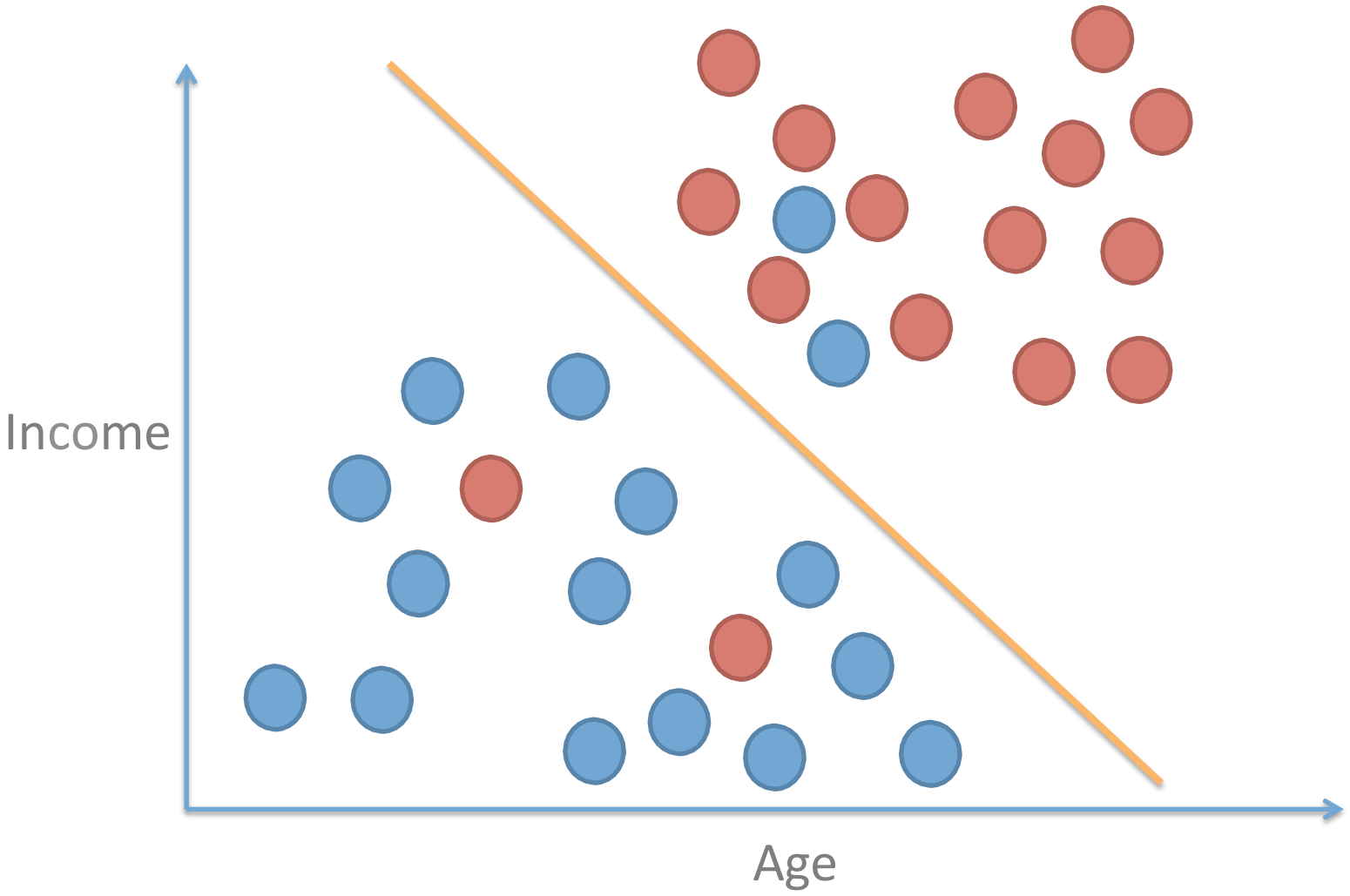


Supervised Learning

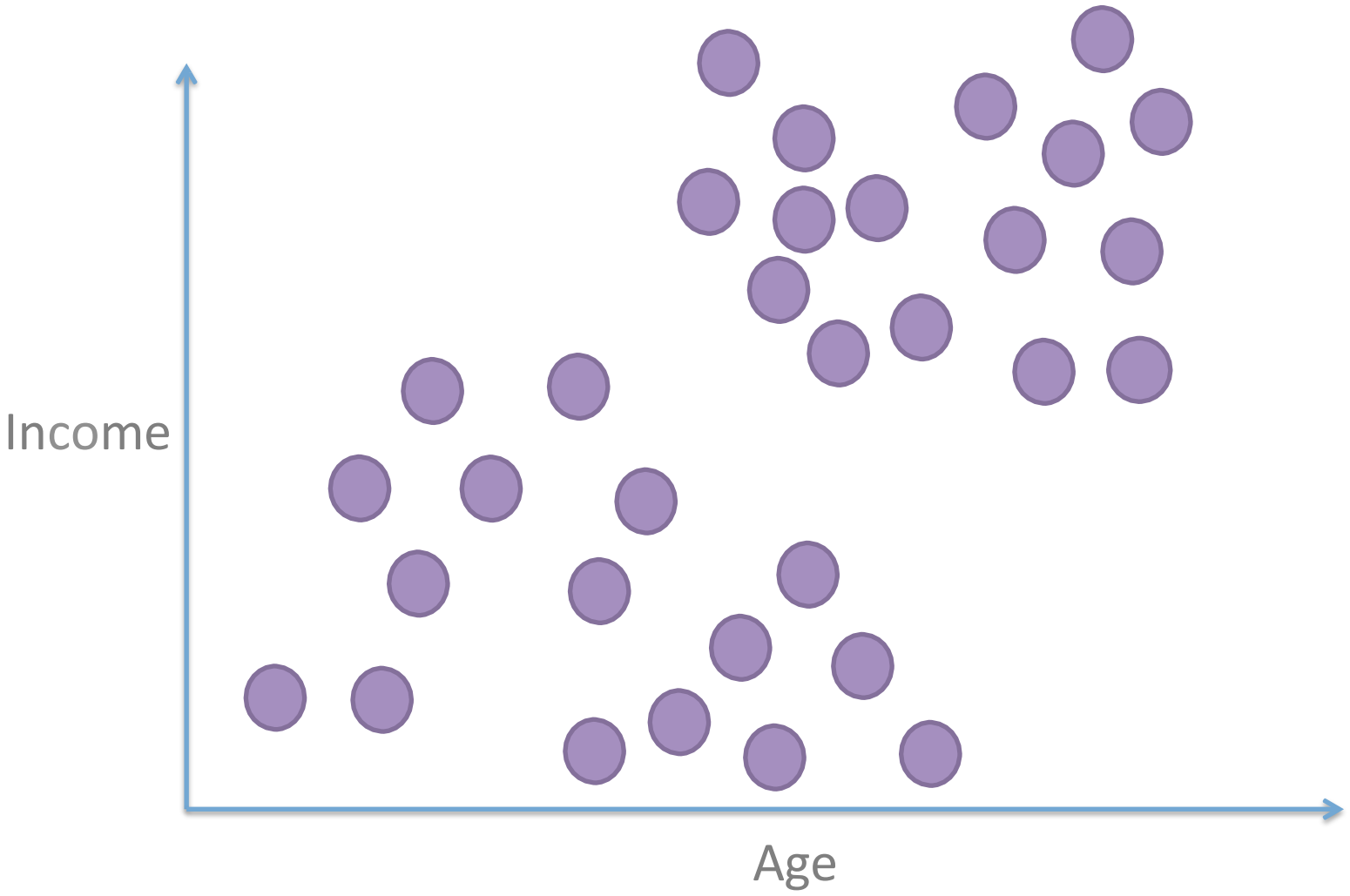


Supervised Learning

Find decision boundary using labels

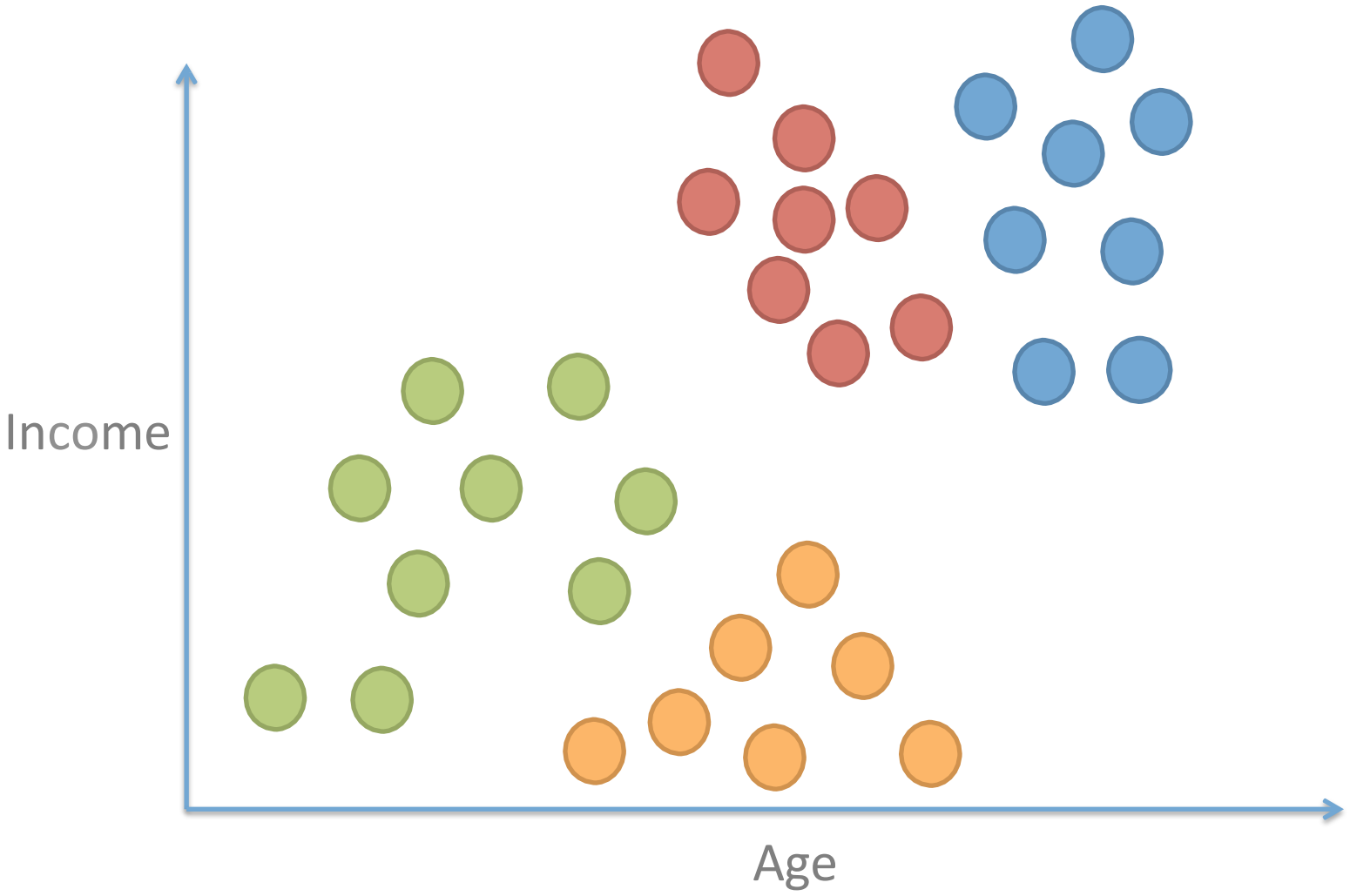


Unsupervised Learning



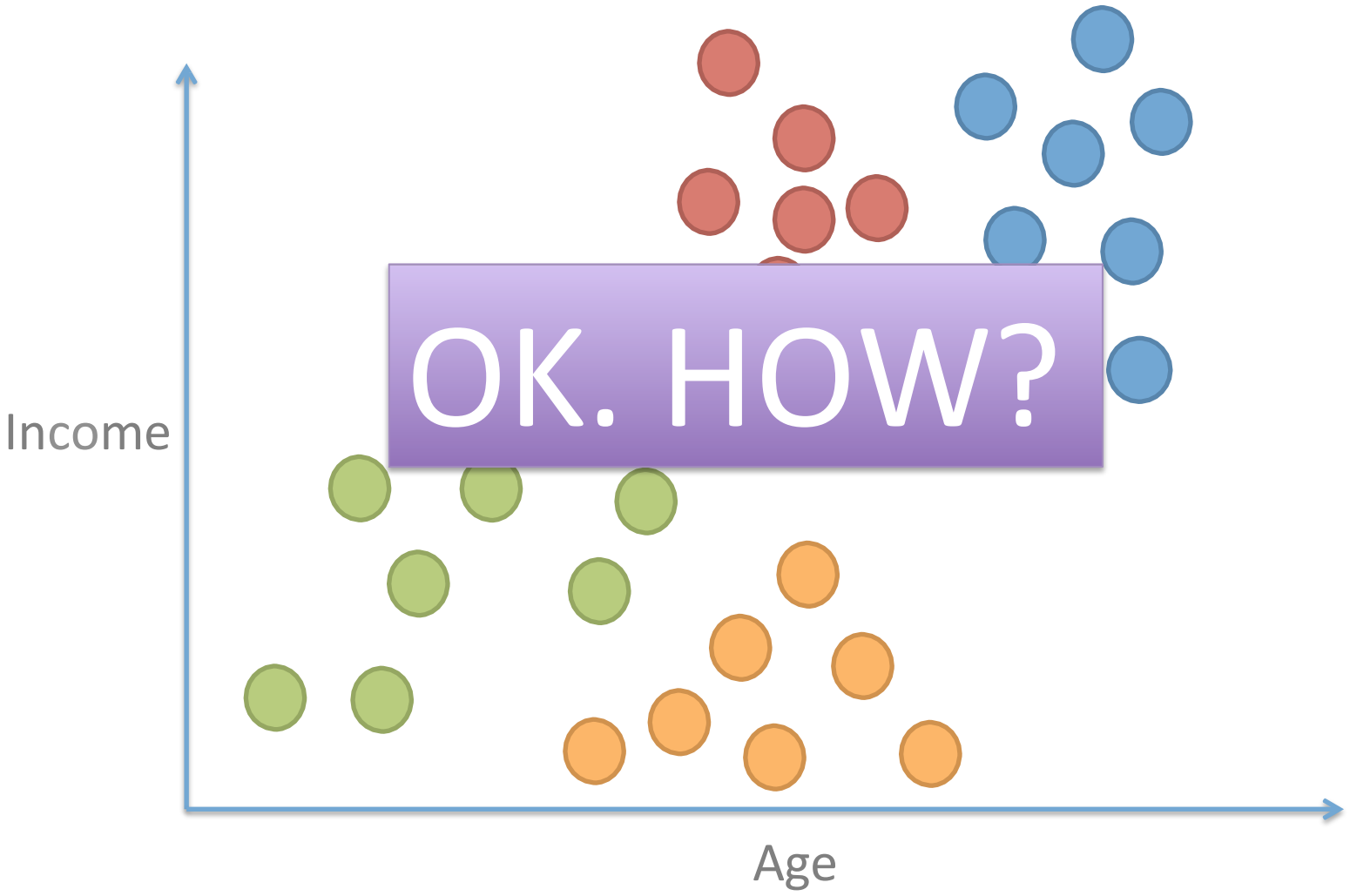
