

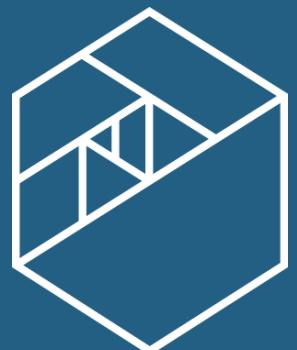
# Learning from Text: Natural Language Processing with Python

ODSC East  
May 3, 2017

---

Michelle L. Gill, Ph.D.  
Senior Data Scientist, Metis  
[michelle@thisismetis.com](mailto:michelle@thisismetis.com)

[github.com/mlgill/ODSC\\_East\\_2017\\_PythonNLP](https://github.com/mlgill/ODSC_East_2017_PythonNLP)

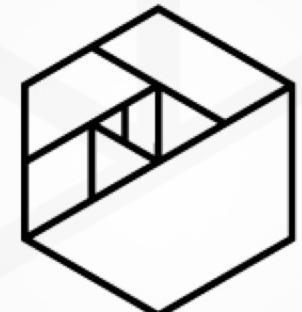


METIS

# Metis Data Science Training

- Data Science Bootcamp
  - 12 Week, In-Person
  - New York, San Francisco, Chicago, Seattle
- Corporate Training
  - Python for Data Science
  - Machine Learning
  - Natural Language Processing
  - Spark
- Evening Professional Development Courses
- Explore Data Science Online Training

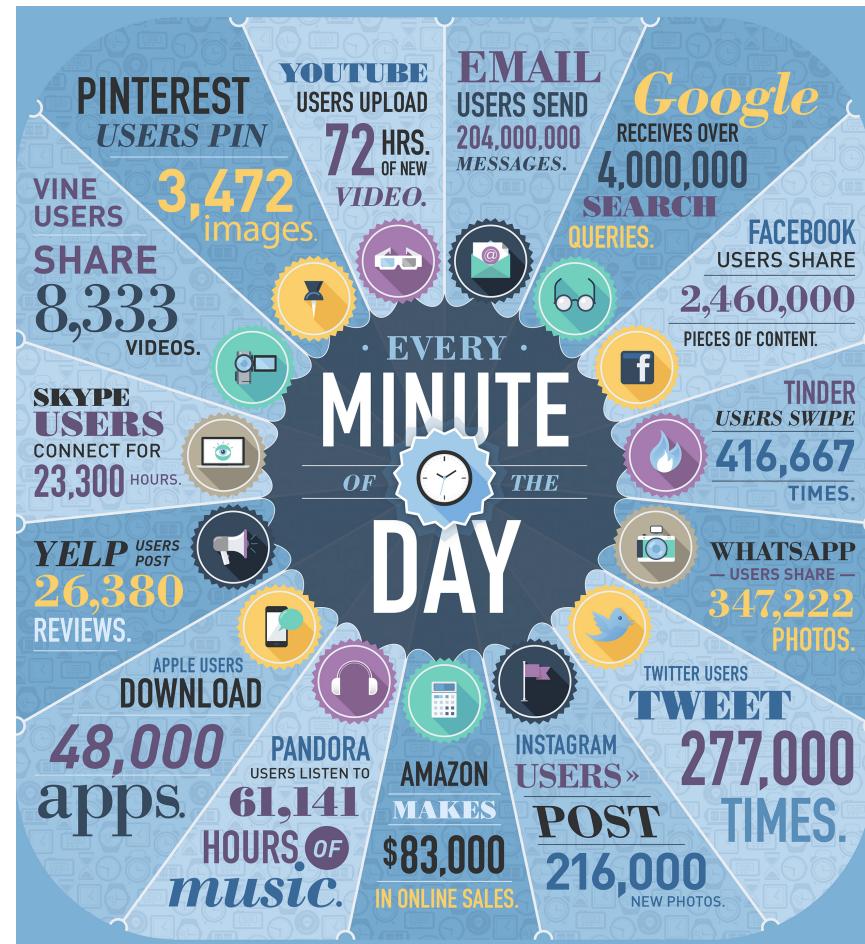
[thisismetis.com](http://thisismetis.com)



# Data Generated Each Minute

Much of this data is text

- Email
- Text messages
- News articles
- Blogs
- Twitter
- Reviews
- 204,000,000 Emails
- 4,000,000 Google Search Queries
- 277,000 Tweets
- 26,380 Yelp Reviews



<https://www.domo.com/blog/2014/04/data-never-sleeps-2-0/>



# How is this Useful?

- Information retrieval
- Classification and clustering
- Sentiment analysis
- Recommendation systems



# How to Represent Text?

Answer

For machine learning tasks  
structured, numeric data is desired

Problem

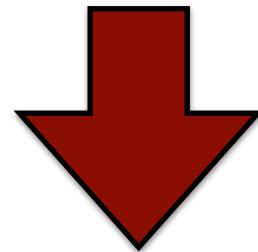
Text is unstructured and  
contains limited numeric values



# Creating Numerical Features

- Split text into words  
(tokenization)

**“This is an example”**



**[This, is, an, example]**



# Creating Numerical Features

- Split text into words  
(tokenization)
- Numerically encode words  
(one-hot encoding)

<b>This</b>	= [1, 0, 0, 0]
<b>is</b>	= [0, 1, 0, 0]
<b>an</b>	= [0, 0, 1, 0]
<b>example</b>	= [0, 0, 0, 1]



# Creating Numerical Features

- Split text into words  
(tokenization)
- Numerically encode words  
(one-hot encoding)
- Represent documents by their numerical tokens

	This	is	an	example
Doc 1	1	1	1	1
Doc 2	1	1	0	0
Doc 3	0	1	1	0



# Creating Numerical Features with Python

Code

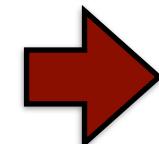
```
import pandas as pd
from sklearn.feature_extraction.text
    import CountVectorizer

corpus = ['This is the first document.',
          'This is the second document.',
          'And the third one.']

cv = CountVectorizer()
X = cv.fit_transform(corpus)
pd.DataFrame(X.toarray(),
              columns=cv.get_feature_names())
```

Output

	and	document	first	is	one	second	the	third	this	
0	0		1	1	1	0	0	1	0	1
1	0		1	0	1	0	1	1	0	1
2	1		0	0	0	1	0	1	1	0



Count Vectorizer



# Is That All?

- How much information is contained in a word?
- Can more information be gained?
- What assumptions are being made?



# Preprocessing Text

- How to tokenize: single word? multiple words?
- Remove: capital letters, punctuation, common words
- Word order and co-occurrence
- Convert to roots: running --> ran
- Misspellings
- Different languages
- Weight words equally or differently?



# Preprocessing Examples

- Common words such as "the", "I", and "a" can be removed
- These are called "stop words"

```
ex = CountVectorizer(stop_words='english')
x = ex.fit_transform(['this is an example with stop words'])
pd.DataFrame(x.toarray(), columns = ex.get_feature_names())
```

example	stop	words
1	1	1



# Preprocessing Examples

- Words can be reduced to their root form
- This is called "stemming" or "lemmatization"

```
import nltk
stemmer = nltk.stem.porter.PorterStemmer()
print stemmer.stem('run')
run
print stemmer.stem('running')
run
```



# Preprocessing Examples

- Word order can be captured by joining words together
- These are called "n-grams"

```
text = ['this is a bigram example']
bigram_vectorizer = CountVectorizer(ngram_range=(2,2))
X = bigram_vectorizer.fit_transform(text)
pd.DataFrame(X.toarray(), columns=bigram_vectorizer.get_feature_names())
```

<b>bigram example</b>	<b>is bigram</b>	<b>this is</b>
1	1	1



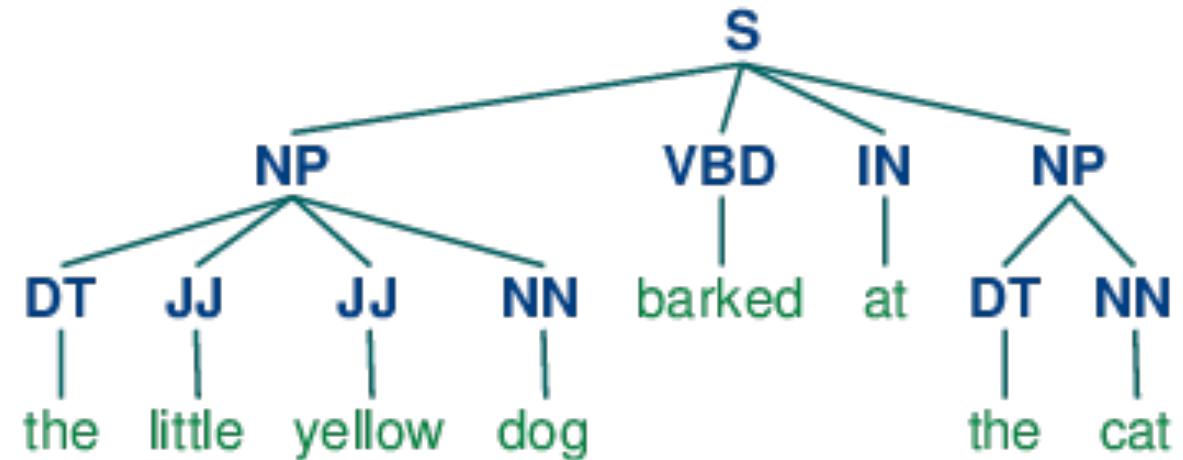
# Other Preprocessing Considerations

- Presupposition: implicit assumptions
  - Jane no longer writes fiction (Jane once wrote fiction)
- Word relationships:
  - Substitution: red and blue, Saturday and Sunday
  - Co-occurrences: car and drive
- Sparsity
  - High dimensionality
  - Similarity and meaning



# Other Preprocessing Considerations

- Chunking segments and labels sentences
- Like parts-of-speech tagging with sentence structure included



# Practice Preprocessing

Walkthrough

01\_Text\_Preprocessing\_Walkthrough

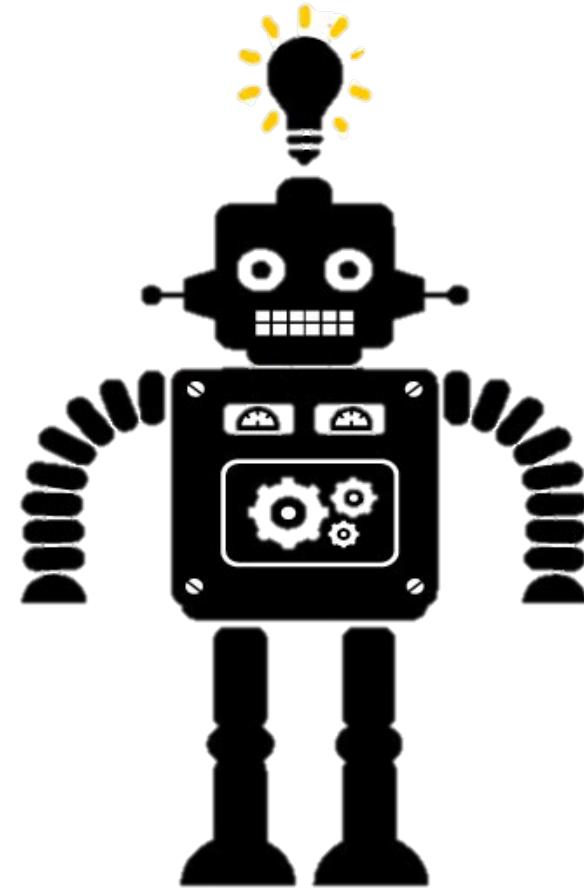
Exercises

02\_Text\_Preprocessing\_Exercises



# What is Machine Learning?

Machine learning allows computers to learn and infer from data.



# Types of Machine Learning

Supervised

Data points have known outcome (prediction)

- **Regression**: predict continuous value
- **Classification**: predict categorical value

Unsupervised

Data points have unknown outcome (find structure)

- **Clustering**: group observations
- **Dimensionality reduction**: reduce features



# Machine Learning Vocabulary

- Features
- Observations
- Labels



sepal length	sepal width	petal length	petal width	species
6.7	3.0	5.2	2.3	virginica
6.4	2.8	5.6	2.1	virginica
4.6	3.4	1.4	0.3	setosa
6.9	3.1	4.9	1.5	versicolor
4.4	2.9	1.4	0.2	setosa
4.8	3.0	1.4	0.1	setosa
5.9	3.0	5.1	1.8	virginica
5.4	3.9	1.3	0.4	setosa
4.9	3.0	1.4	0.2	setosa
5.4	3.4	1.7	0.2	setosa



# Machine Learning Vocabulary

- Features
- Observations
- Labels



sepal length	sepal width	petal length	petal width	species
6.7	3.0	5.2	2.3	virginica
6.4	2.8	5.6	2.1	virginica
4.6	3.4	1.4	0.3	setosa
6.9	3.1	4.9	1.5	versicolor
4.4	2.9	1.4	0.2	setosa
4.8	3.0	1.4	0.1	setosa
5.9	3.0	5.1	1.8	virginica
5.4	3.9	1.3	0.4	setosa
4.9	3.0	1.4	0.2	setosa
5.4	3.4	1.7	0.2	setosa



# Machine Learning Vocabulary

- Features
- Observations
- Labels

sepal length	sepal width	petal length	petal width	species
6.7	3.0	5.2	2.3	virginica
6.4	2.8	5.6	2.1	virginica
4.6	3.4	1.4	0.3	setosa
6.9	3.1	4.9	1.5	versicolor
4.4	2.9	1.4	0.2	setosa
4.8	3.0	1.4	0.1	setosa
5.9	3.0	5.1	1.8	virginica
5.4	3.9	1.3	0.4	setosa
4.9	3.0	1.4	0.2	setosa
5.4	3.4	1.7	0.2	setosa



# What are the Text Features?

- Tokenized words
- N-grams
- Parts-of-speech tags
- Chunks
- Syntactic structure
- etc.



# Text Classification

- Want to predict a categorical variable
  - Example: Predict if an email is spam or not
- Given
  - X: Email text
  - Y: Spam or not spam
- Goal: Build model that predicts if a new email is spam or not



# Many Choices for Model Algorithm

- Logistic regression
- Naive Bayes
- Support vector machines
- Random forest
- Boosting



# Supervised Modeling Process

- Have historical email data with spam/not spam labels
- Preprocess the email text data
  - Tokenize
  - Remove stop words
  - Stem or lemmatize
  - Count vectorize
- Use this numerical dataset in a classification algorithm
- Given a new email, process the same way and predict



# Practice Machine Learning

Walkthrough

03\_Classification\_Walkthrough

Exercises

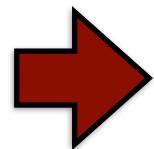
04\_Text\_Classification\_Exercises



# Beyond Count Vectorizer

Problems with binary values or word count:

- Binary counts can be too simplistic
- High counts can dominate--especially for high frequency words or long documents
- Each word is treated equally--some terms might be more important than others



Want a metric that accounts for these issues



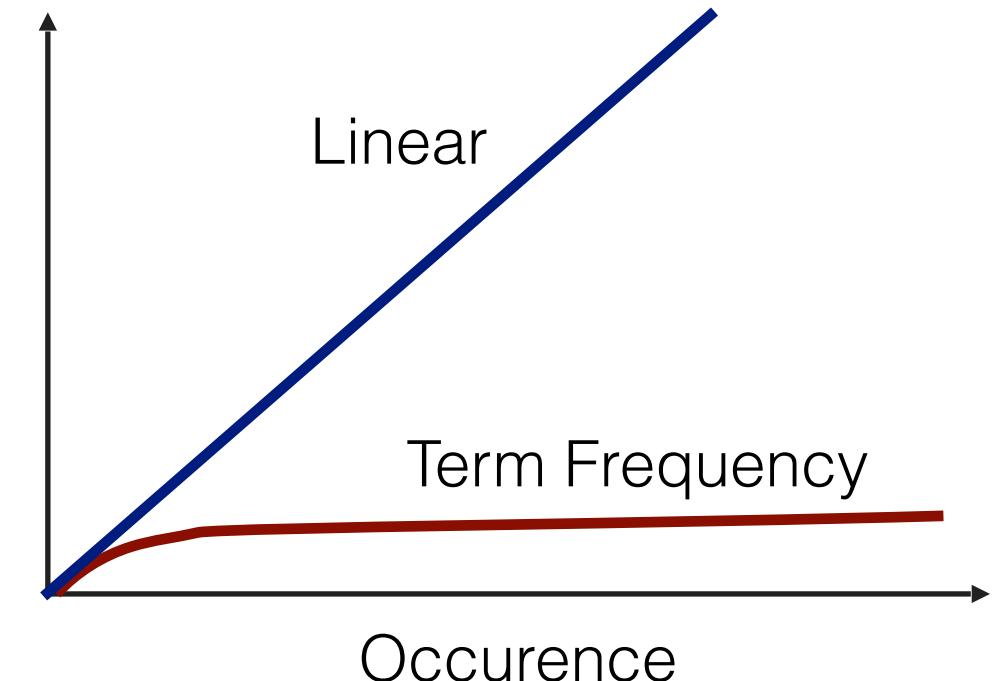
# Term Frequency (TF)

- Alternative to binary values or counting word appearance

- Balance count with a transformation:

$$tf = \log(x + 1)$$

$x$  = term occurrence



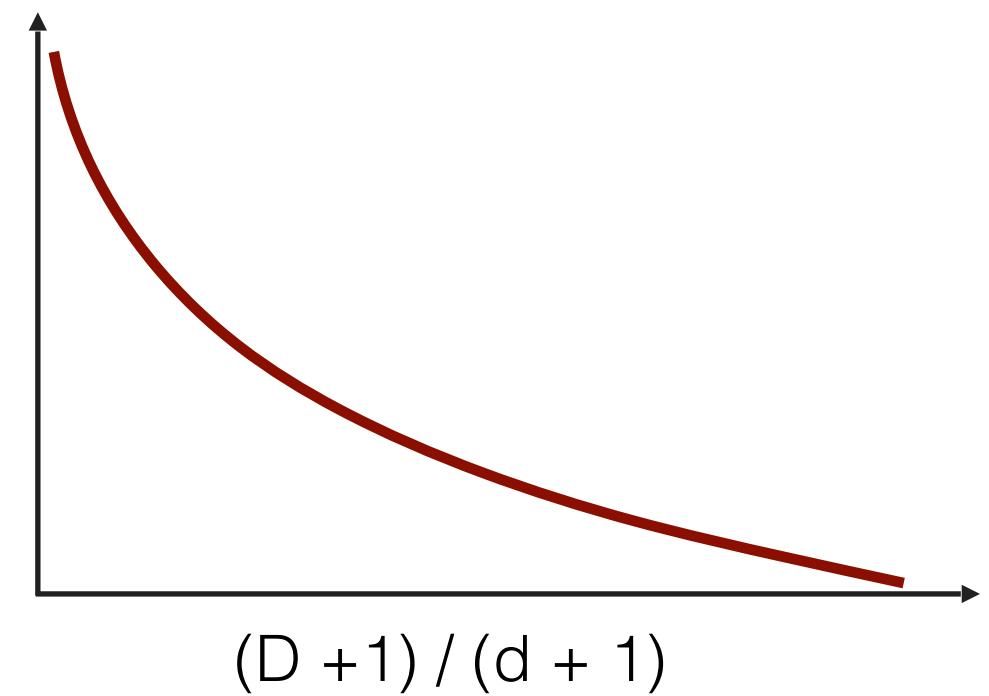
# Inverse Document Frequency (IDF)

- Also want to increase weight of words appearing in only a few documents
- Calculate inverse document frequency:

$$\text{idf} = \log(D + 1 / d + 1)$$

$D$  = total documents

$d$  = documents containing word

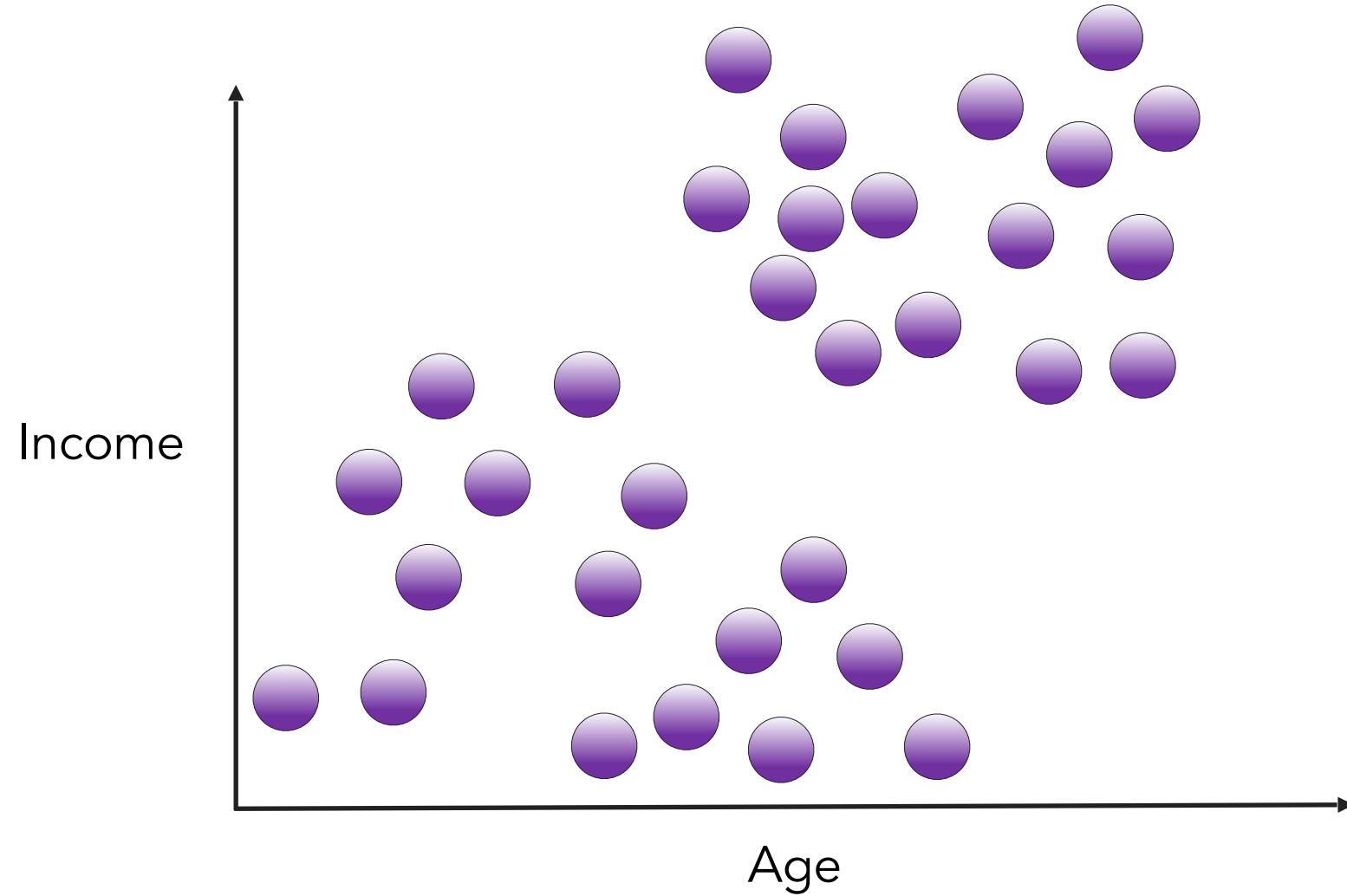


# Practice TF-IDF

- Complete question 4 from the previous exercises  
(04\_Text\_Classification\_Exercises)
- We will discuss the syntax briefly within the exercise