Unsupervised Learning

Find structure in unlabeled data



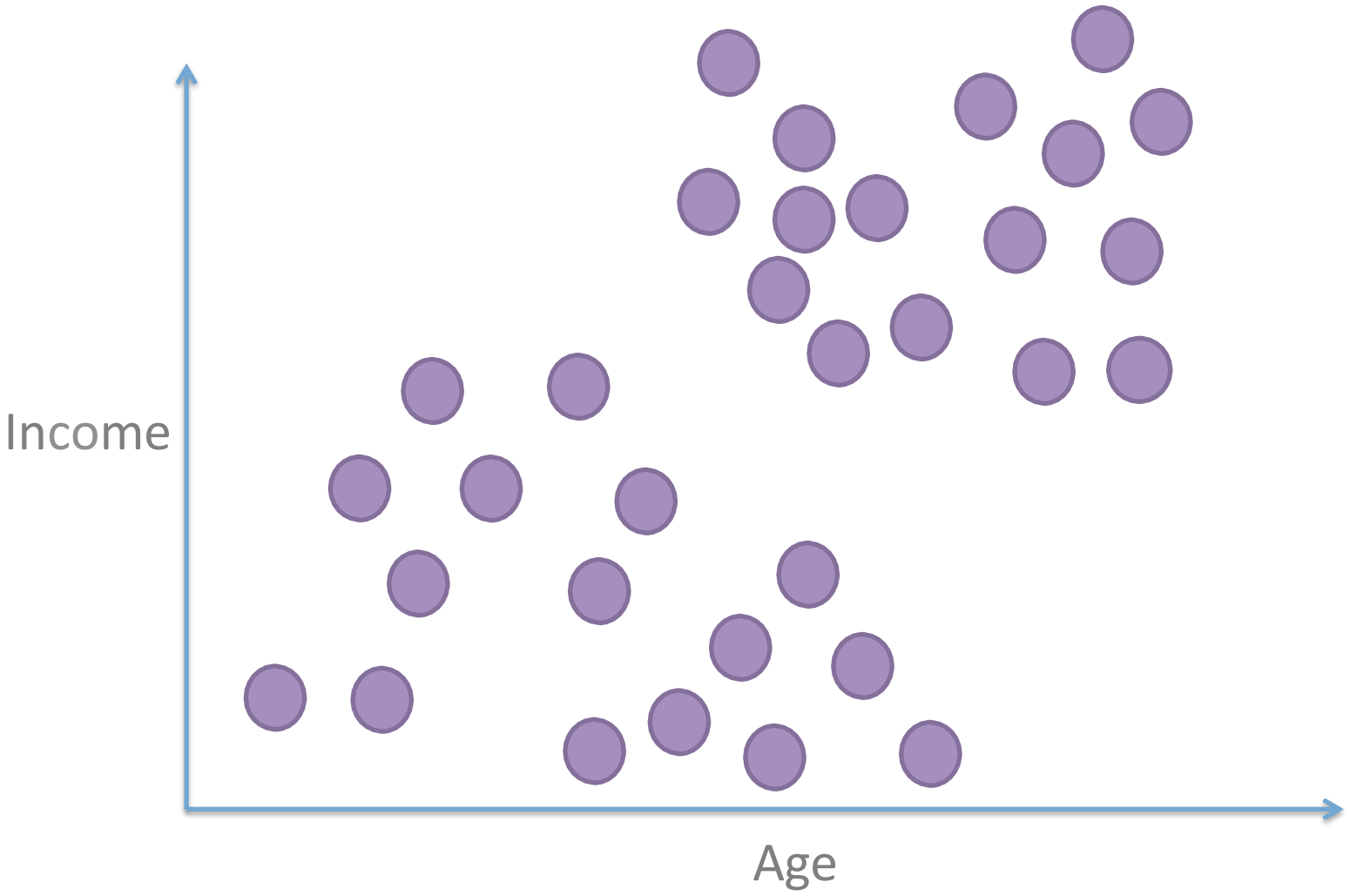
Unsupervised Learning

Find structure in unlabeled data



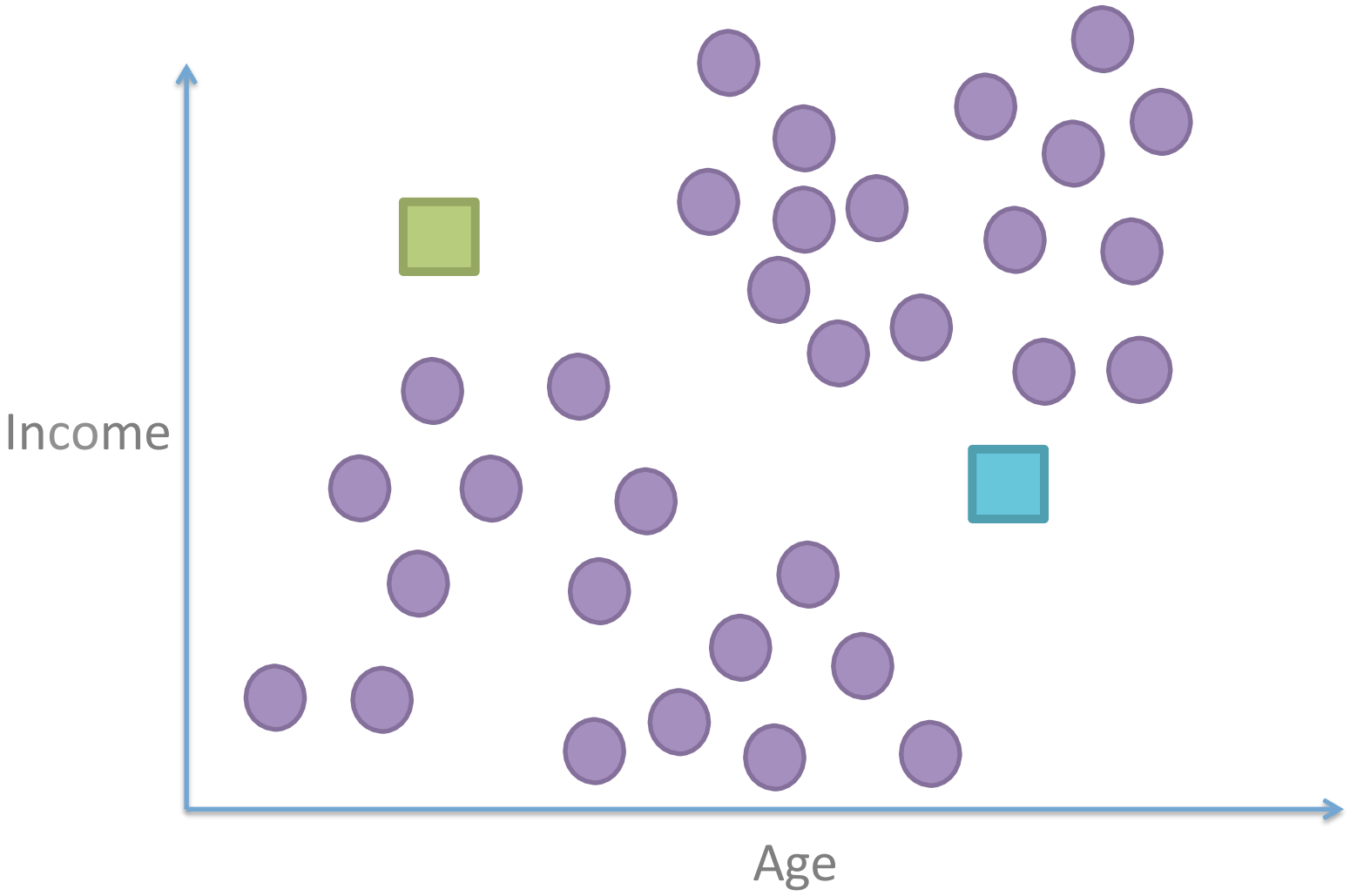
K-Means algorithm

K=2 (find 2 clusters)