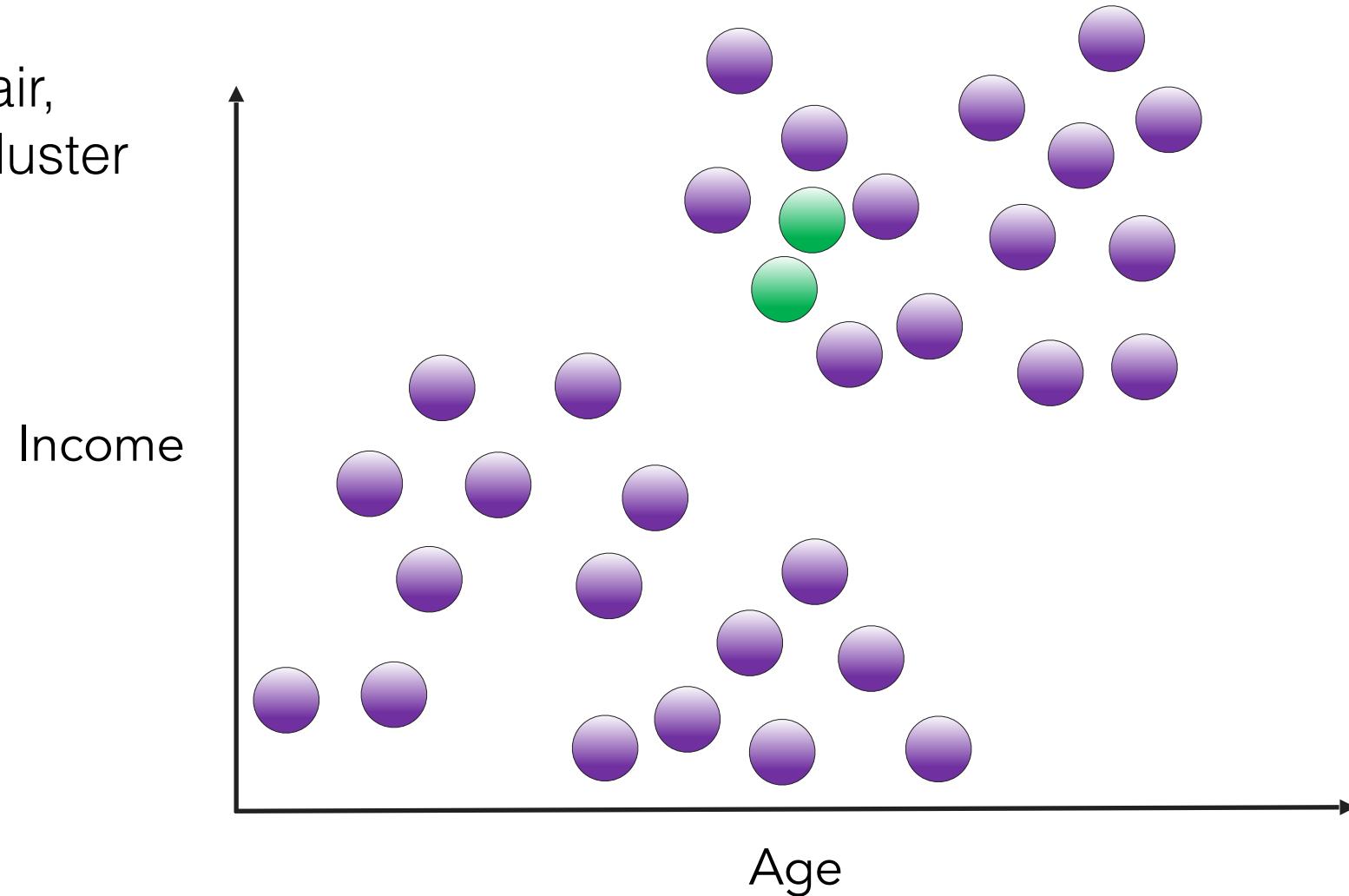


# Clustering



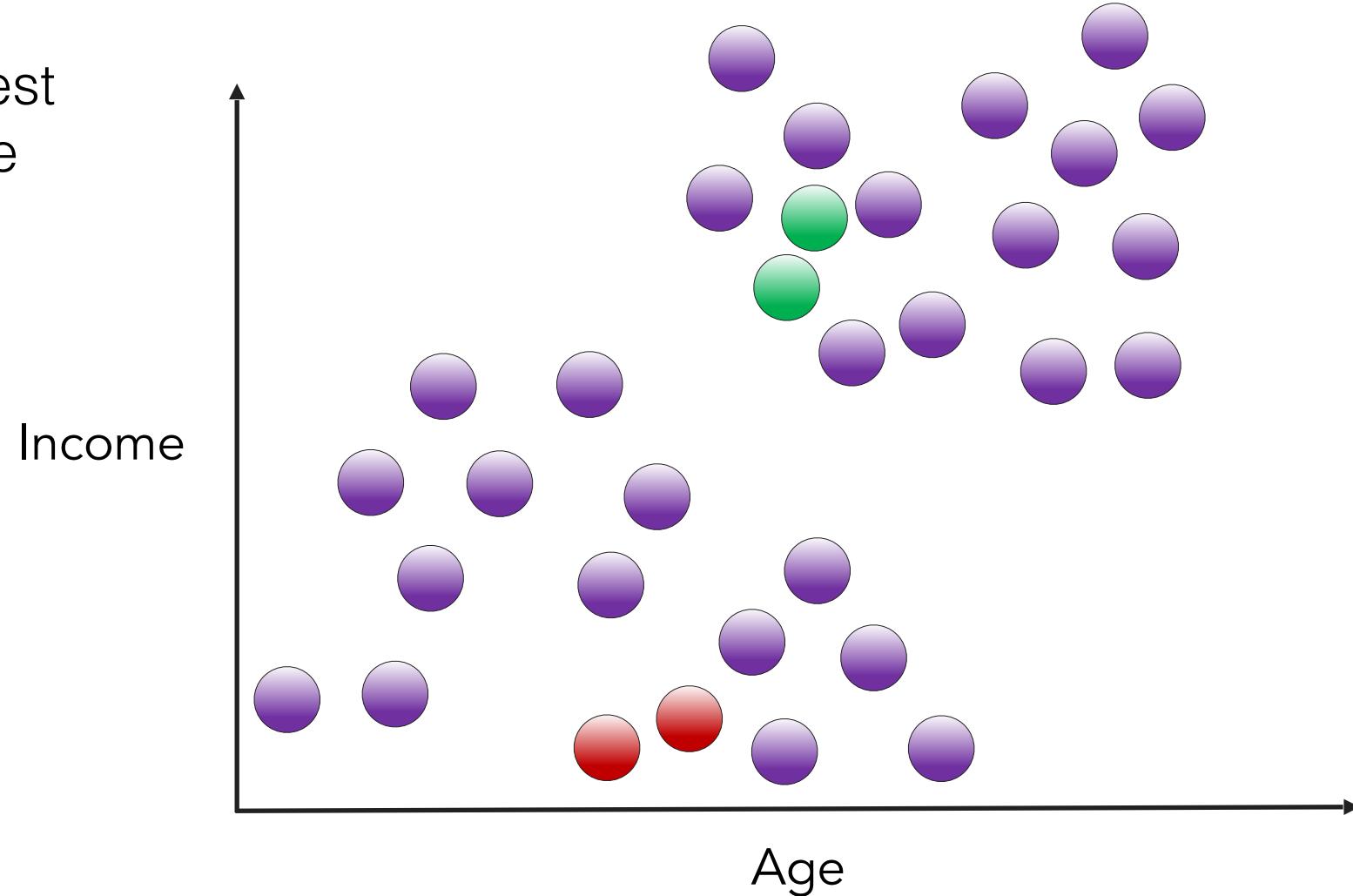
# Clustering

Find closest pair,  
merge into a cluster



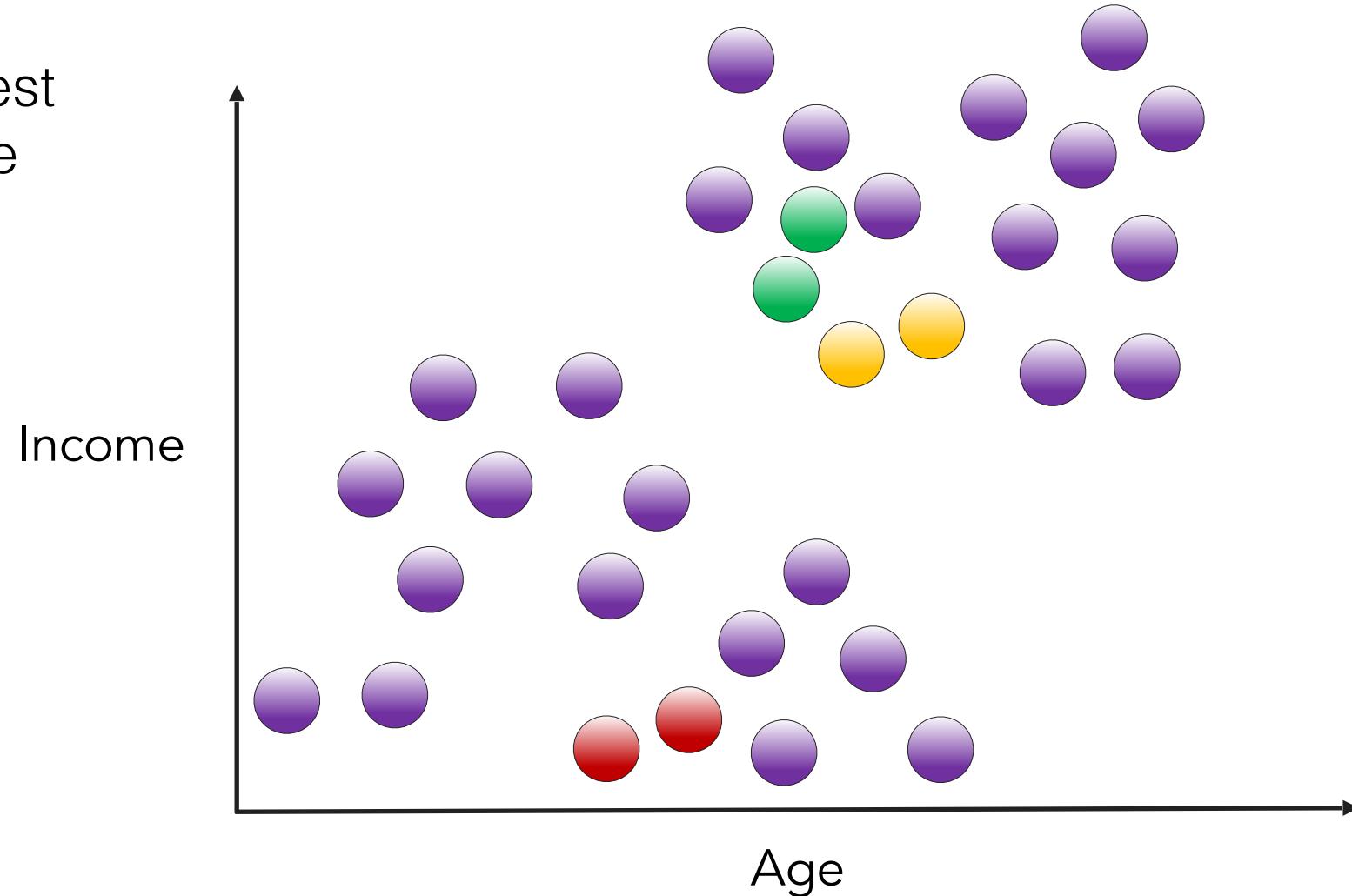
# Clustering

Find next closest pair and merge



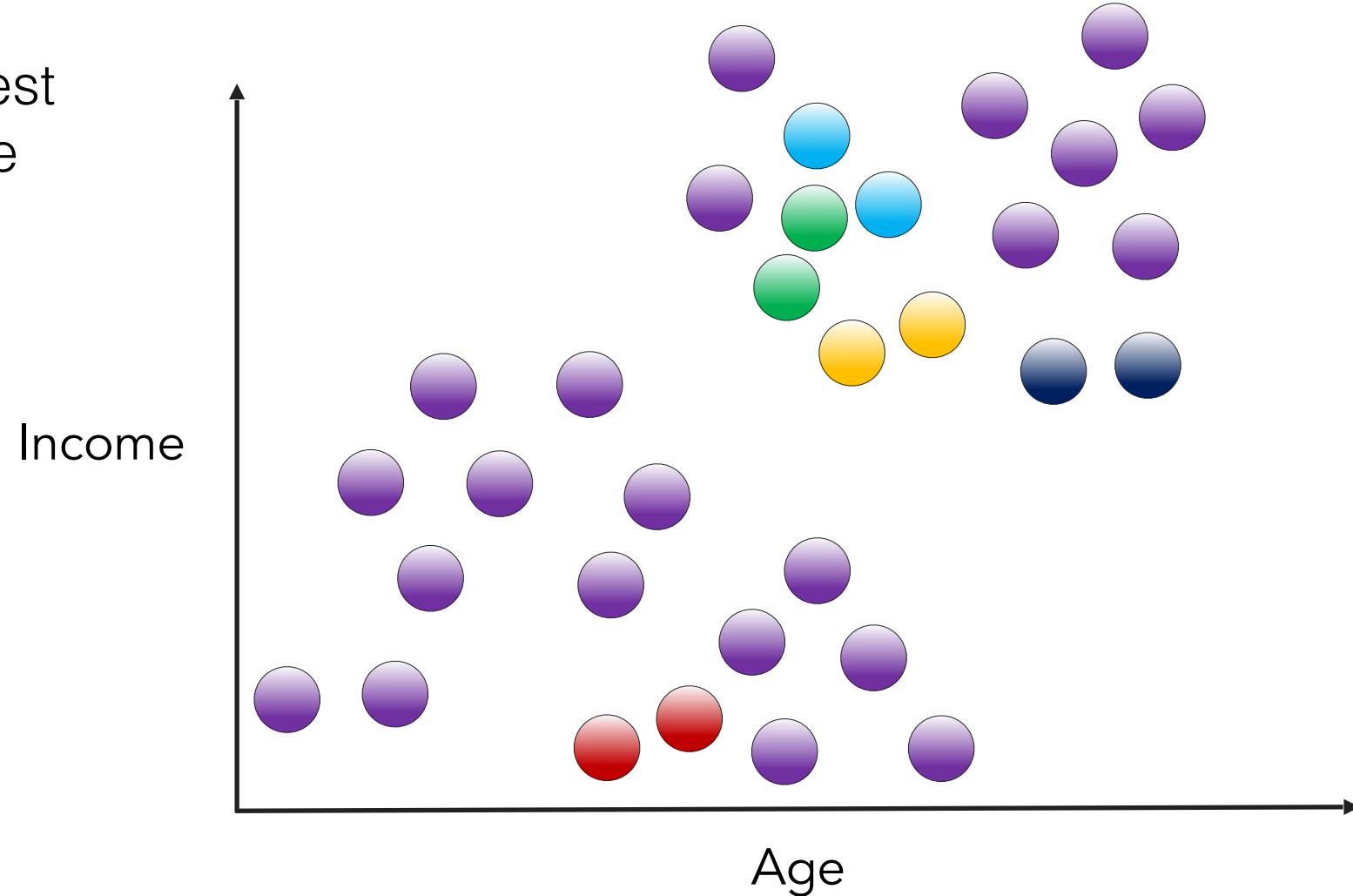
# Clustering

Find next closest pair and merge



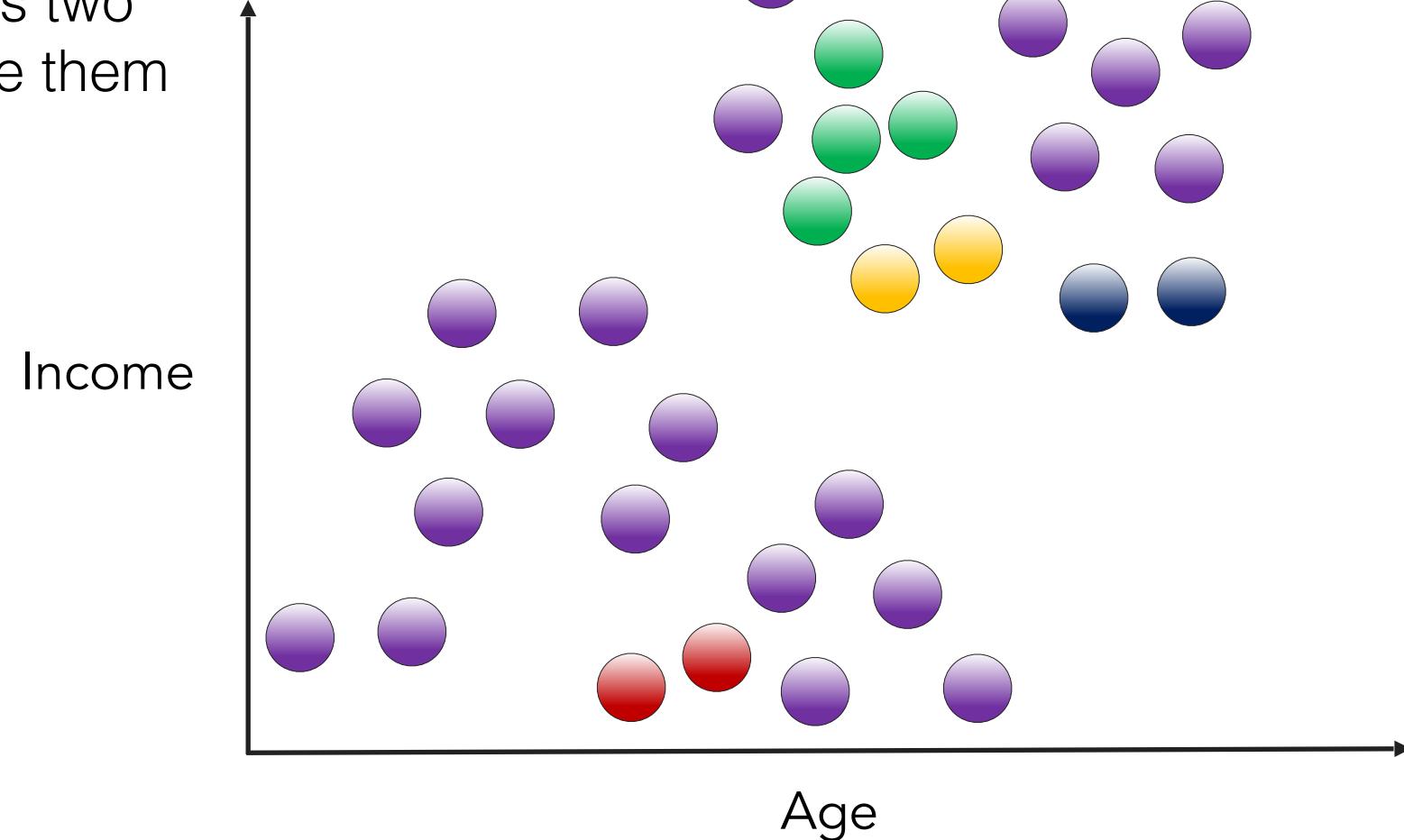