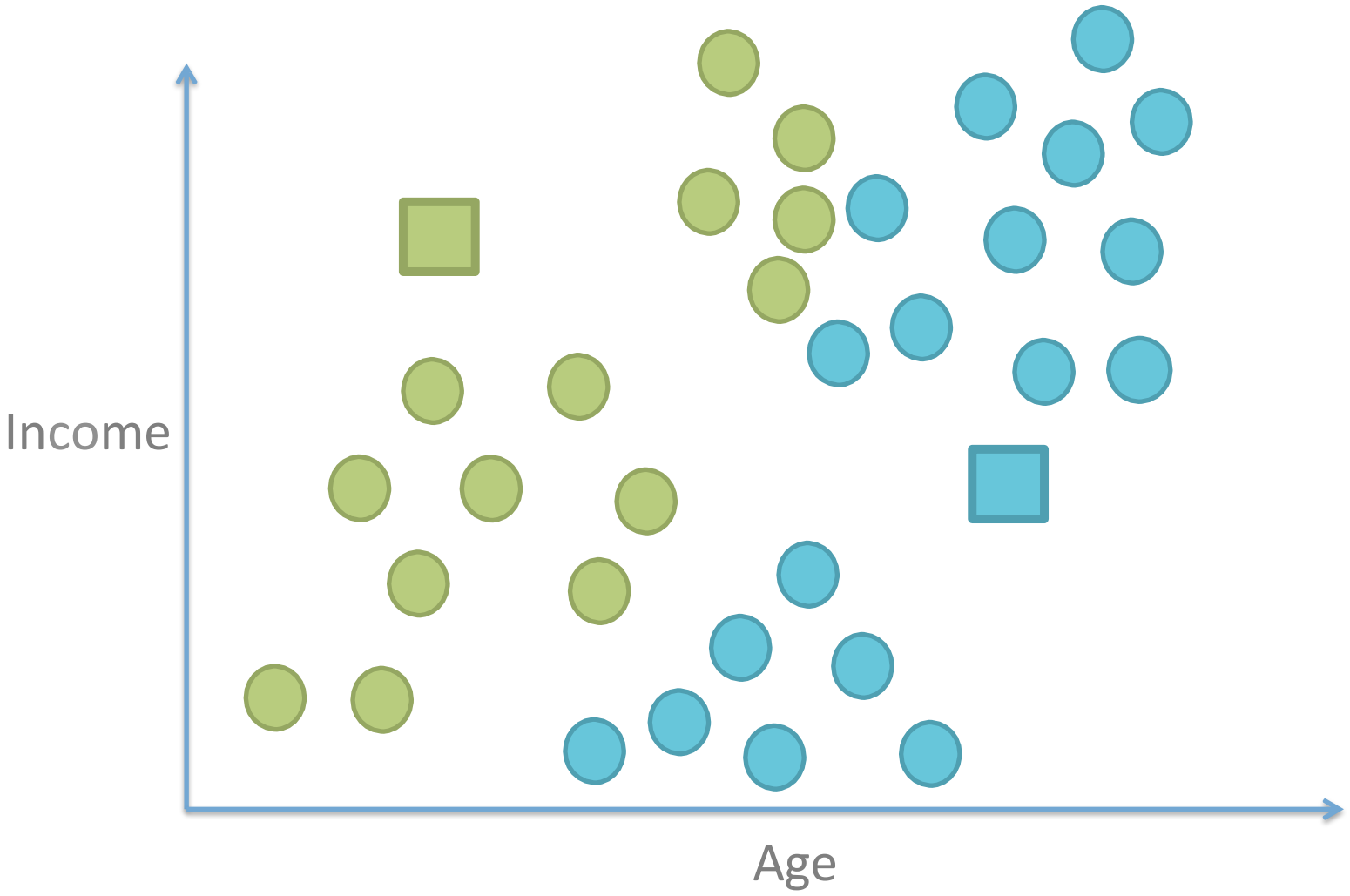
K-Means K=2

Randomly assign two cluster centers



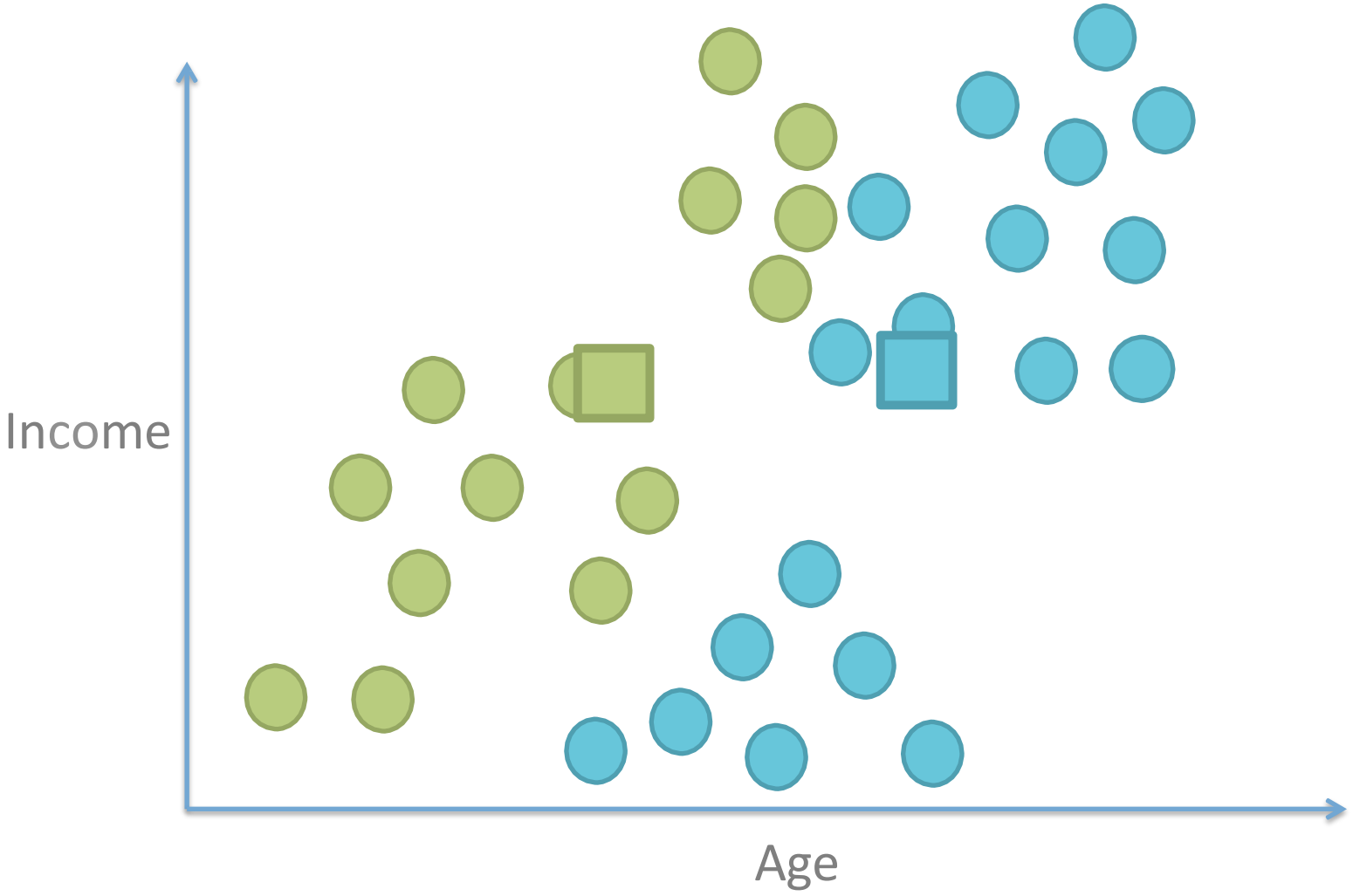
K-Means K=2

Each point belongs to closest center



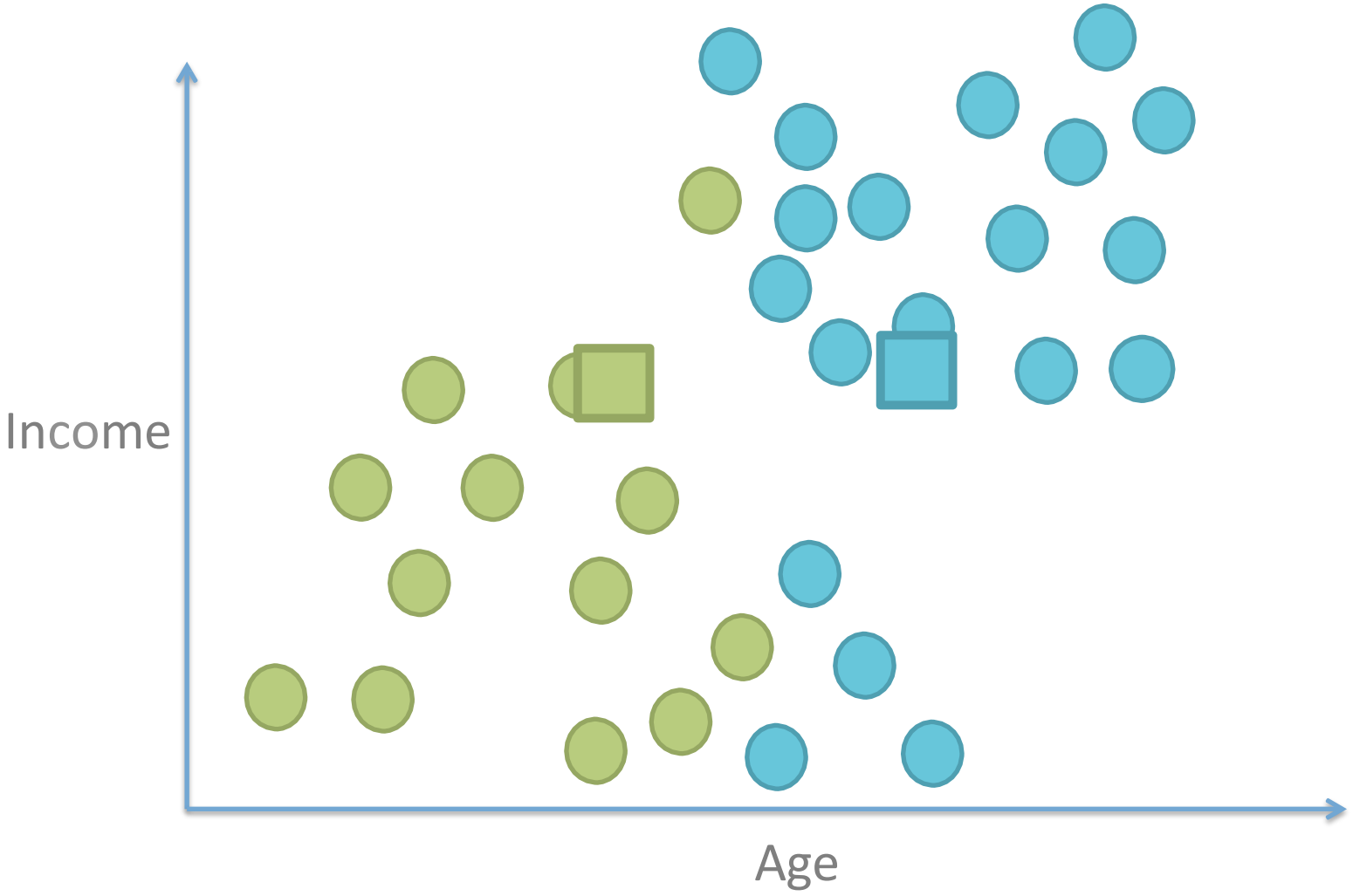
K-Means K=2

Move each center to the cluster's mean



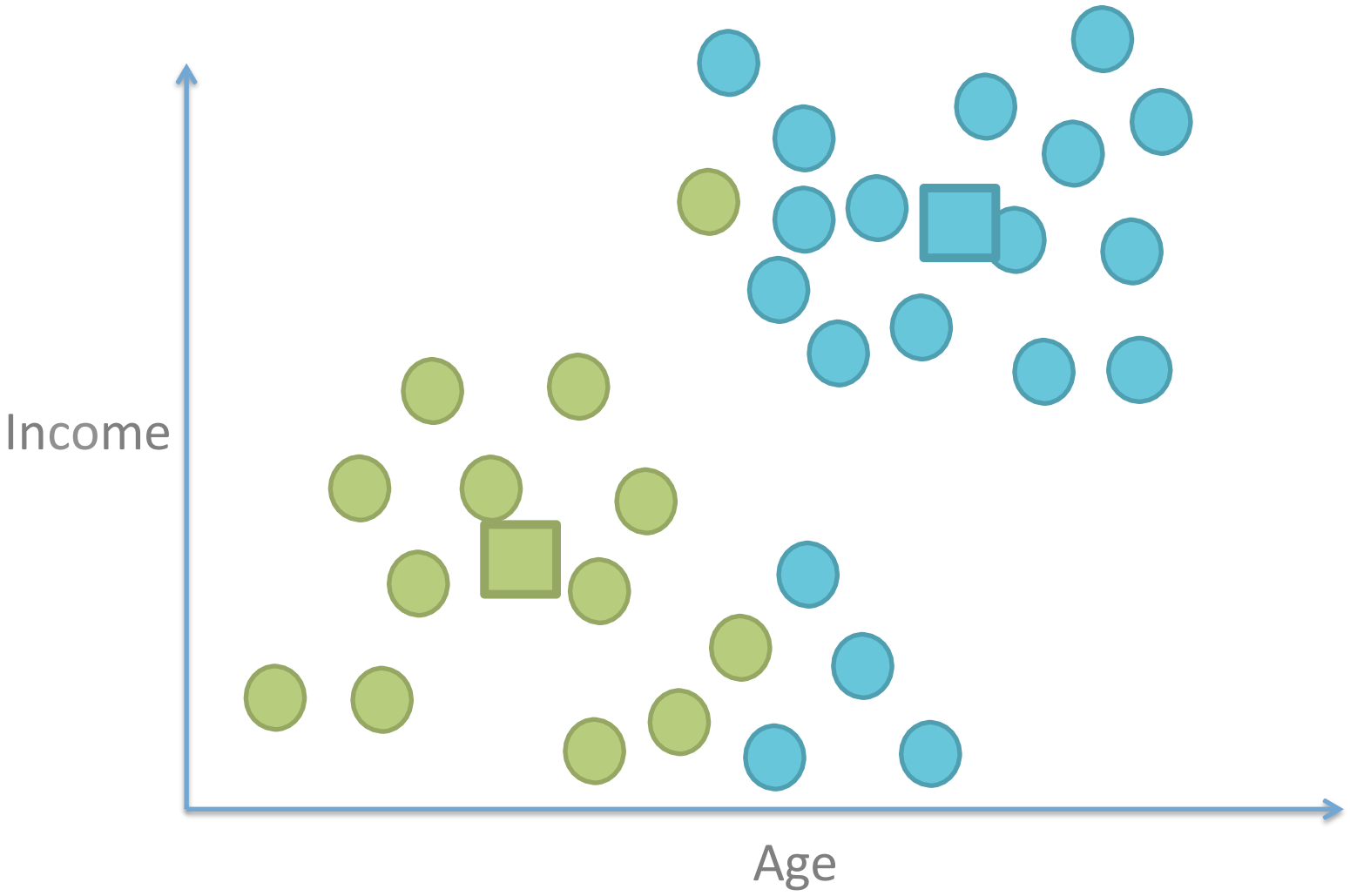
K-Means K=2

Each point belongs to the closest center



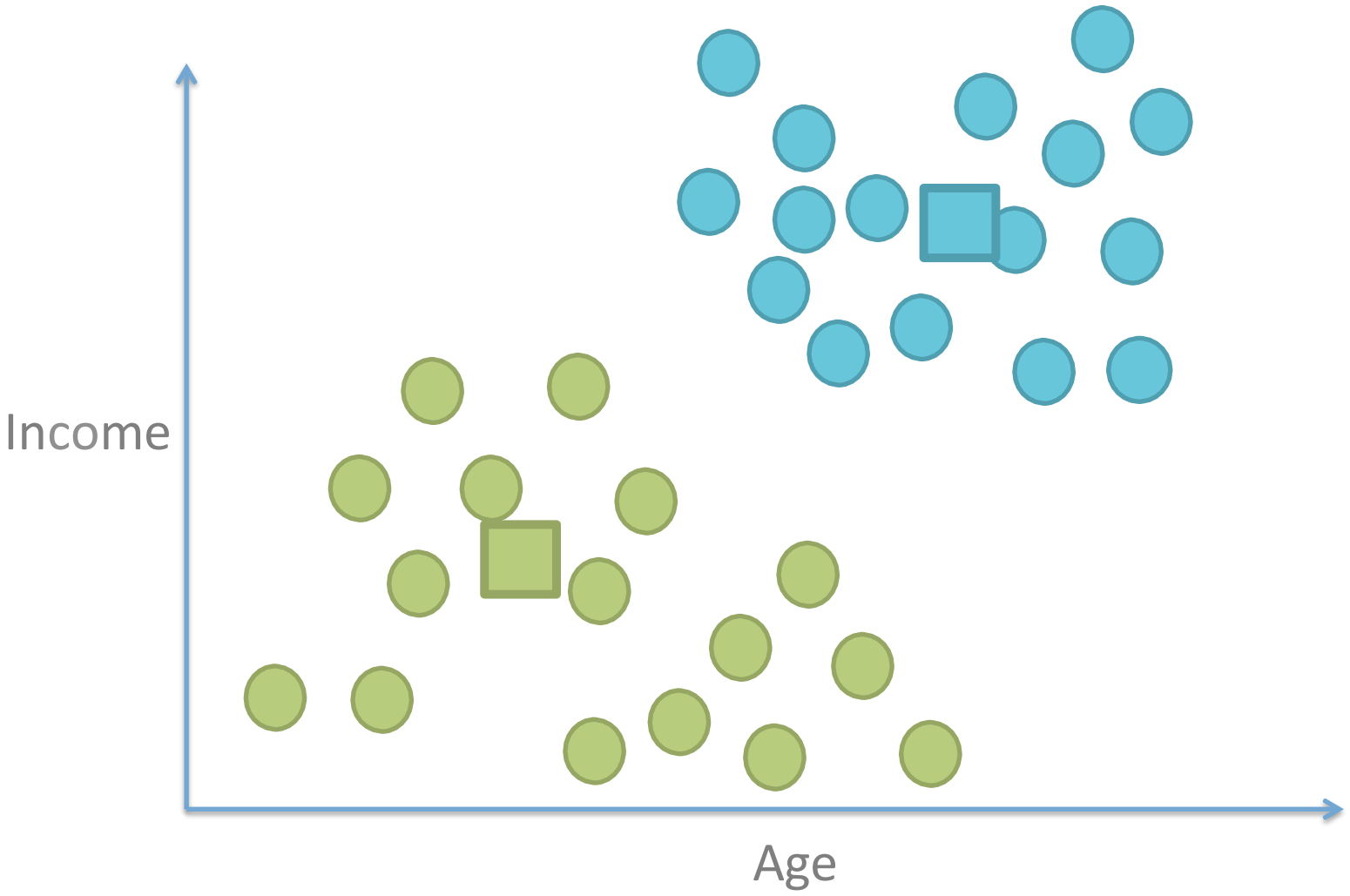
K-Means K=2

Move each center to the cluster's mean



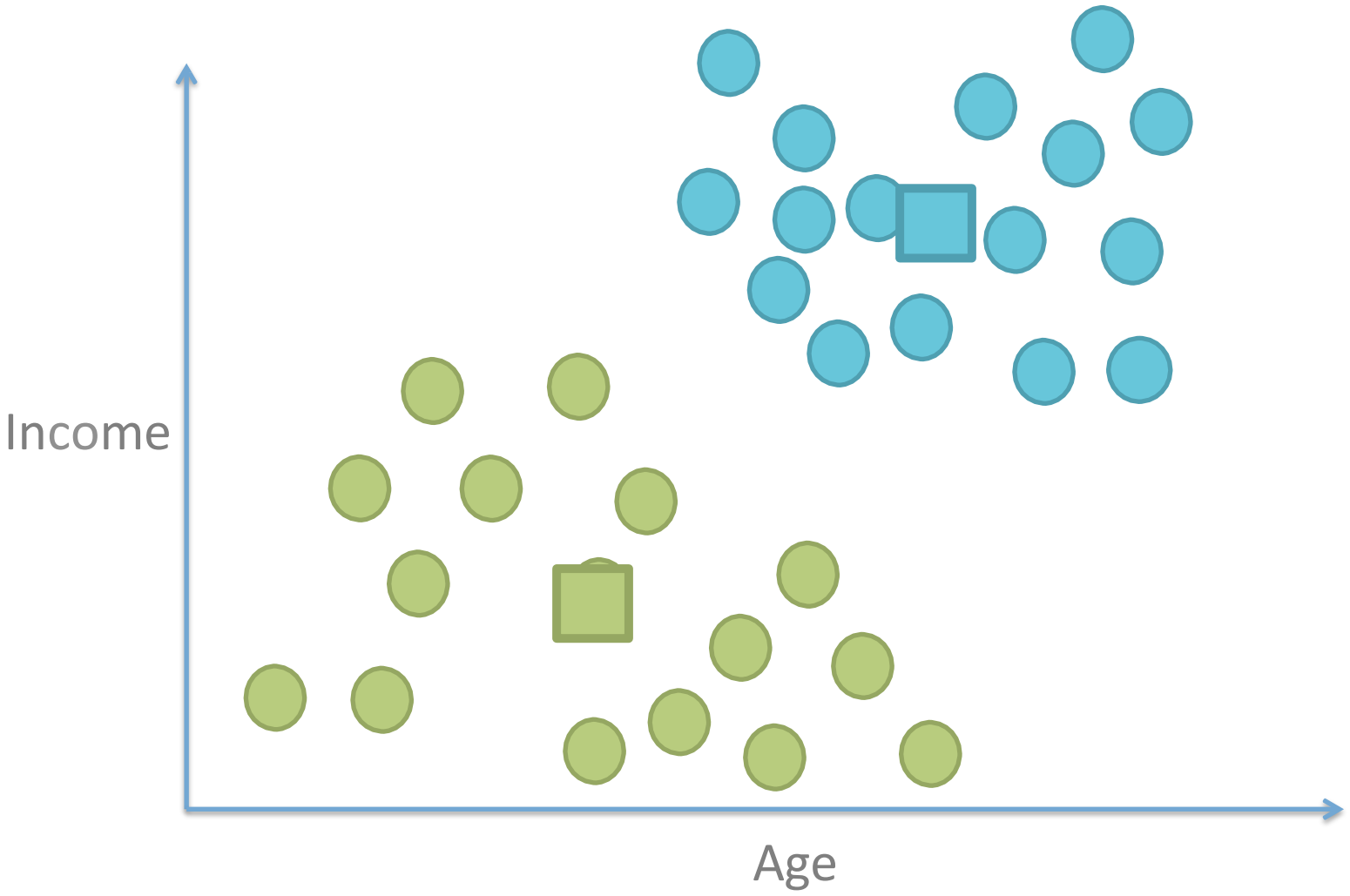