# Clustering

Find next closest pair and merge



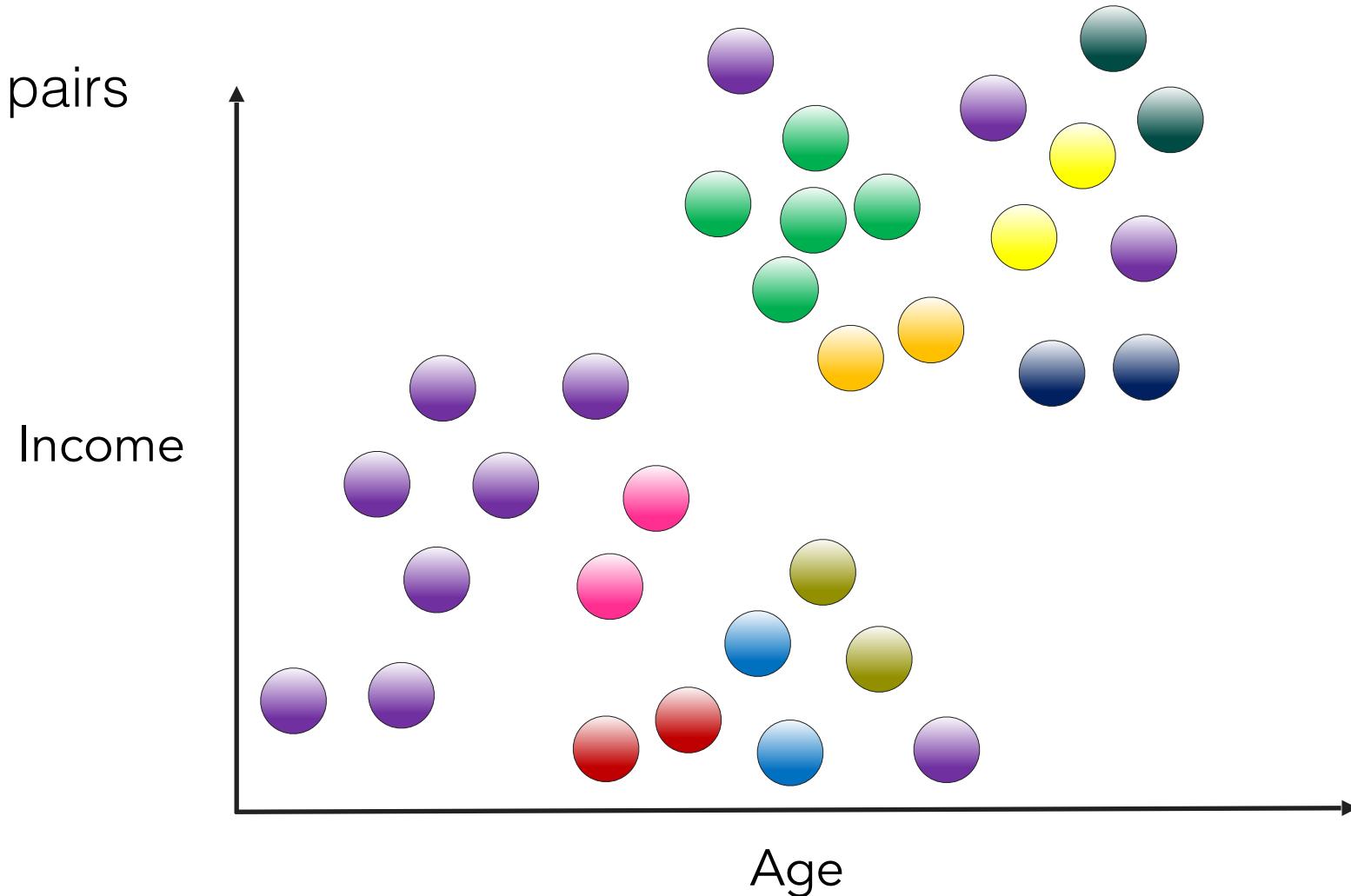
# Clustering

If closest pair is two clusters, merge them



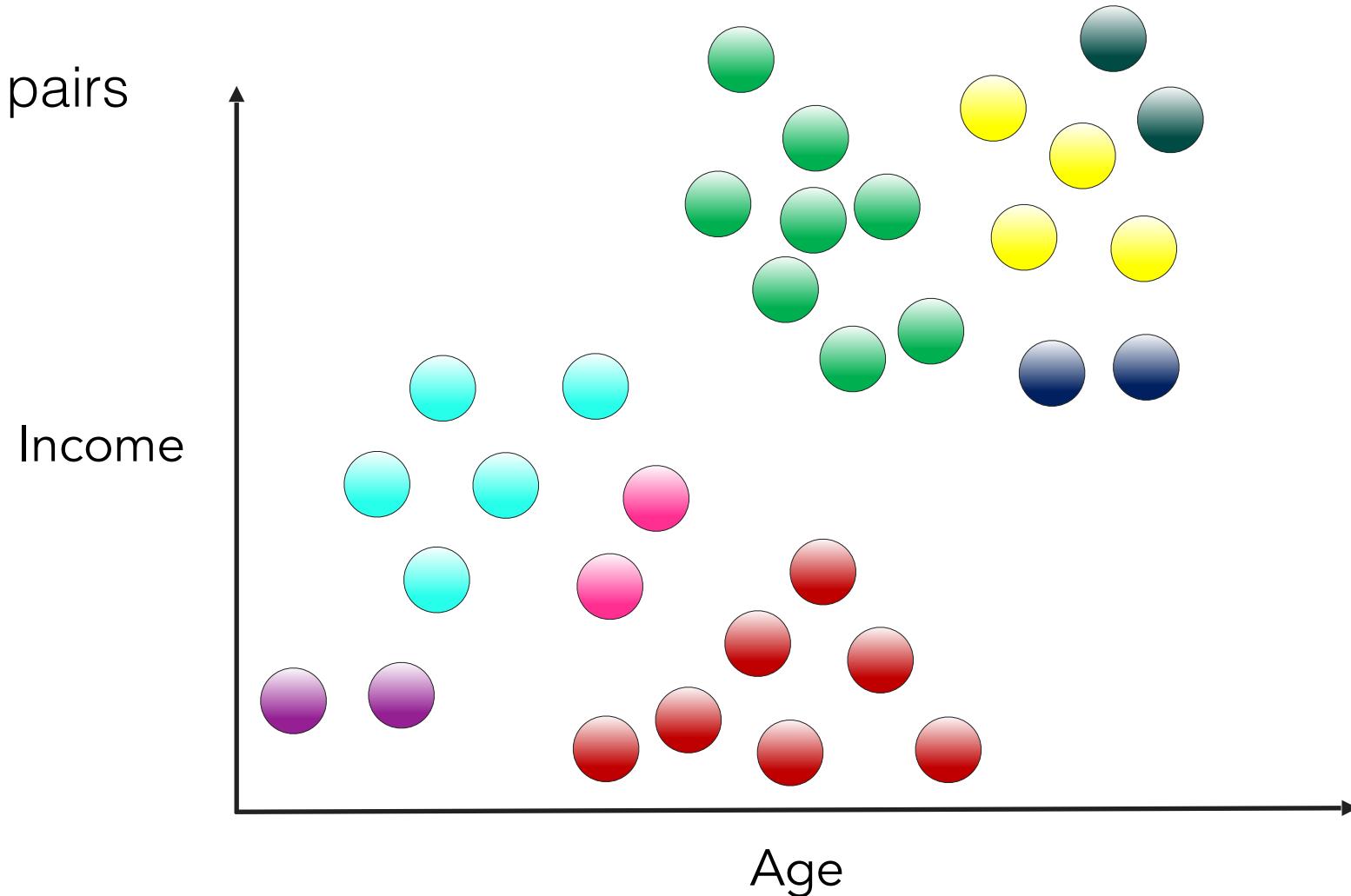
# Clustering

Keep merging pairs  
and clusters



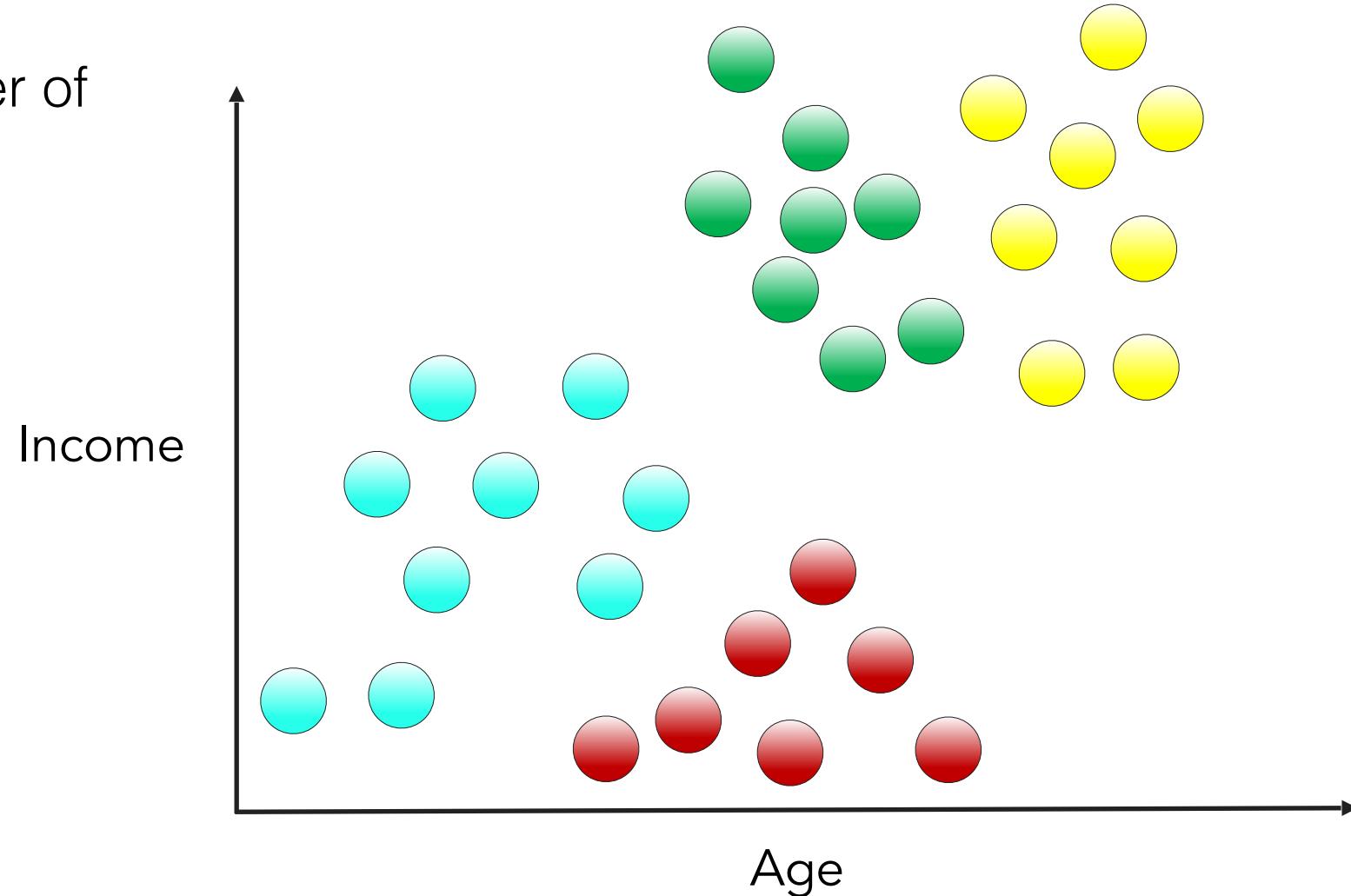
# Clustering

Keep merging pairs  
and clusters