K-Means K=2

Each point belongs to the closest center



K-Means K=2

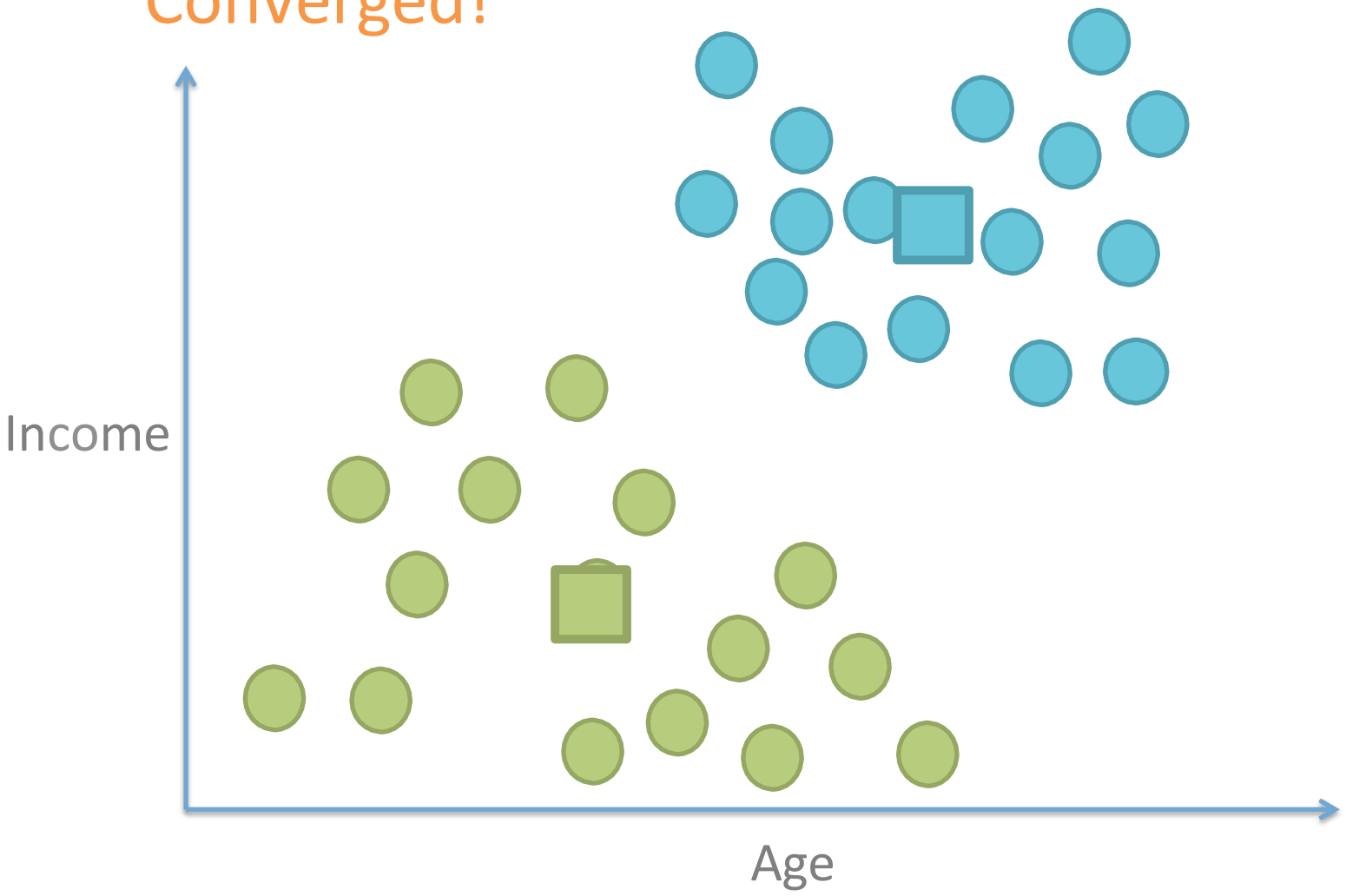
Move each center to the cluster's mean



K-Means K=2

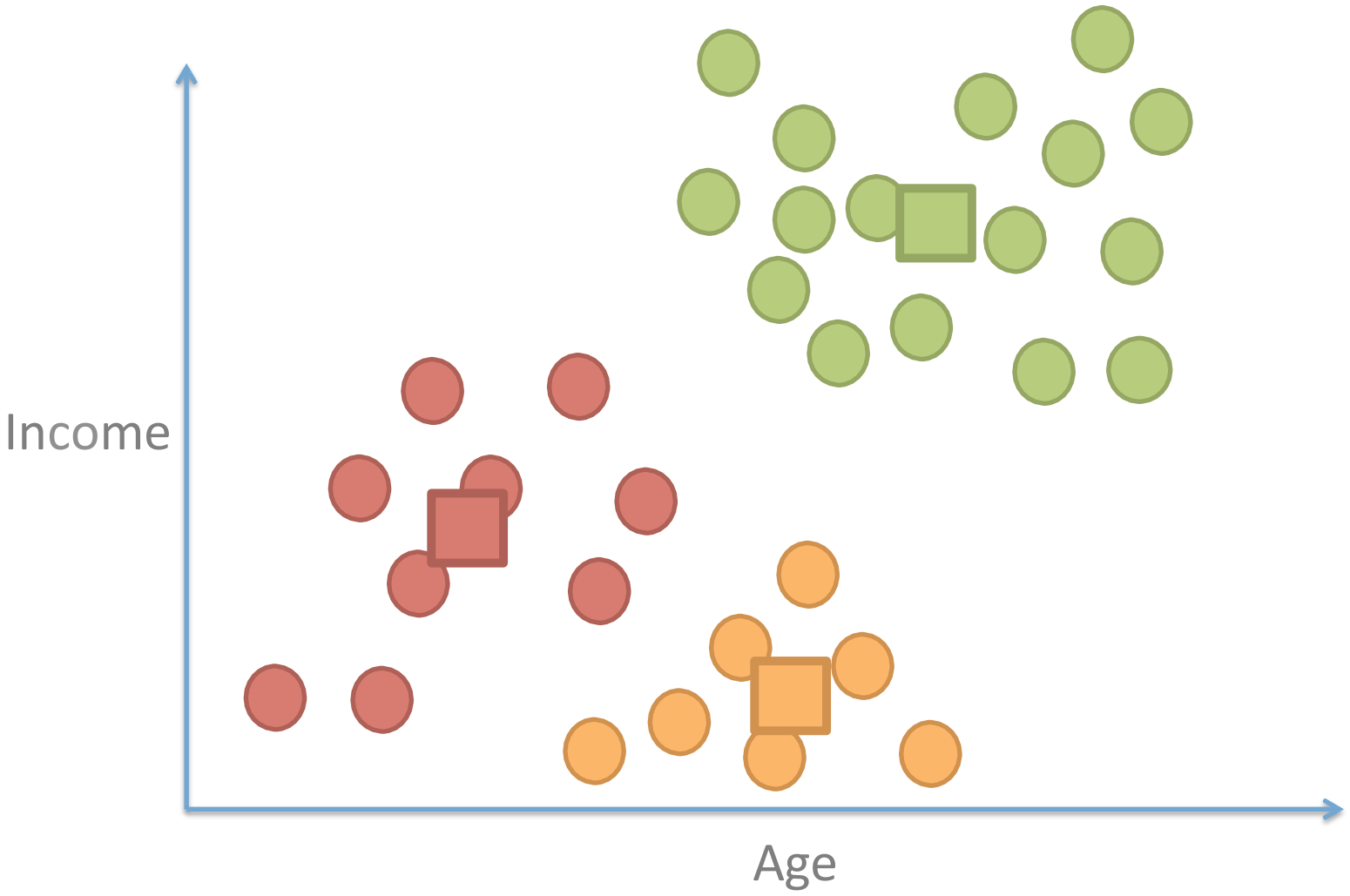
Points don't change anymore.

Converged!



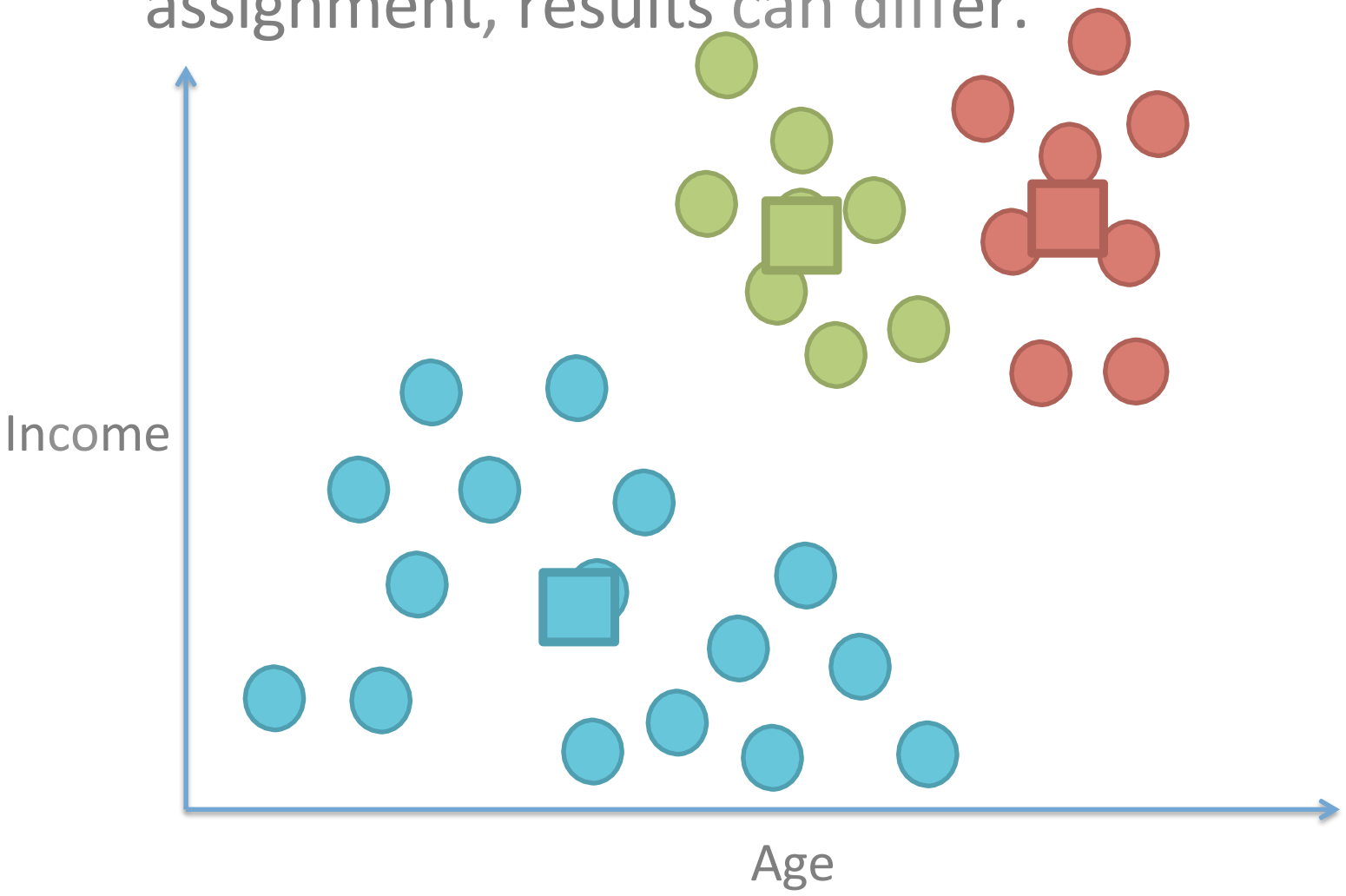
K-Means K=3

Result:



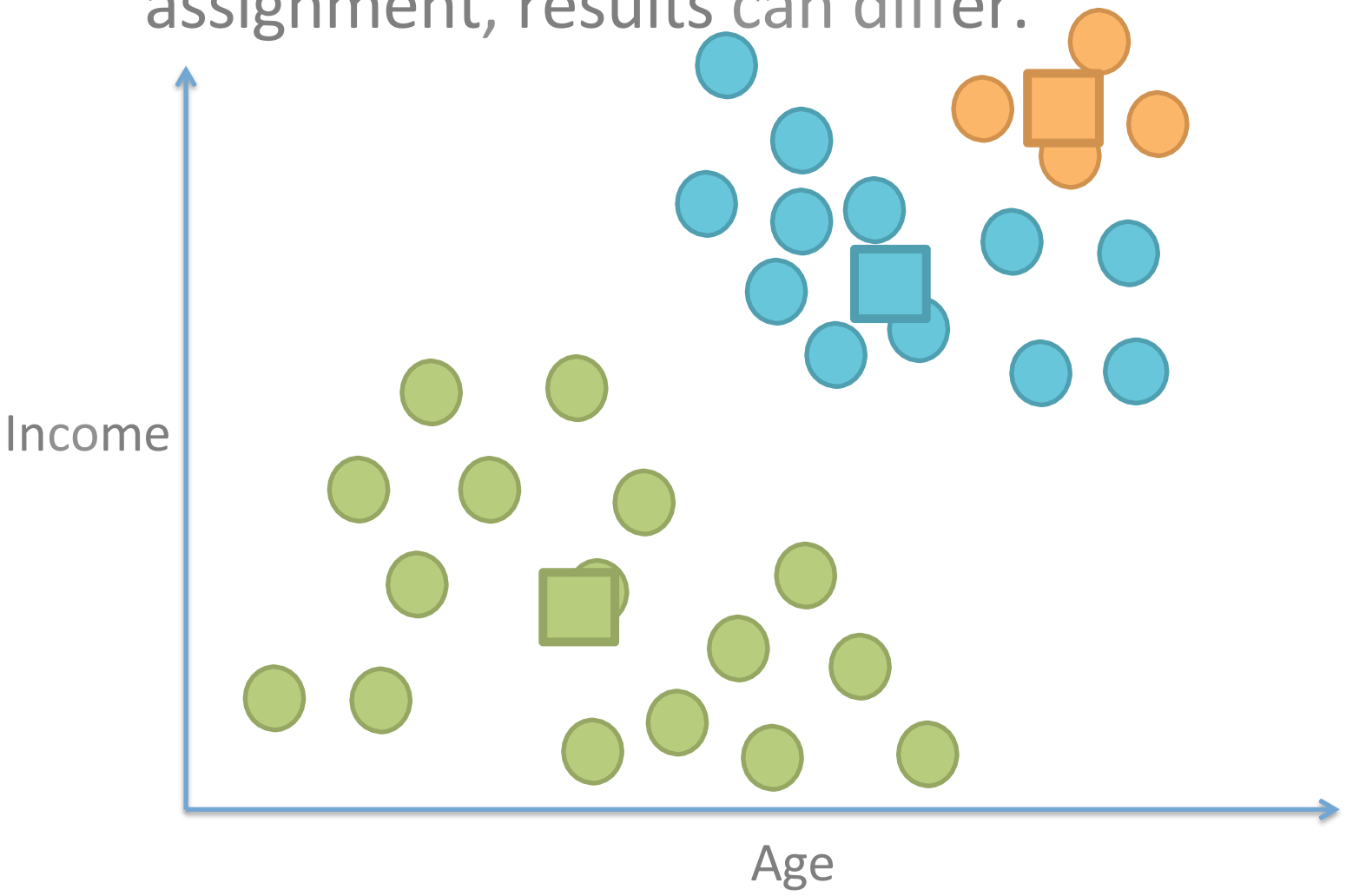
K-Means K=3

Depending on random initial assignment, results can differ.



K-Means K=3

Depending on random initial assignment, results can differ.