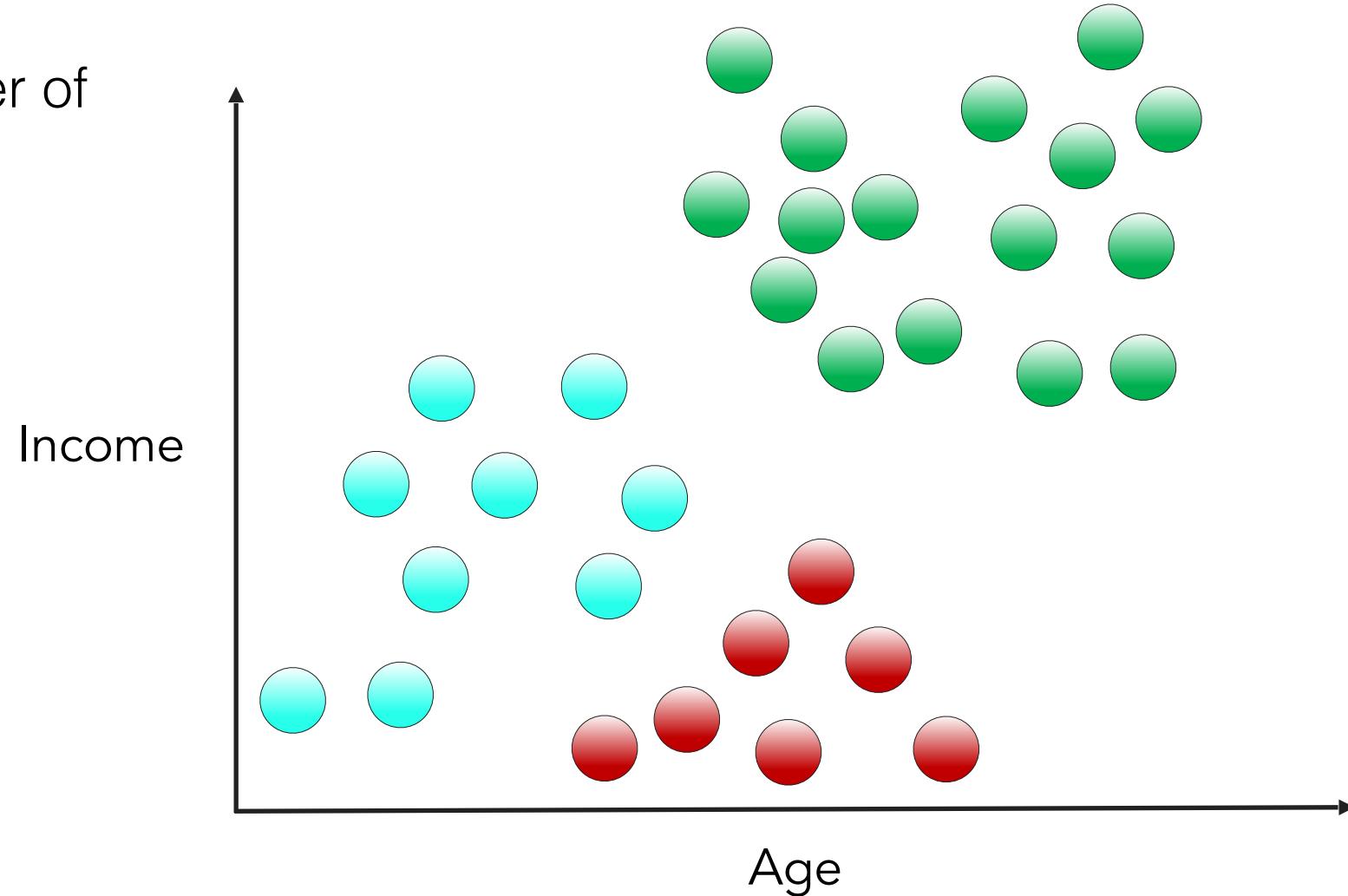
# Clustering

Current number of clusters = 4



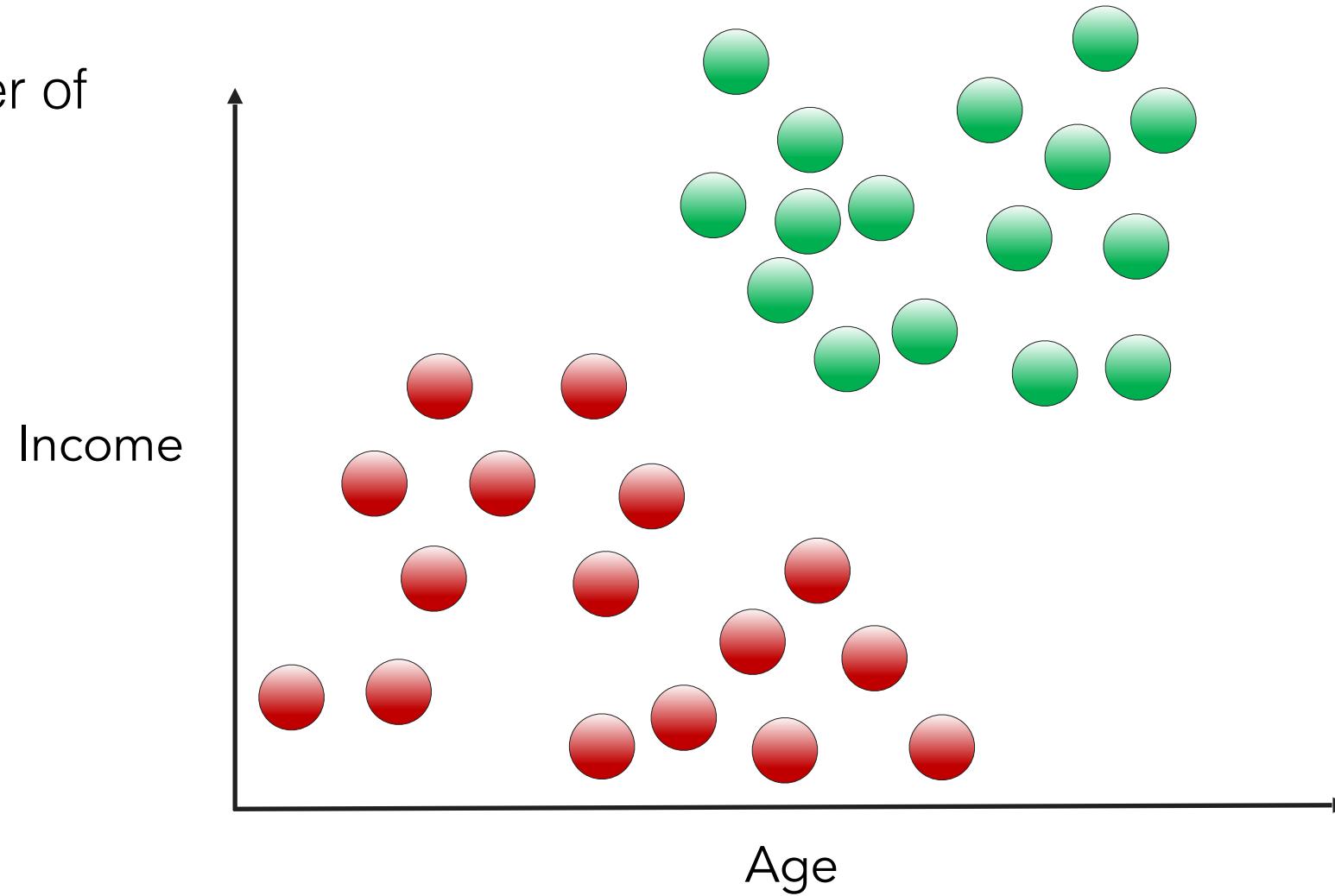
# Clustering

Current number of clusters = 3



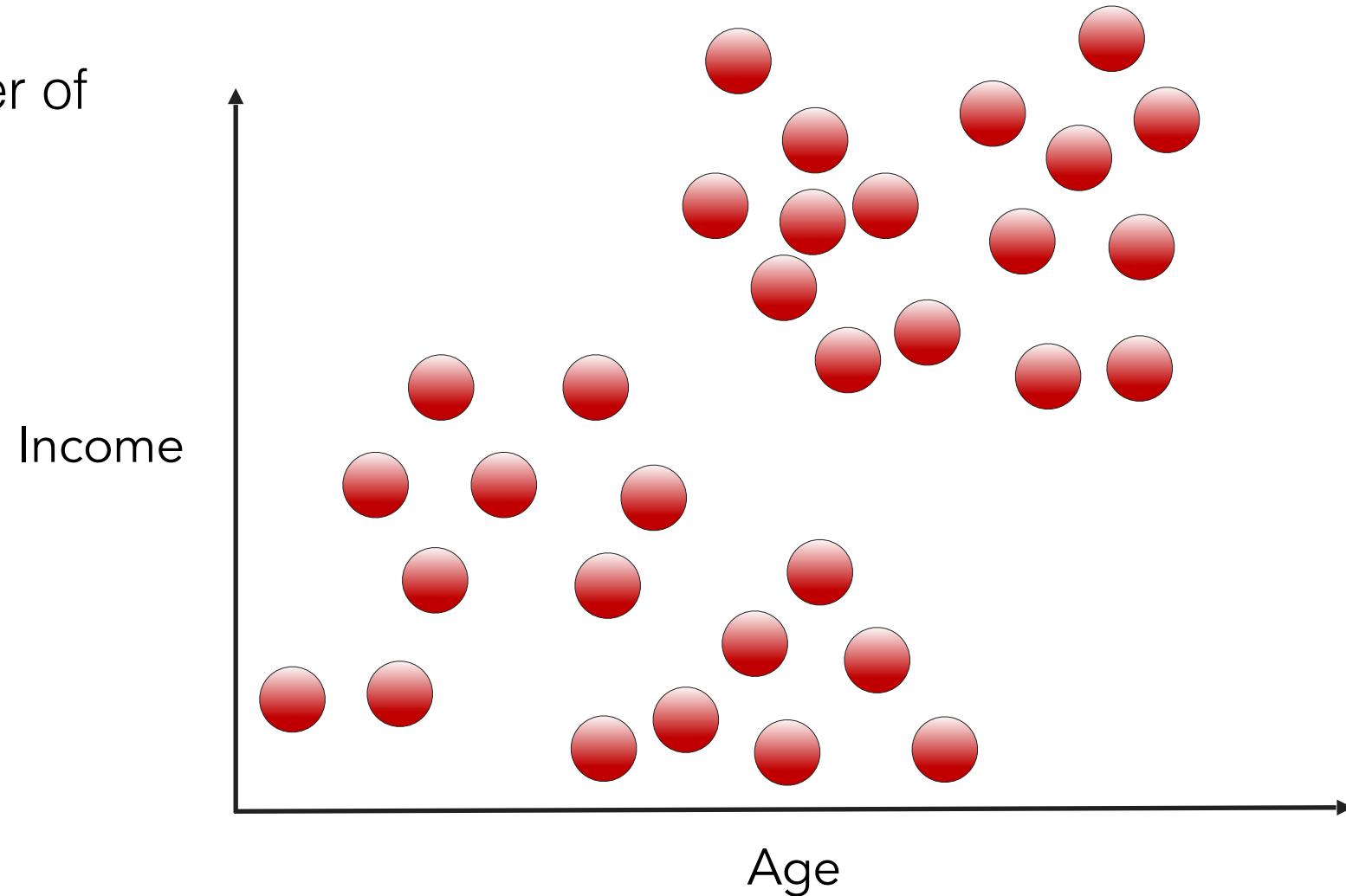
# Clustering

Current number of clusters = 2



# Clustering

Current number of clusters = 1



# When to Stop Clustering?

Condition 1

Correct number of clusters is reached