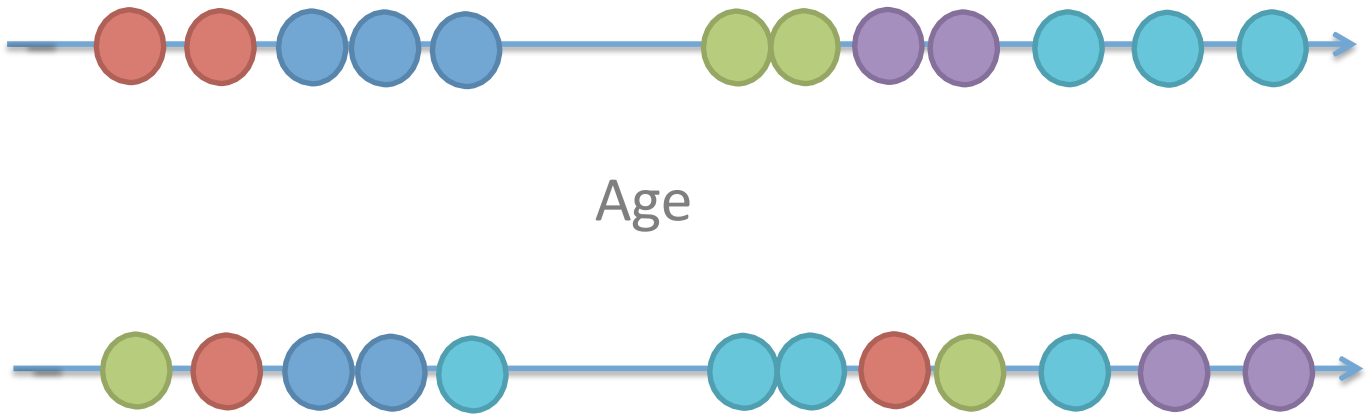
Users of a web app

1 Feature: Age

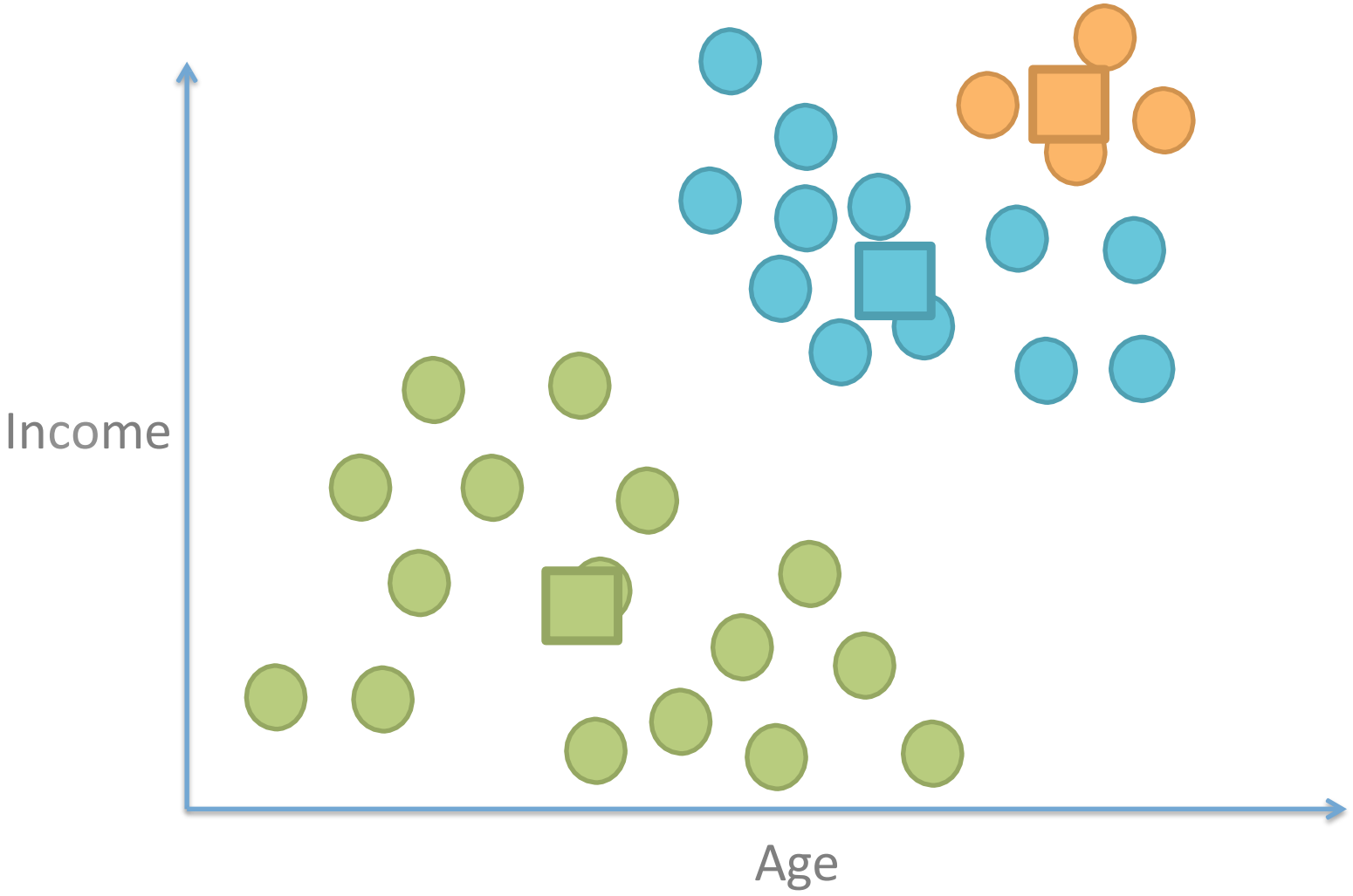
5 clusters

No “correct” answer

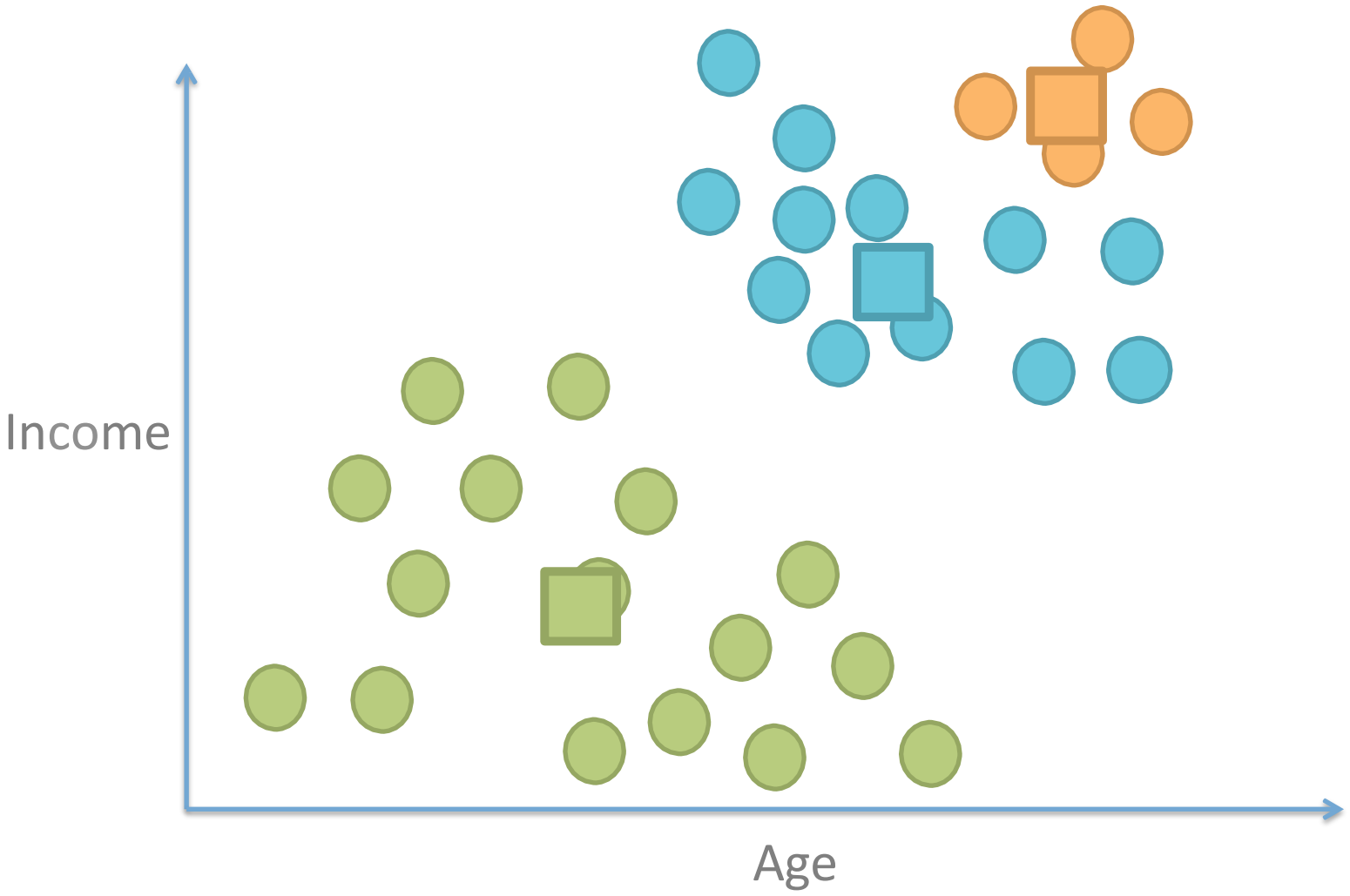
But there are better/worse maps within the same structure



Some maps are better than others: Let's find a score for each

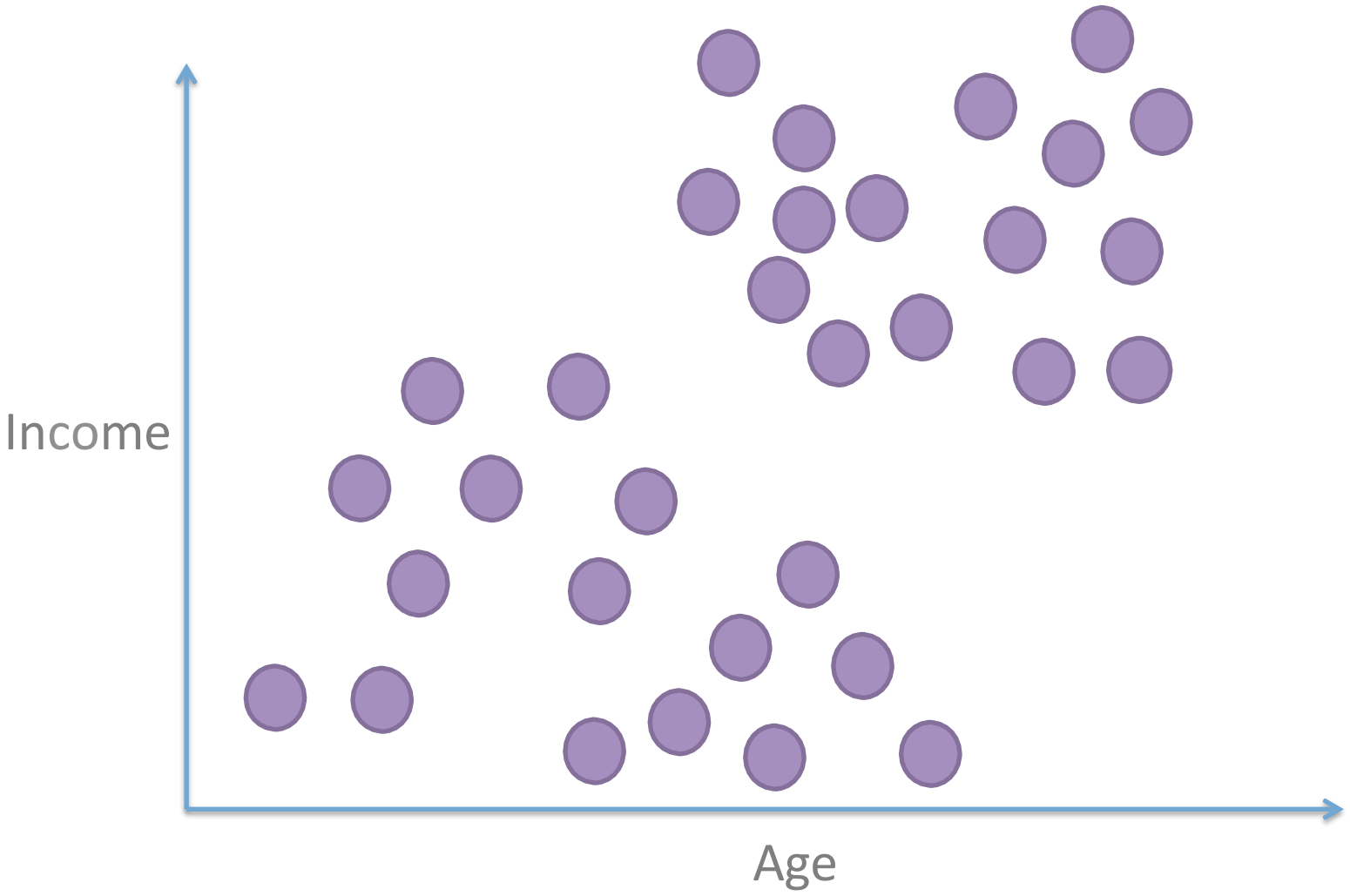


Inertia: sum of square distances in each cluster (low inertia = high density)

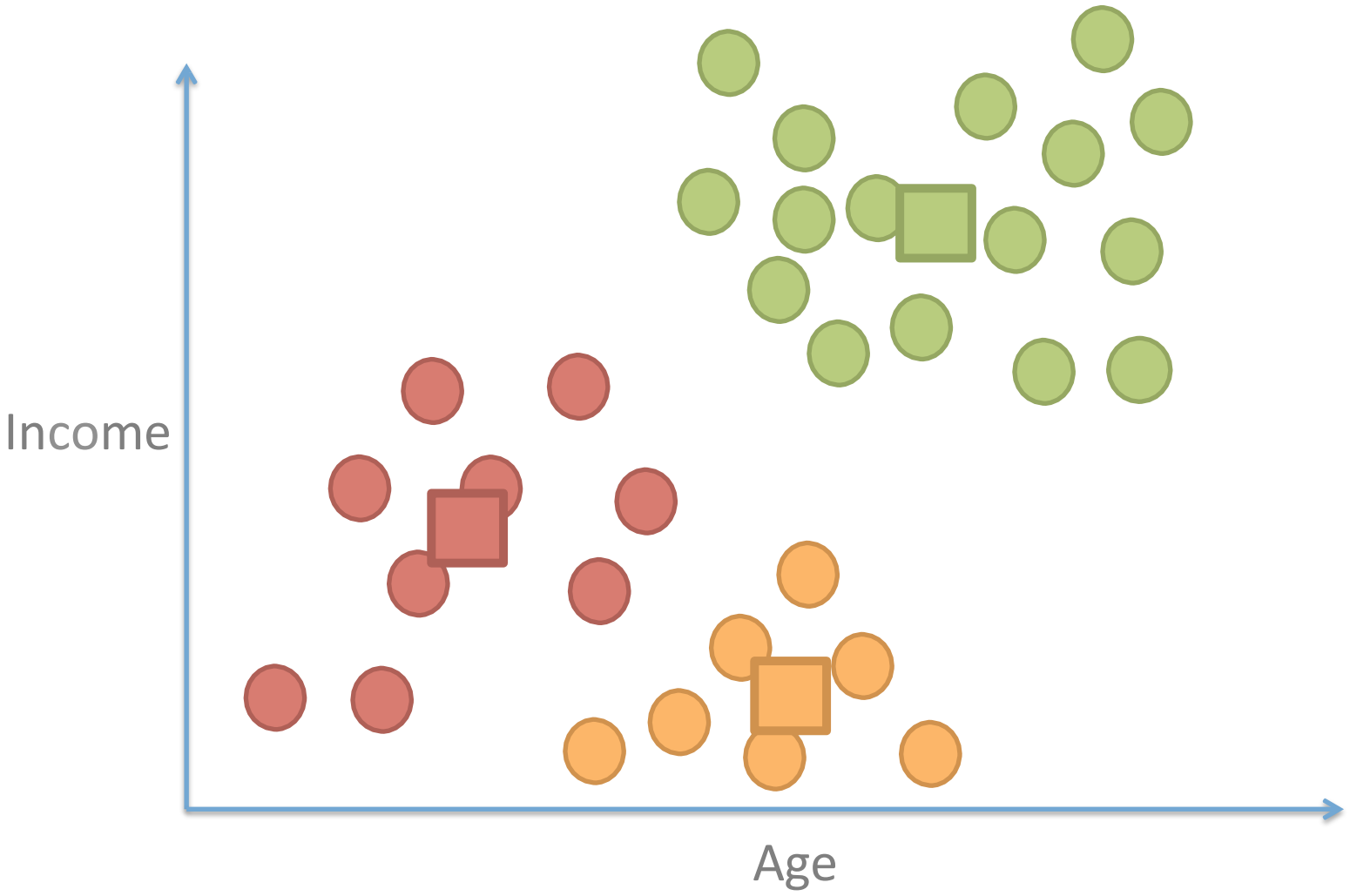


Initiate at random a bunch of times,

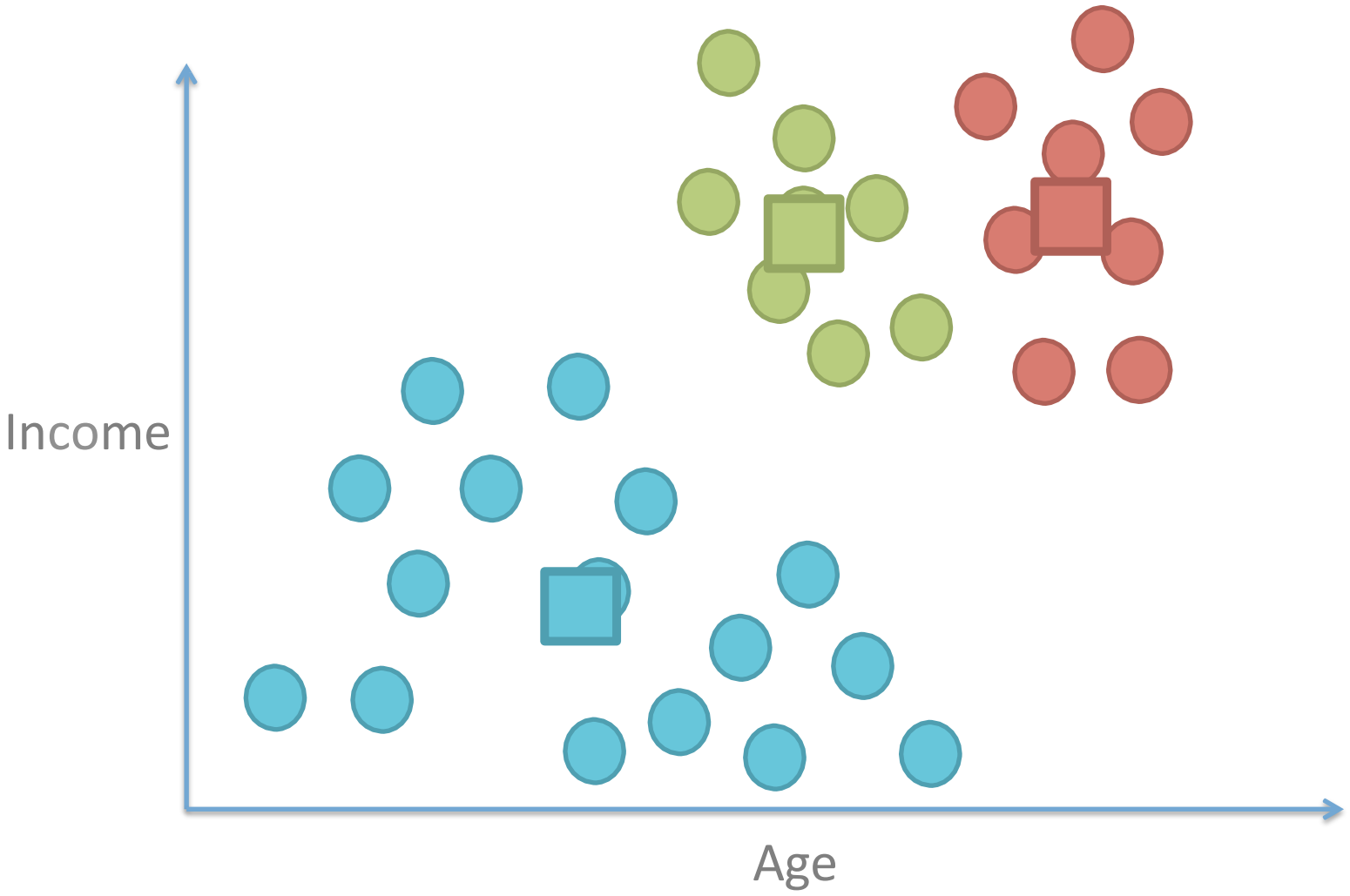
Take the clustering with the best score!



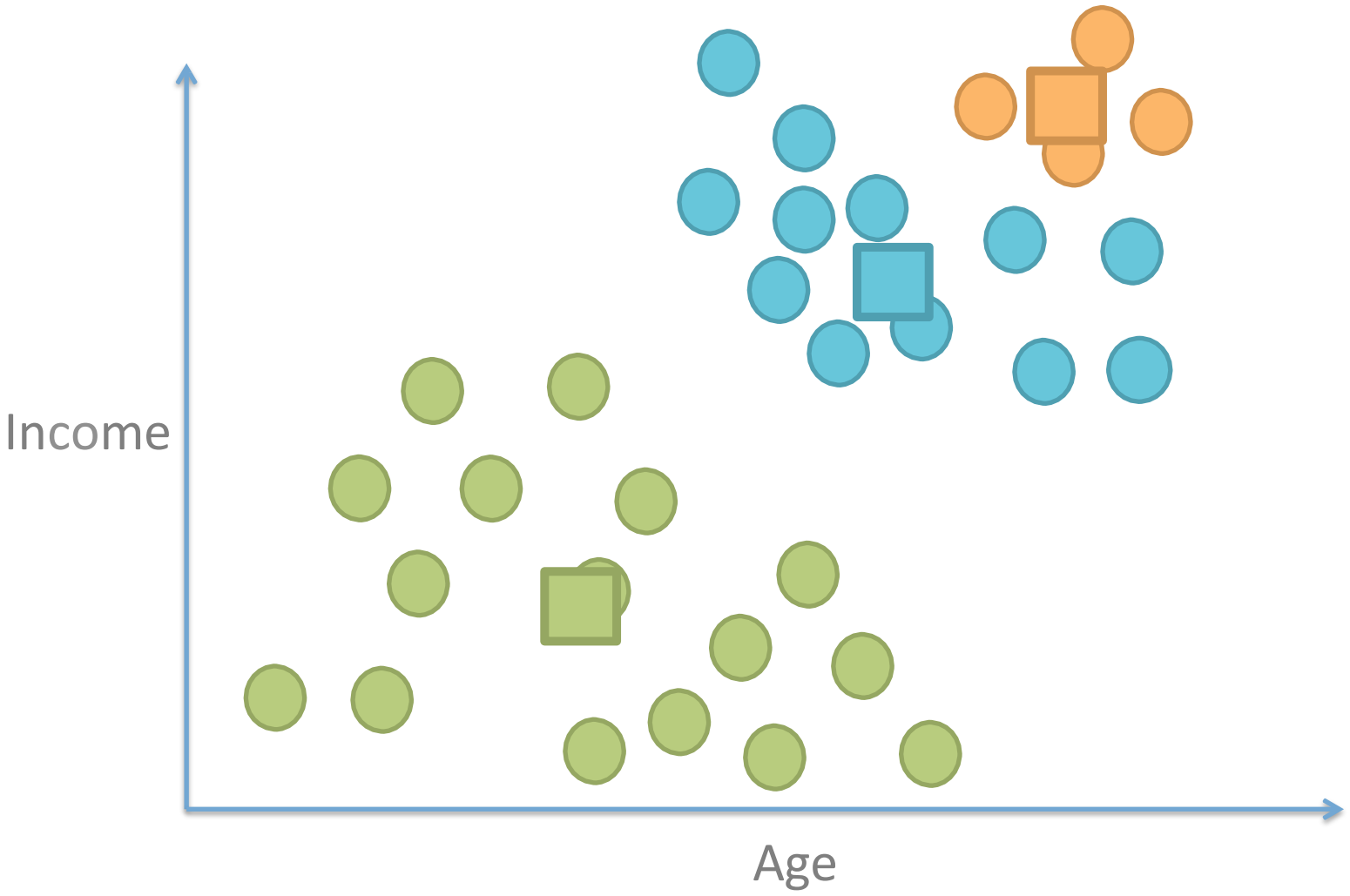
Inertia = 12.645



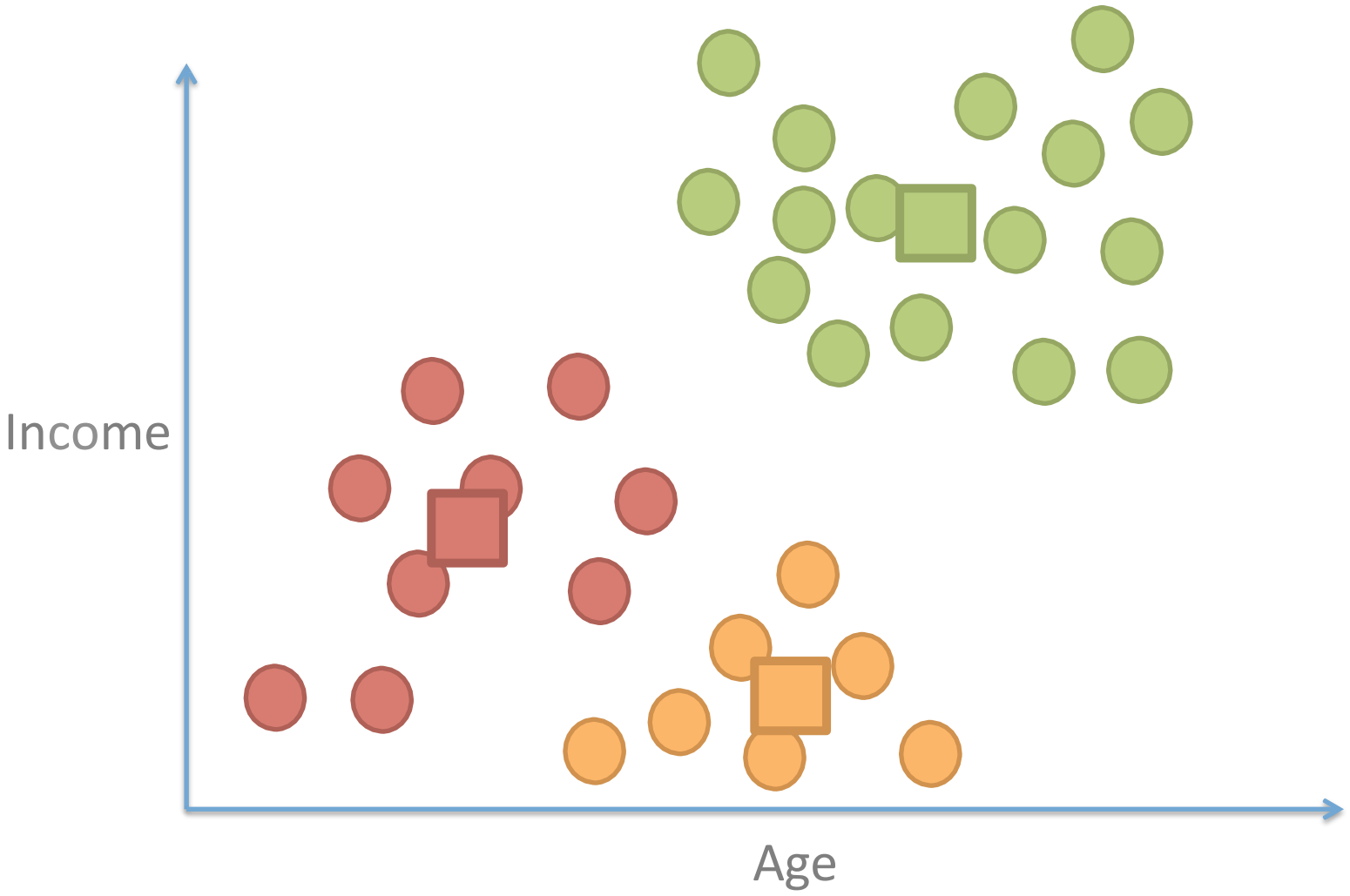
Inertia = 12.943



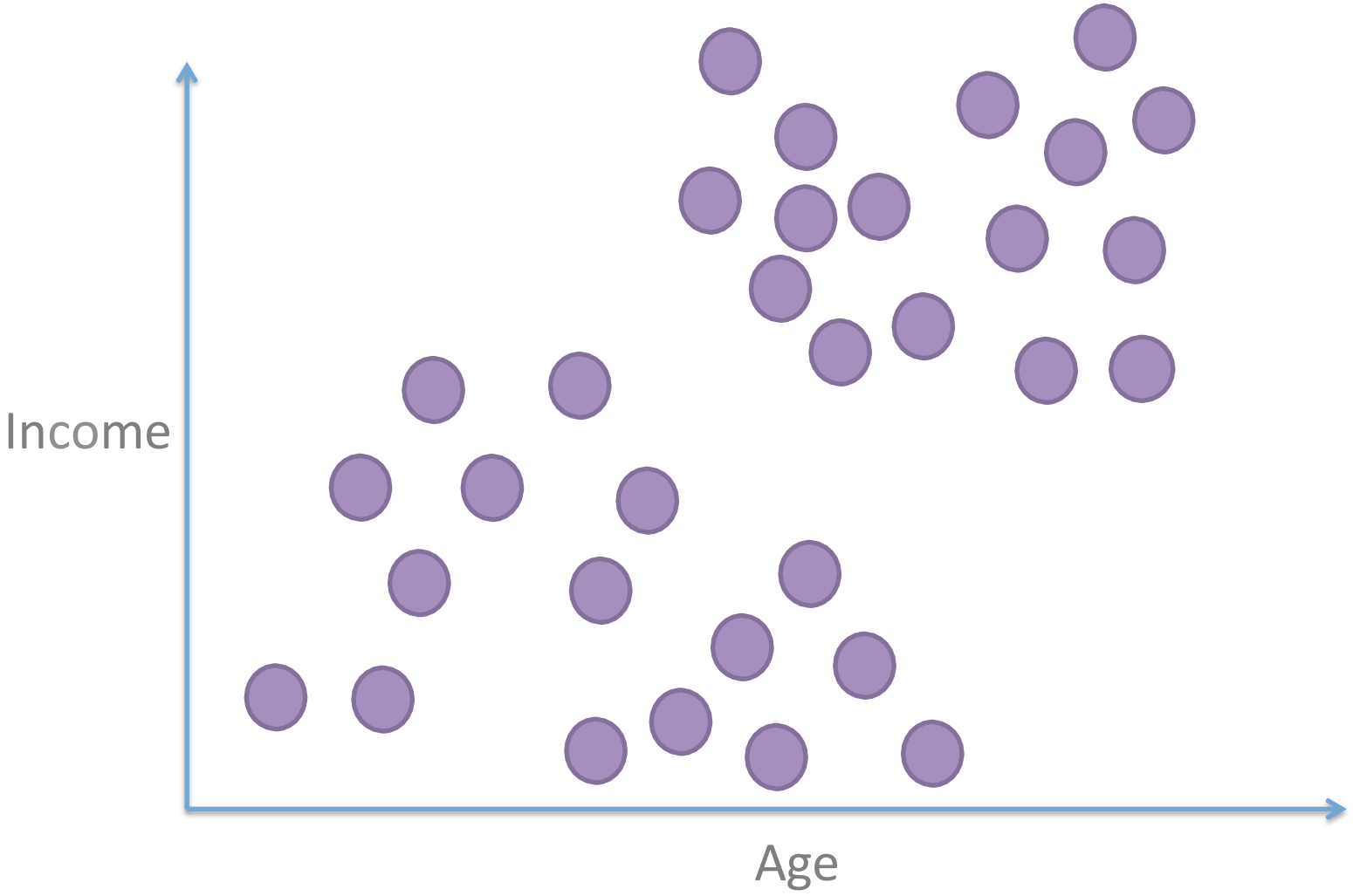
Inertia = 13.112



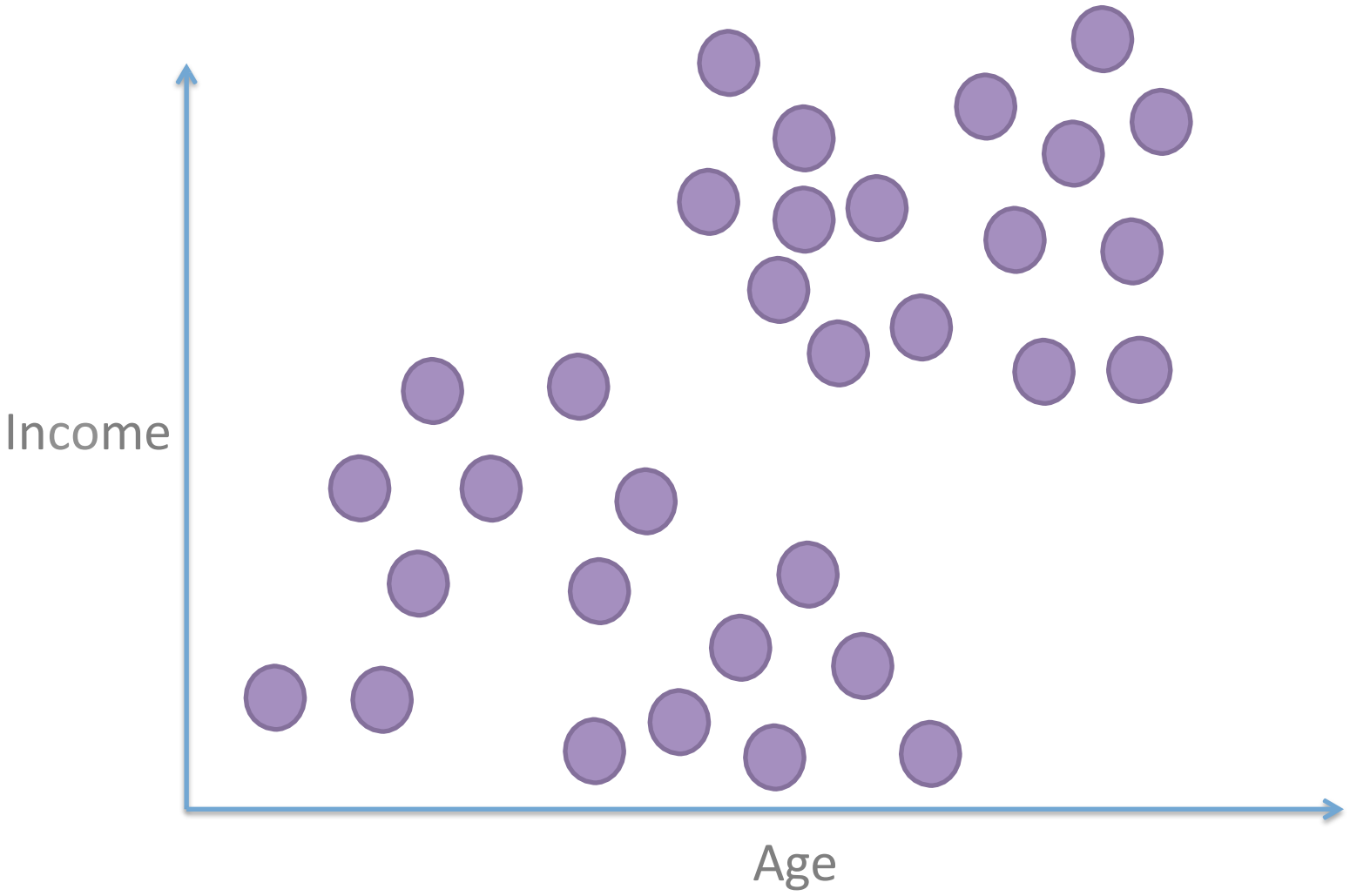
Inertia = 12.645 ← MIN



With higher dimensions, I may have to try a lot! Smarter way to initialize?