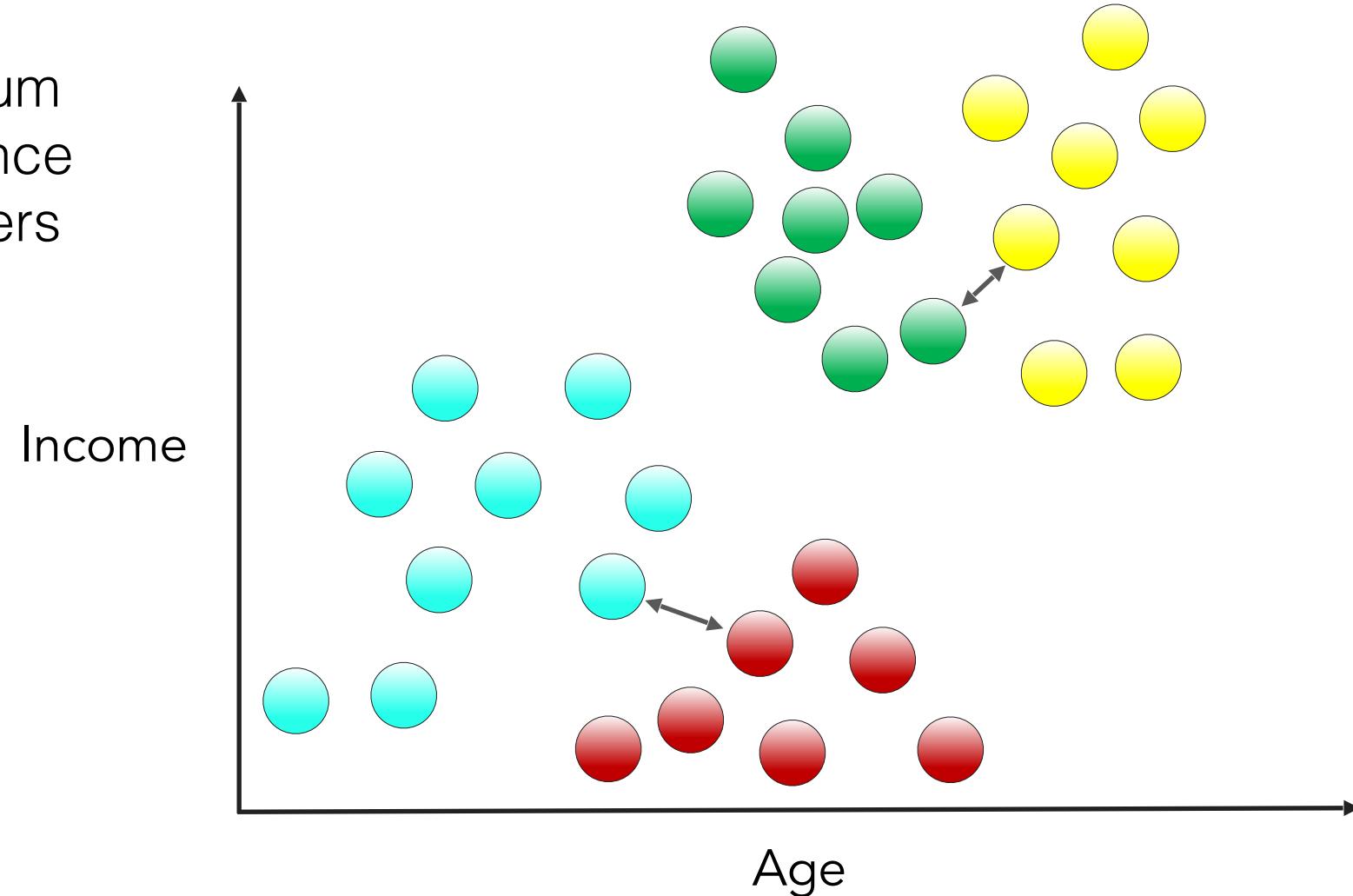
Condition 2

Cluster linkage (distance) reaches a set value



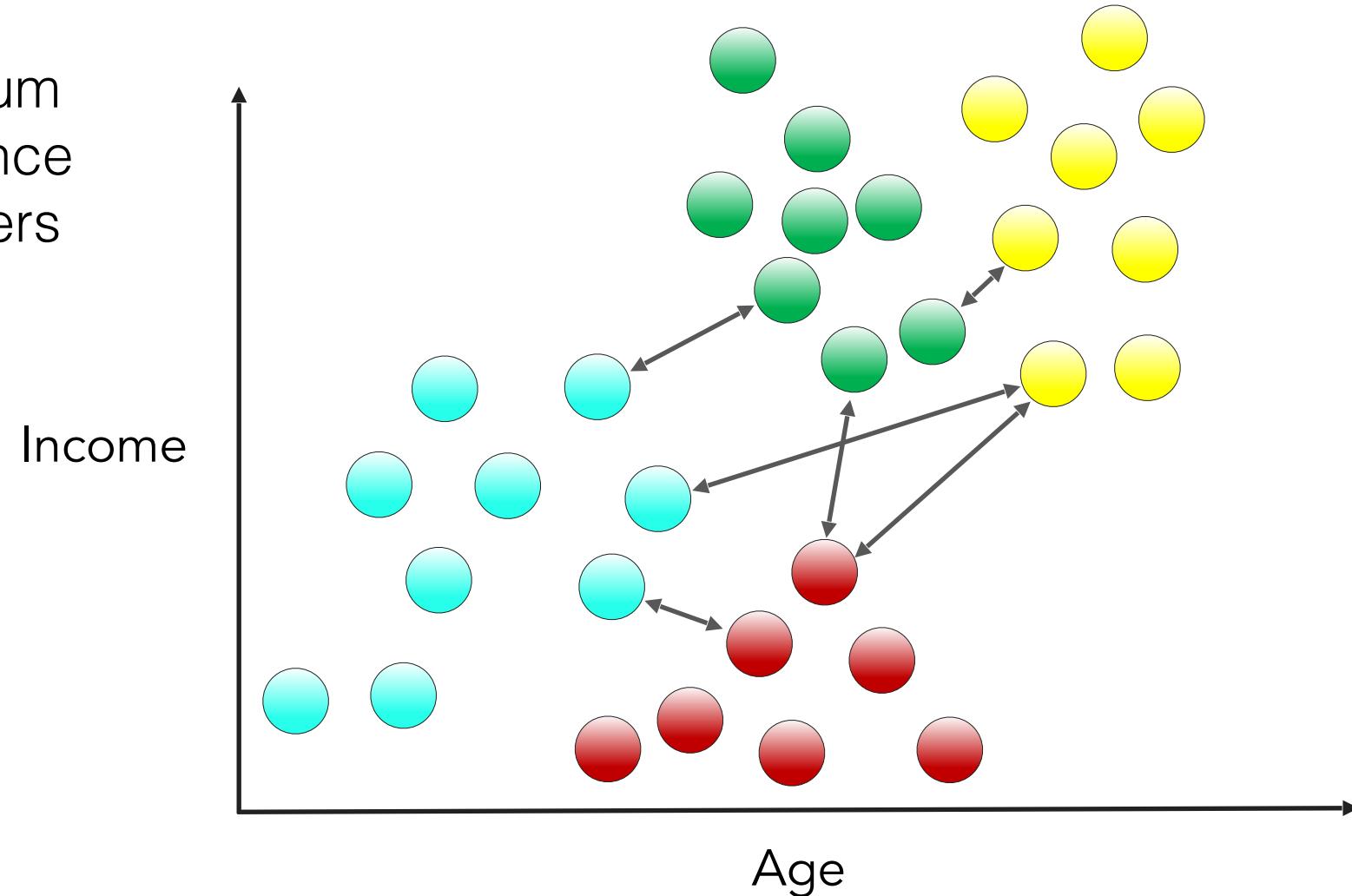
# Cluster Linkage

**Single:** minimum pairwise distance between clusters



# Cluster Linkage

**Single:** minimum pairwise distance between clusters

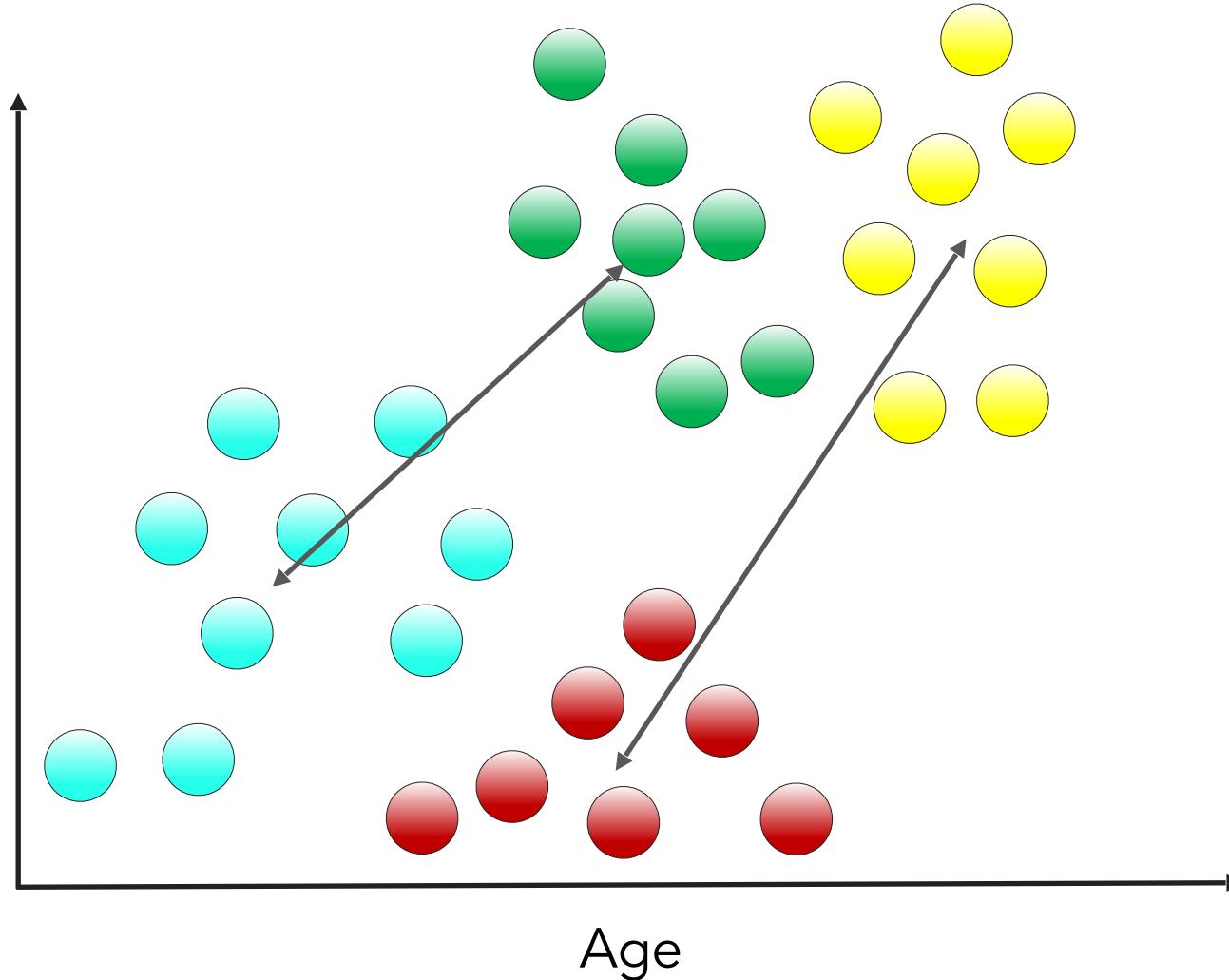


# Cluster Linkage

**Average:** average pairwise distance between clusters

Income

Age

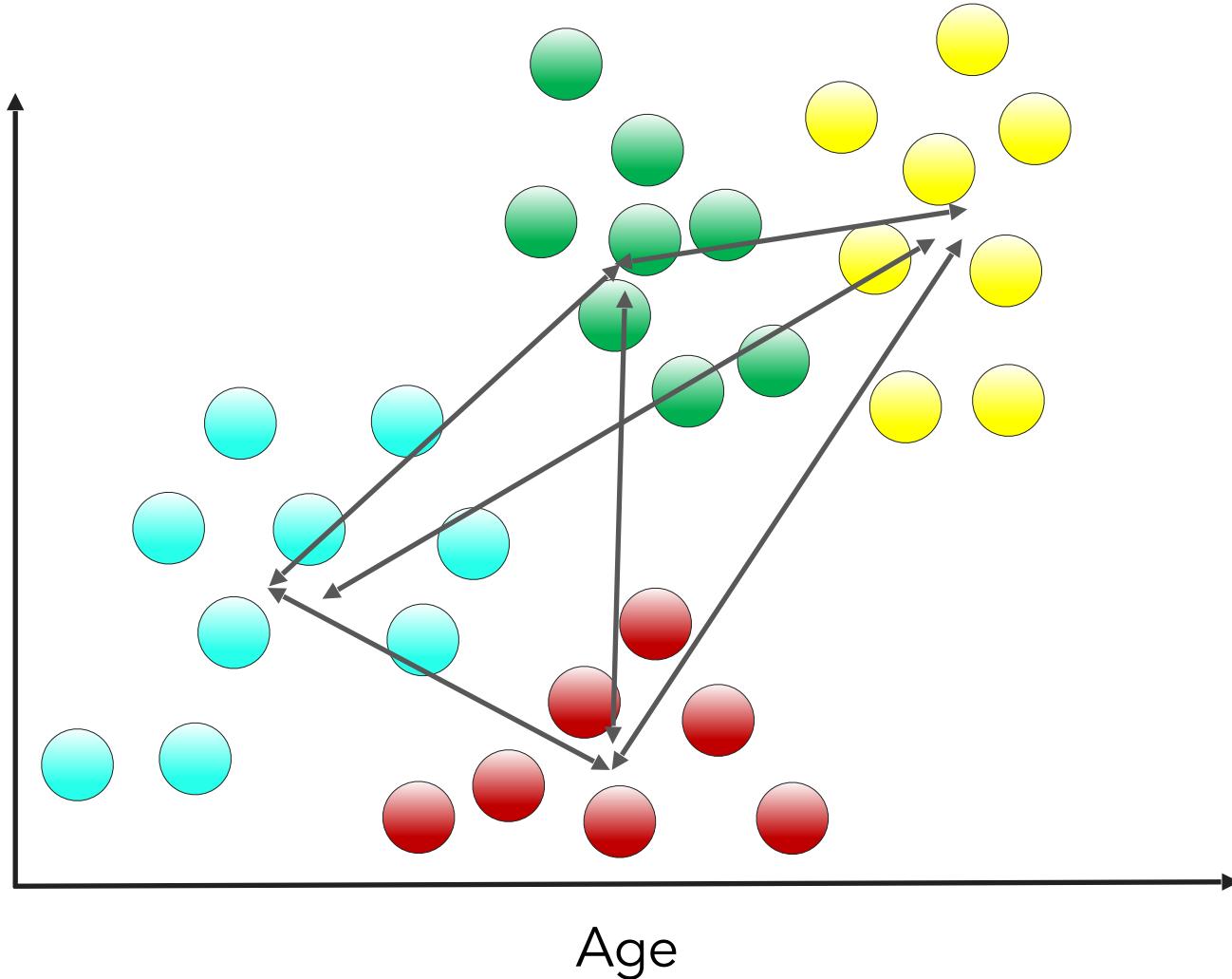


# Cluster Linkage

**Average:** average pairwise distance between clusters

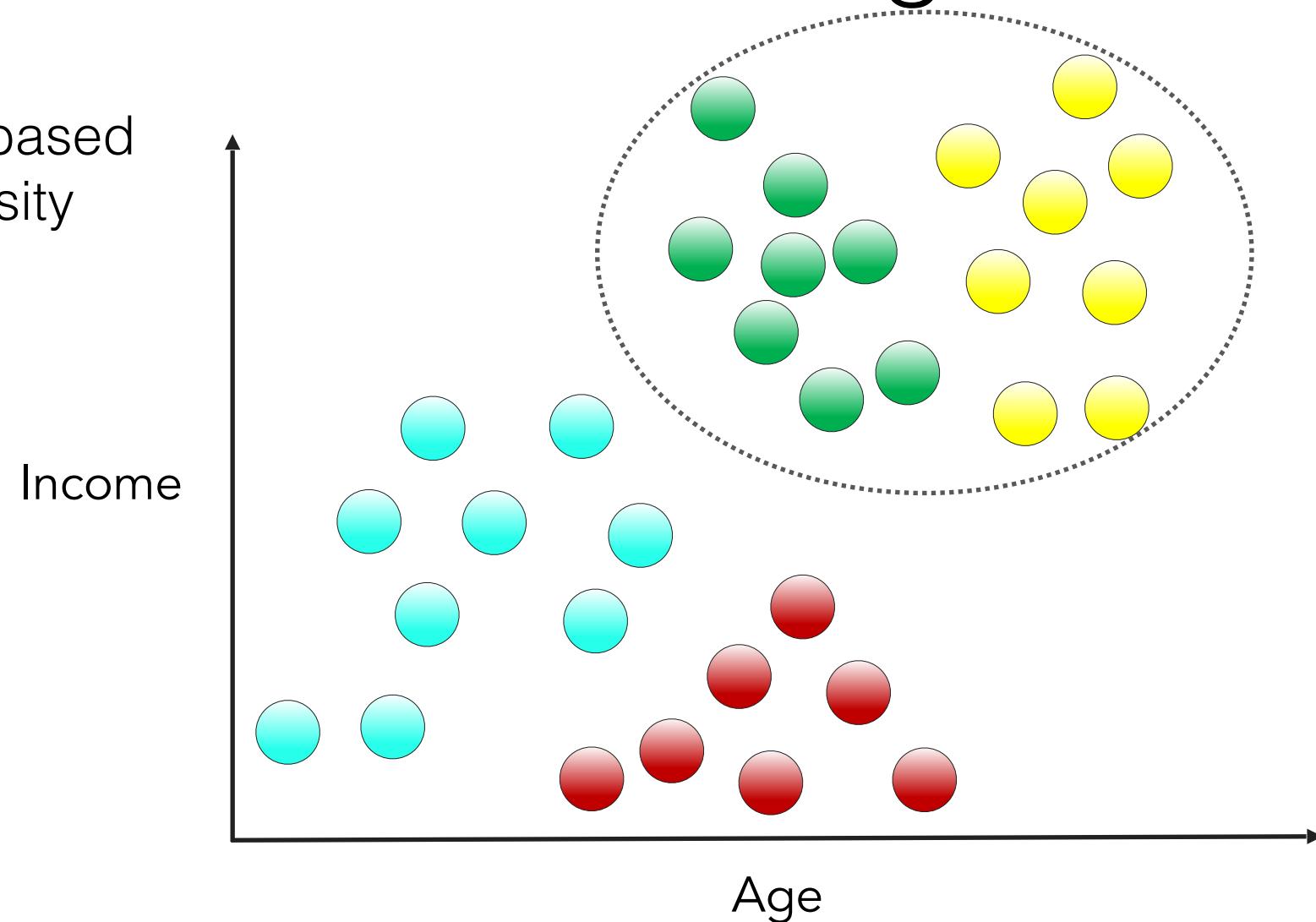
Income

Age

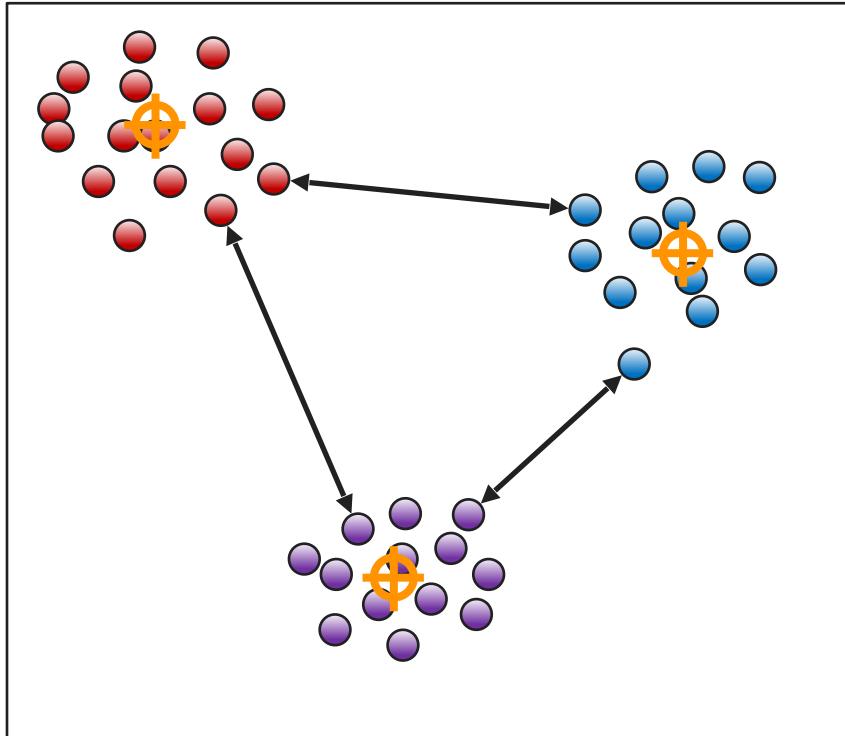


# Cluster Linkage

**Ward:** merge based  
on cluster density  
(inertia)