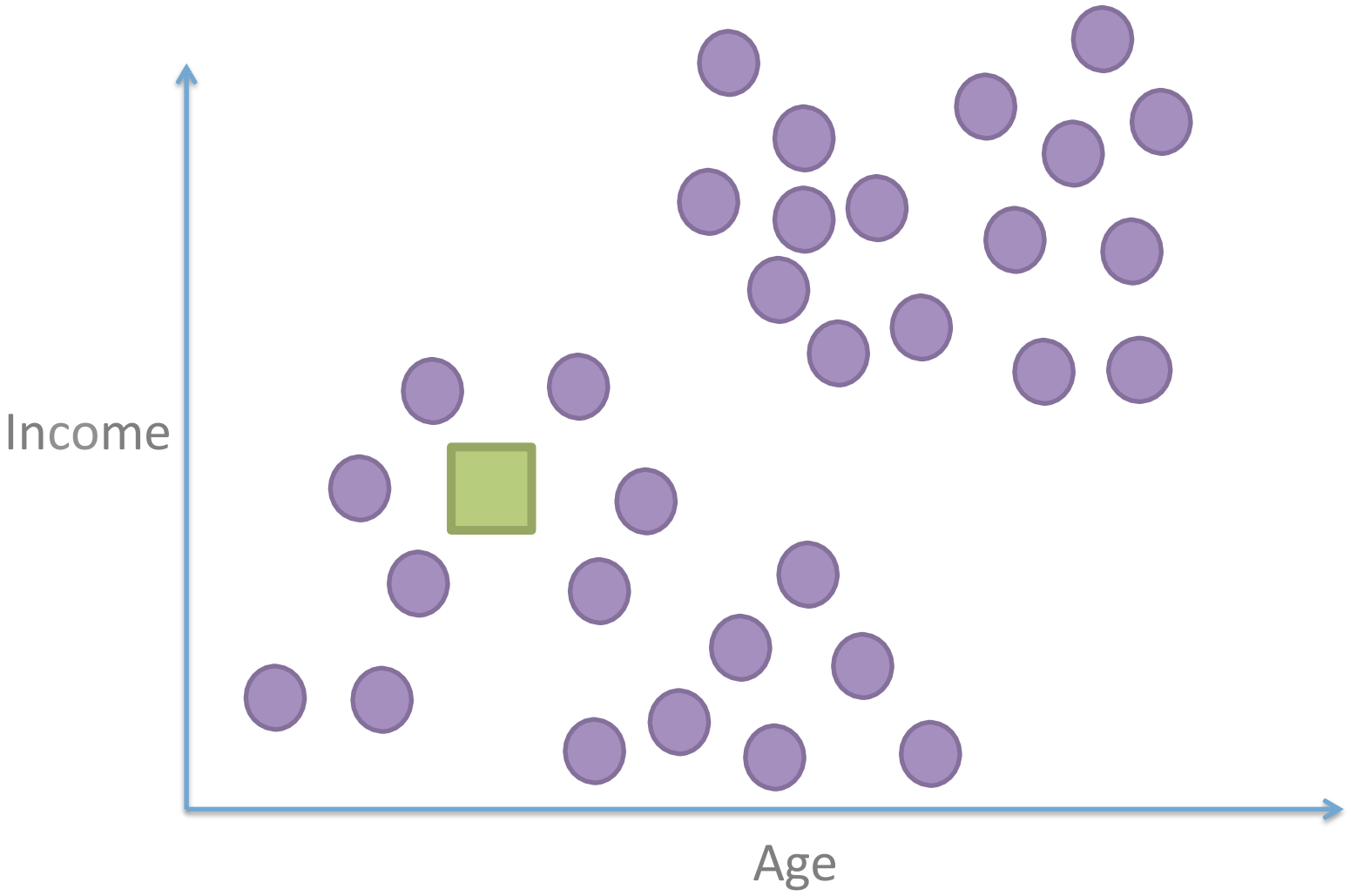


K-means++



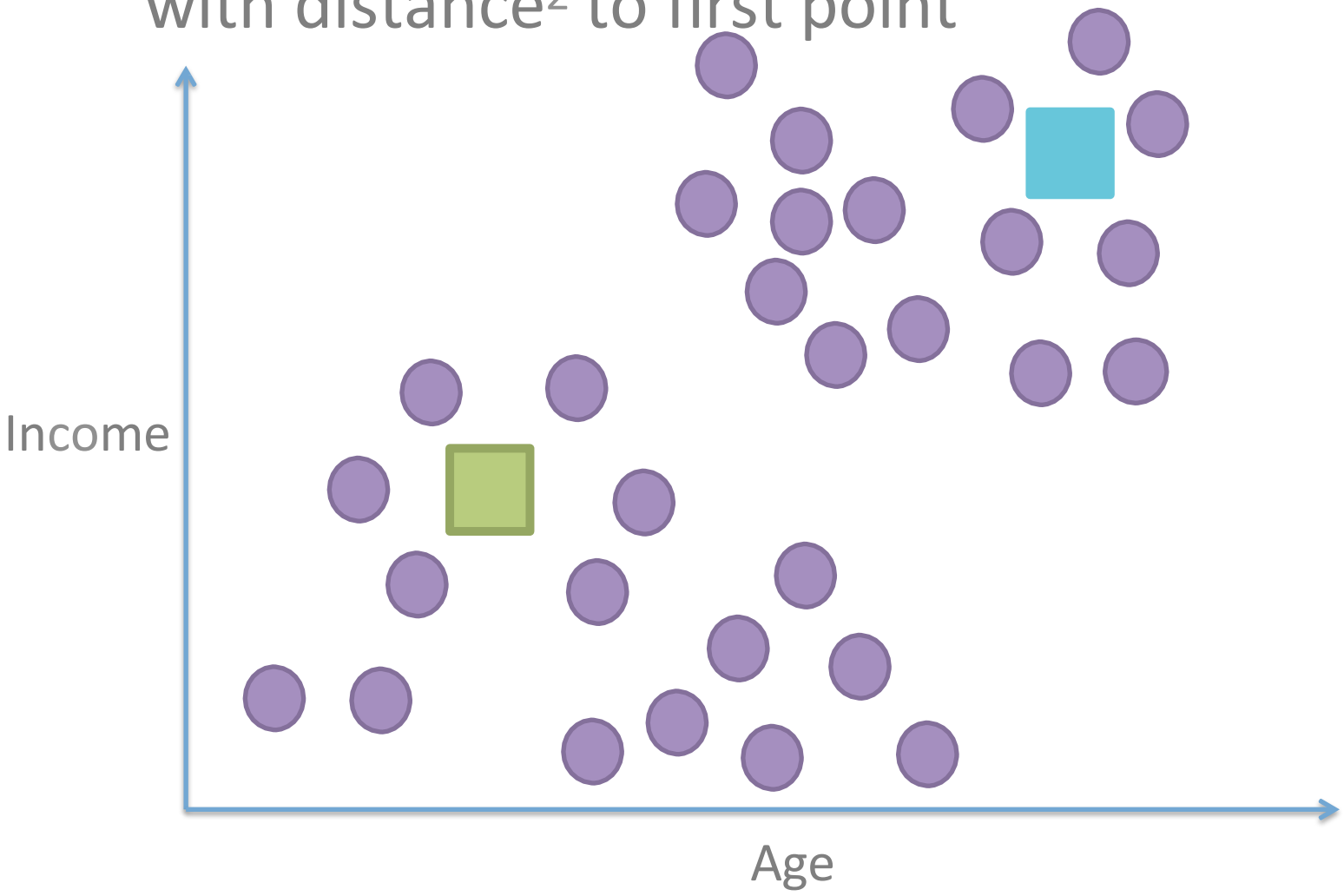
K-means++

Pick one point at random as initial point



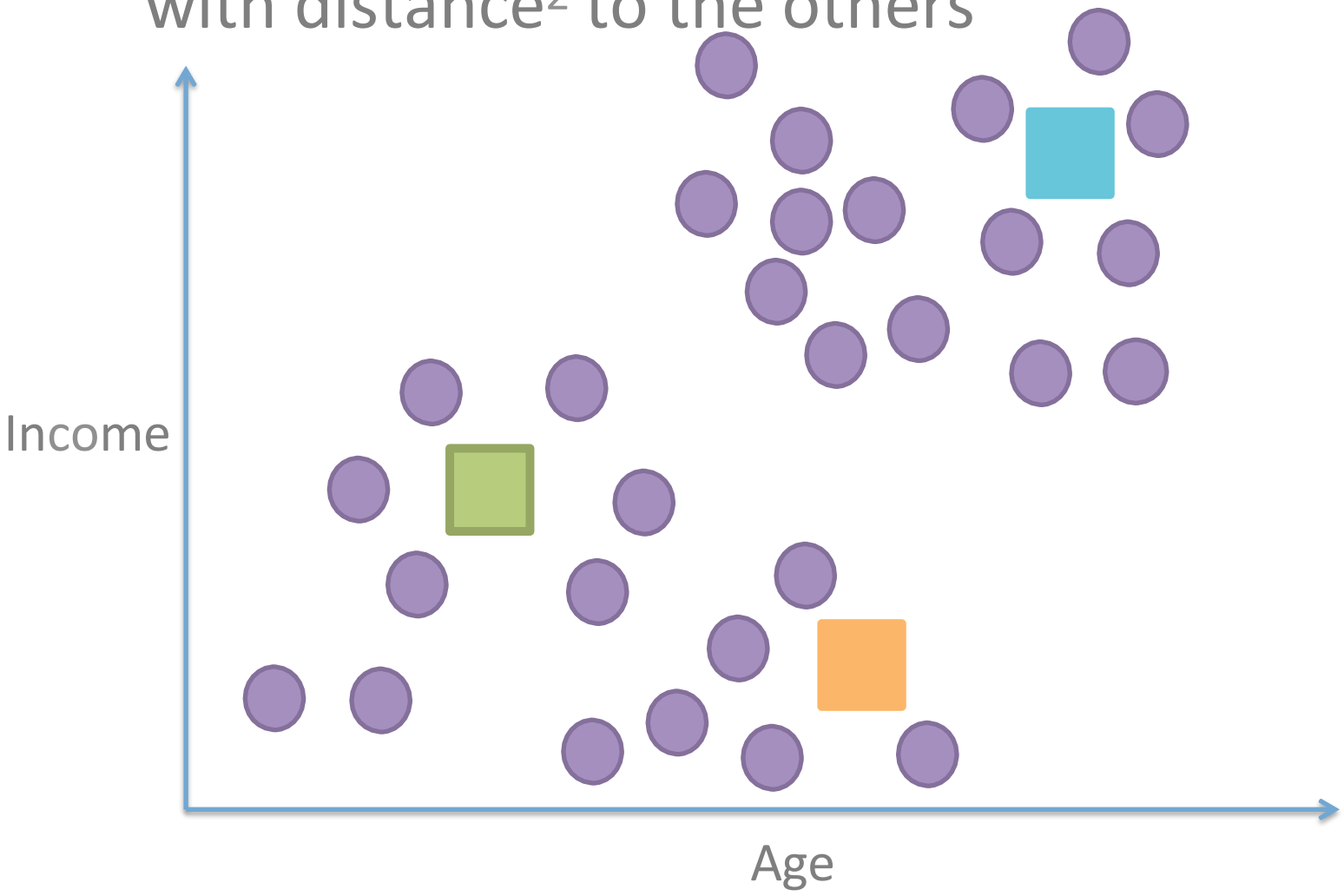
K-means++

Pick next point with prob increasing with distance^2 to first point



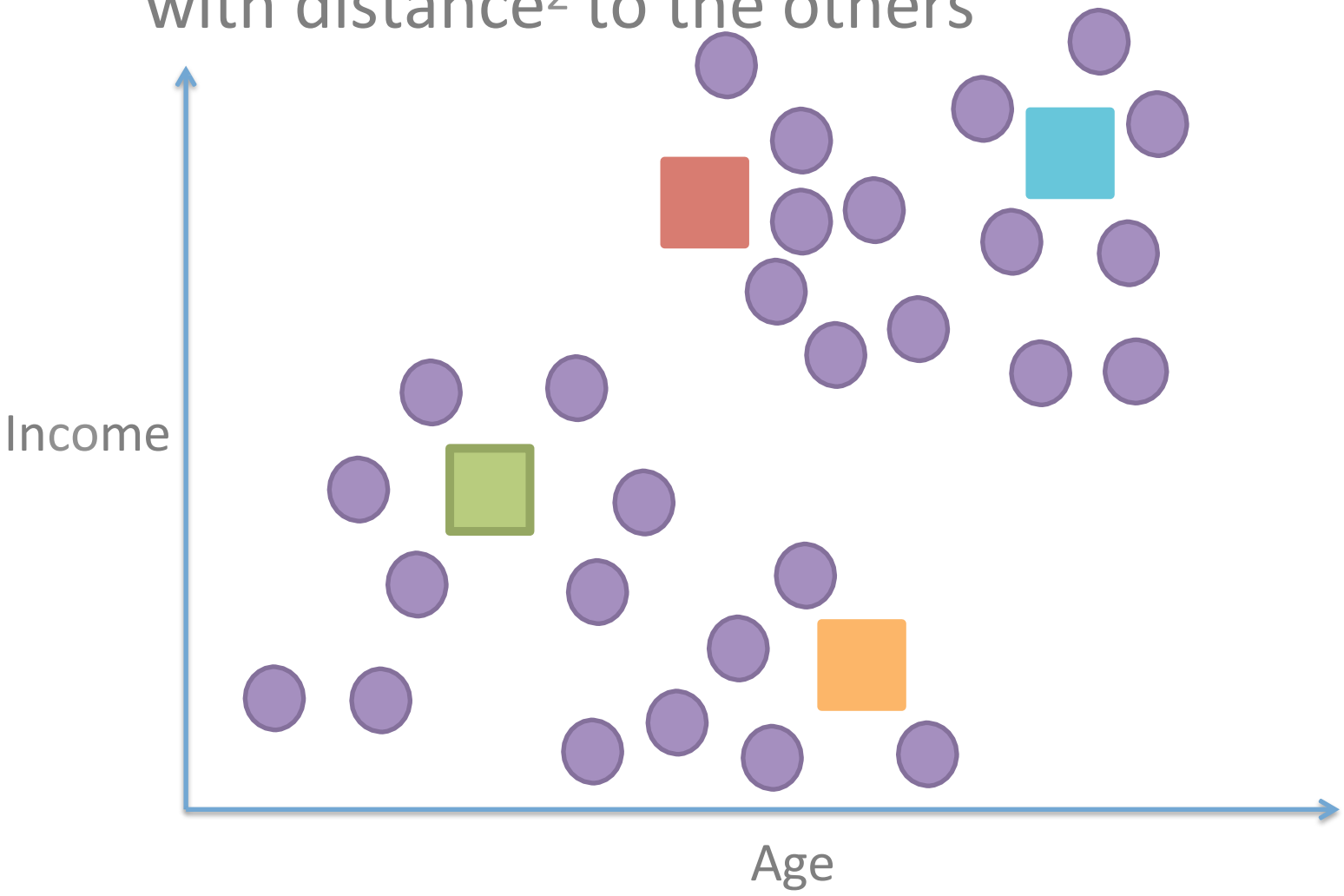
K-means++

Pick next point with prob increasing with distance^2 to the others



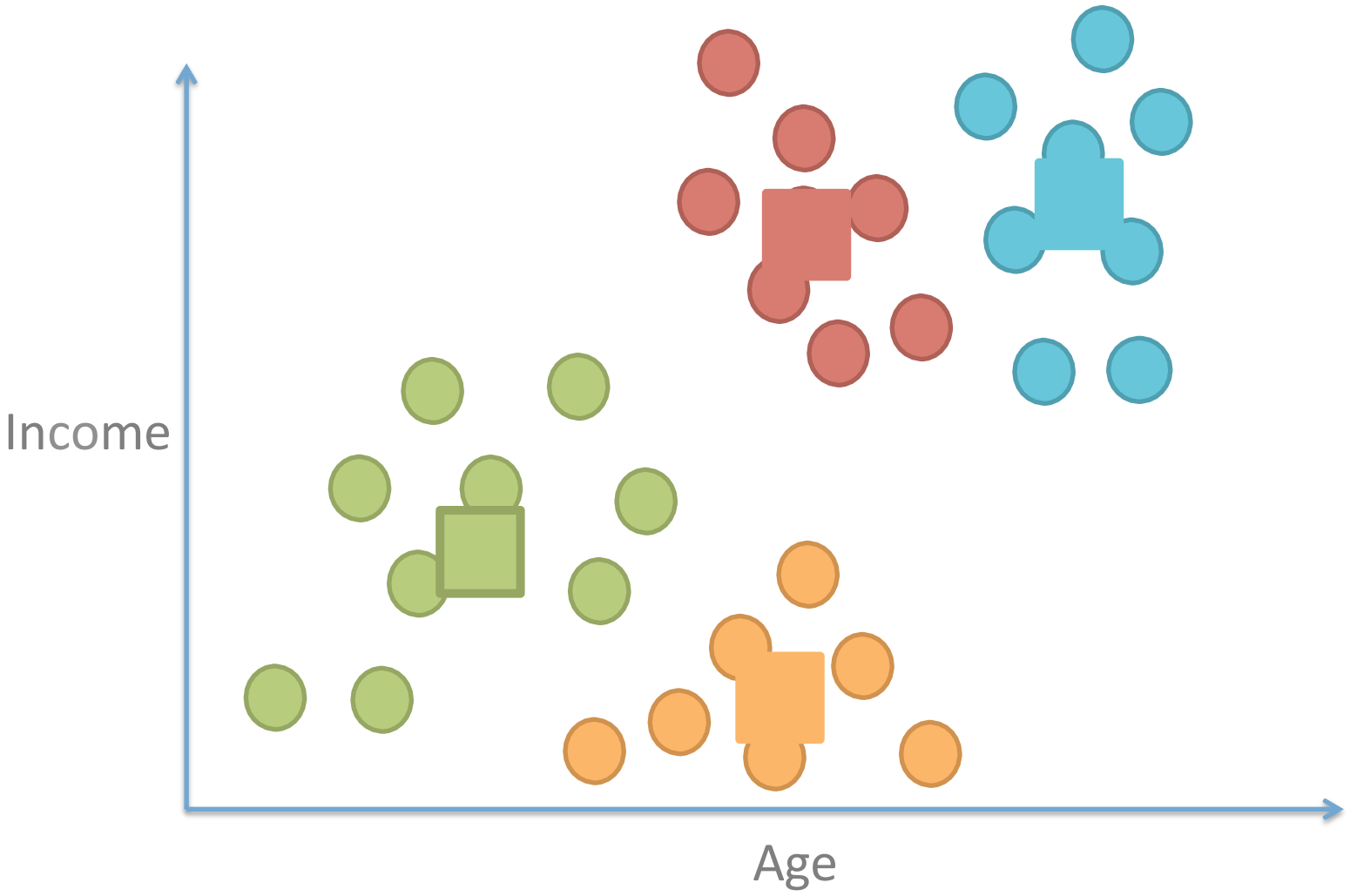
K-means++

Pick next point with prob increasing with distance^2 to the others



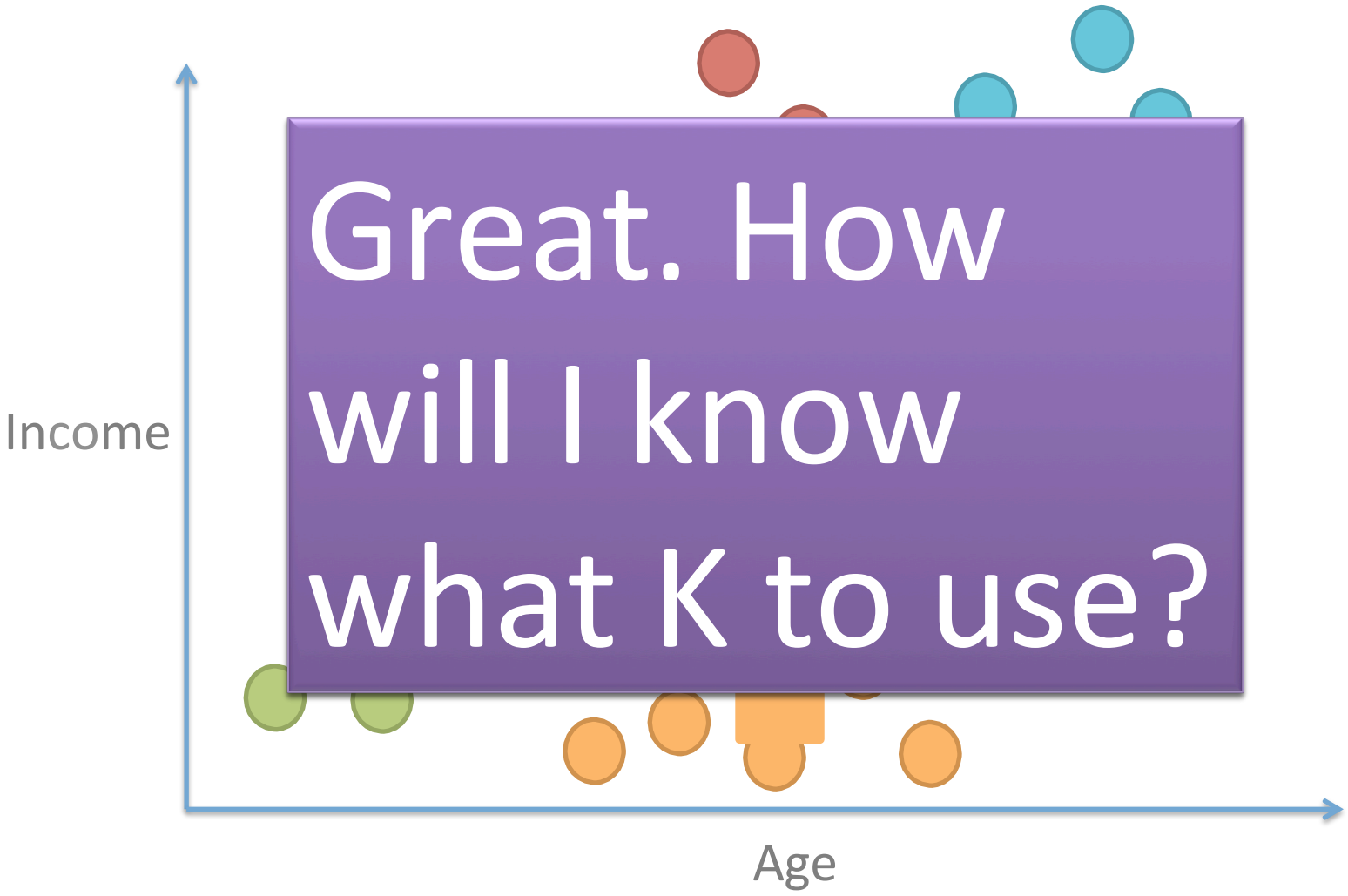
K-means++

Now proceed with regular K-means



K-means++

Now proceed with regular K-means



Sometimes the question has a K

I have 4 CPU cores. I need to cluster similar jobs so I can send each group to one CPU (K=4)

Sometimes the question has a K

I have 4 CPU cores. I need to cluster similar jobs so I can send each group to one CPU (K=4)

We want to make jeans that can cover many different body shapes. We took a lot of measurements. Come up with 10 different sizes to cover most people (K=10)

Sometimes the question has a K

I have 4 CPU cores. I need to cluster similar jobs so I can send each group to one CPU (K=4)

We want to make jeans that can cover many different body shapes. We took a lot of measurements. Come up with 10 different sizes to cover most people (K=10)

I want to build a navigation interface for browsing scientific papers. My design dictates dividing them into 20 disciplines (K=20)

Sometimes the question has a K

I have 4 CPU cores. I need to cluster similar jobs so I can send each group to one CPU (K=4)

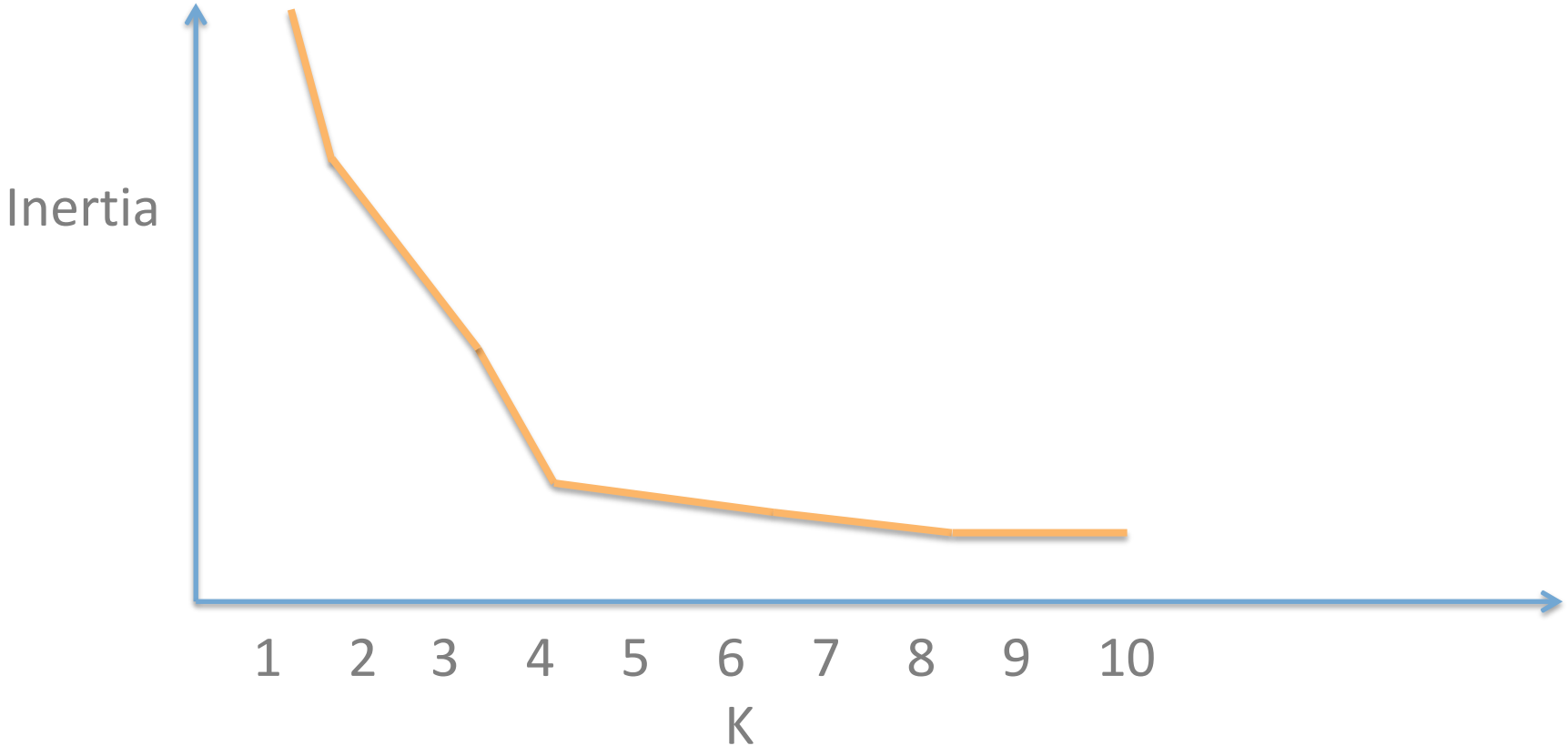
We wa… many differe… f measu… rent sizes to cove…

Yeah, I don't really have that. :/

I want to build a navigation interface for browsing scientific papers. My design dictates dividing them into 20 disciplines (K=20)

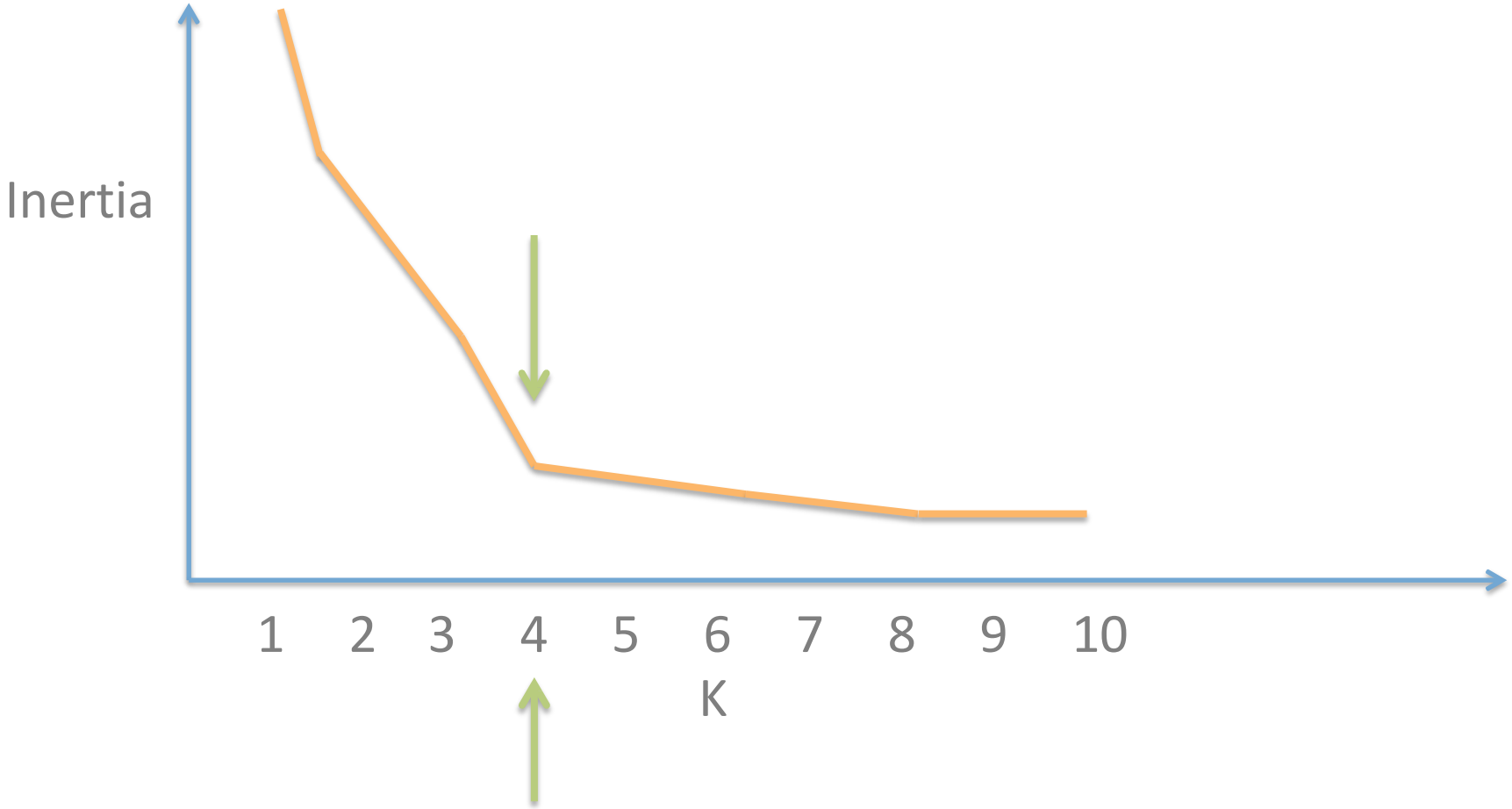
Our score: Inertia

Higher within-cluster density with higher K, inertia will go down



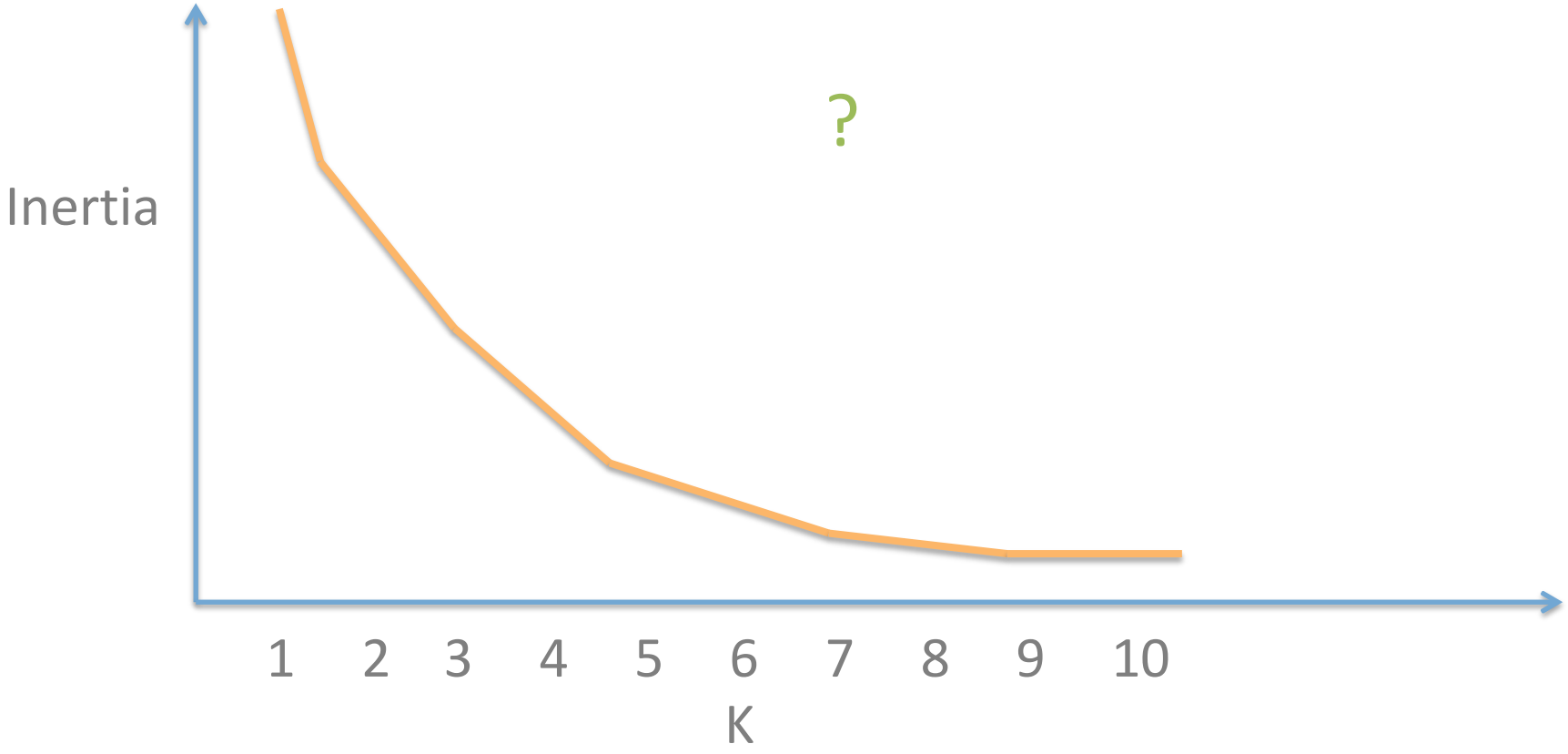
Our score: Inertia

Higher within-cluster density with higher K, inertia will go down



Our score: Inertia

Higher within-cluster density with higher K, inertia will go down



Even Better: Silhouette Score

- Like F1 Score, tracks a balance of two competing metrics
- Is a more objective, standardized value than “finding the elbow in the curve” with intertia.
- SKLearn has other options if ground truth labels are known or determinable.

For anything with distances,
scaling is very important!

sklearn.preprocessing.scale(X)

```
from sklearn.cluster import Kmeans


model = Kmeans.fit(X)
clusters = model.predict(X)

or...


clusters = Kmeans.fit_predict(X)
```