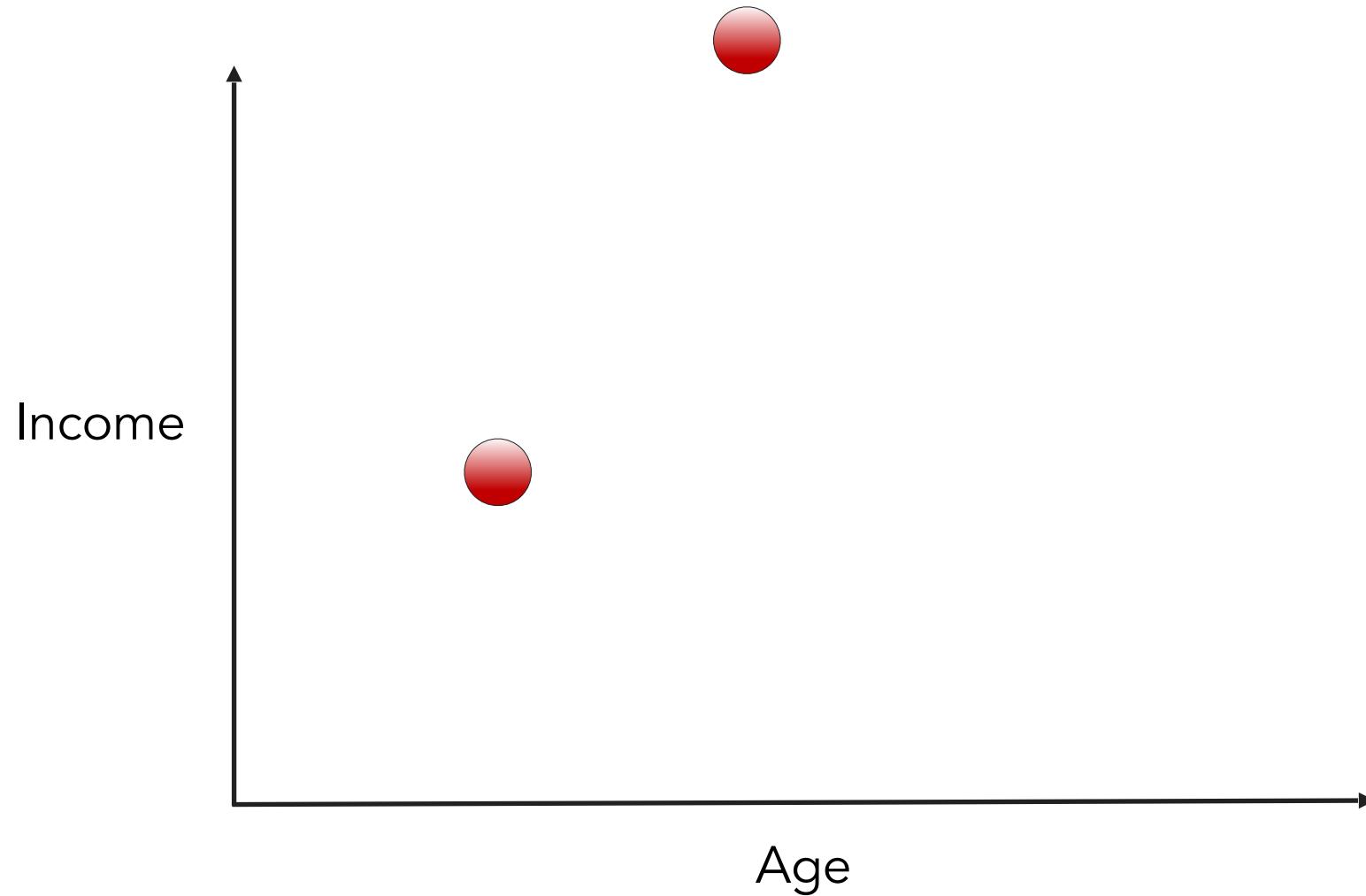
# Distance Metrics



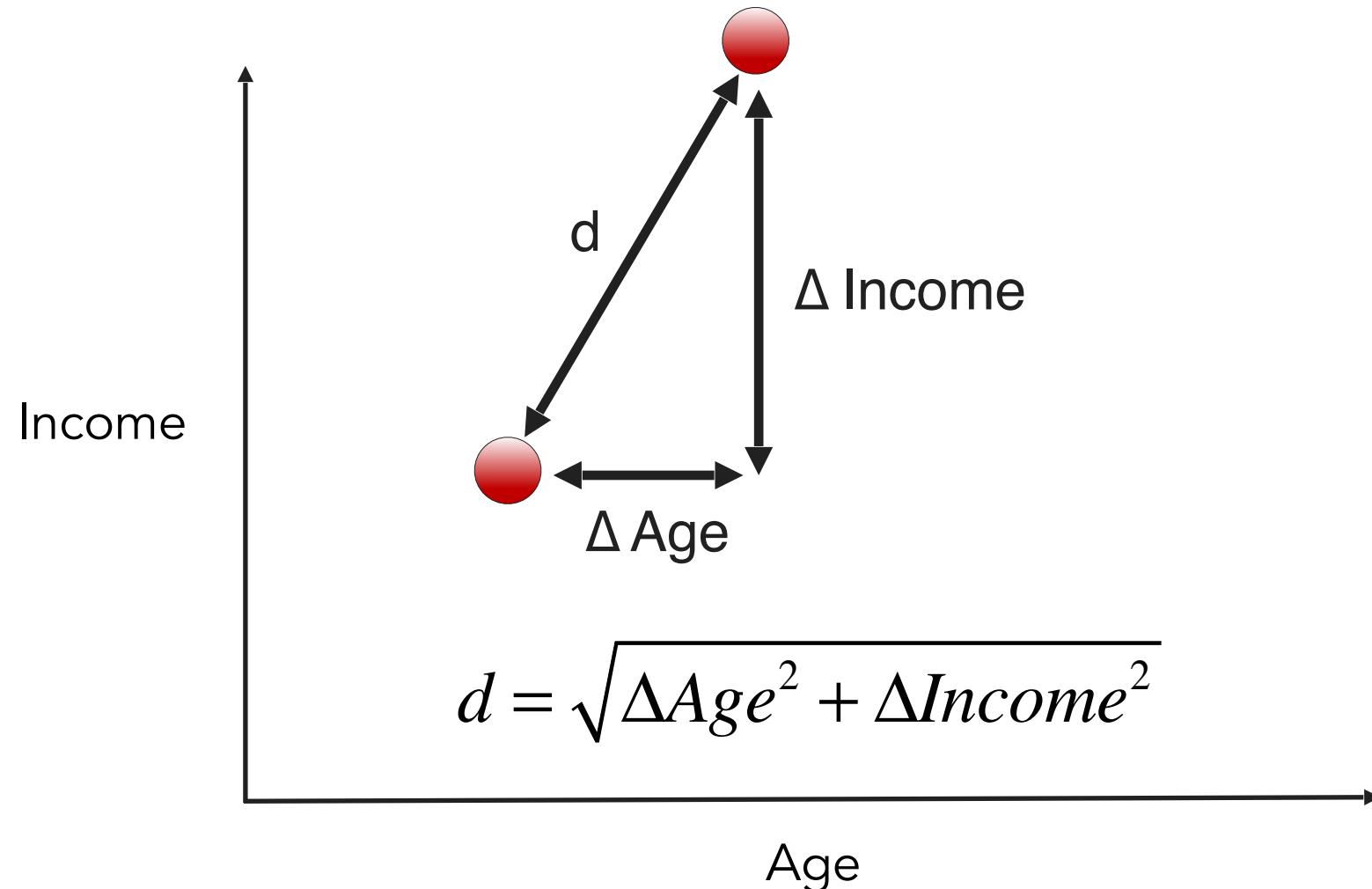
- Clustering requires measuring distance between observations
- Choice of distance metric is extremely important to clustering success



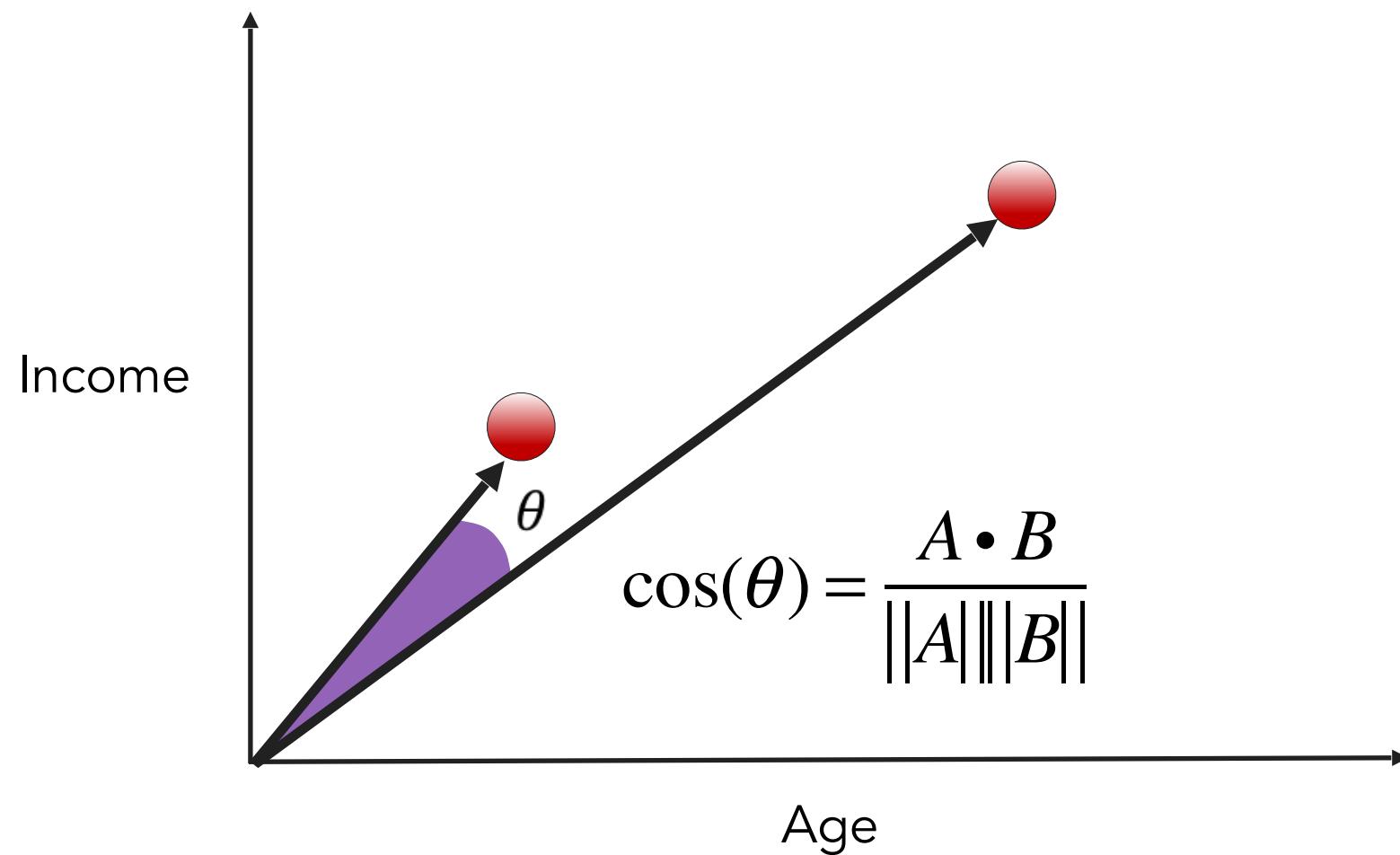
# Distance Metrics



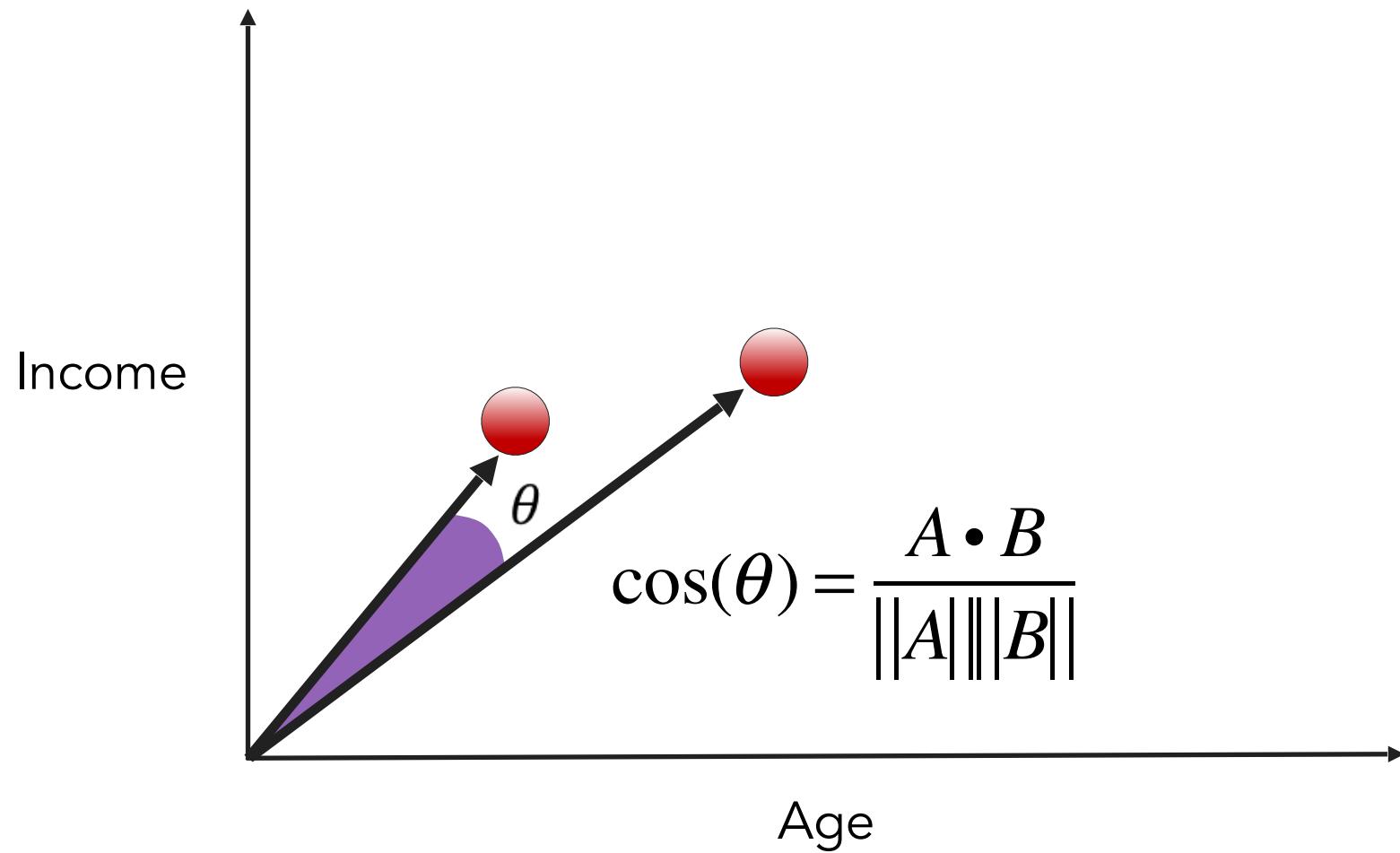
# Euclidean Distance



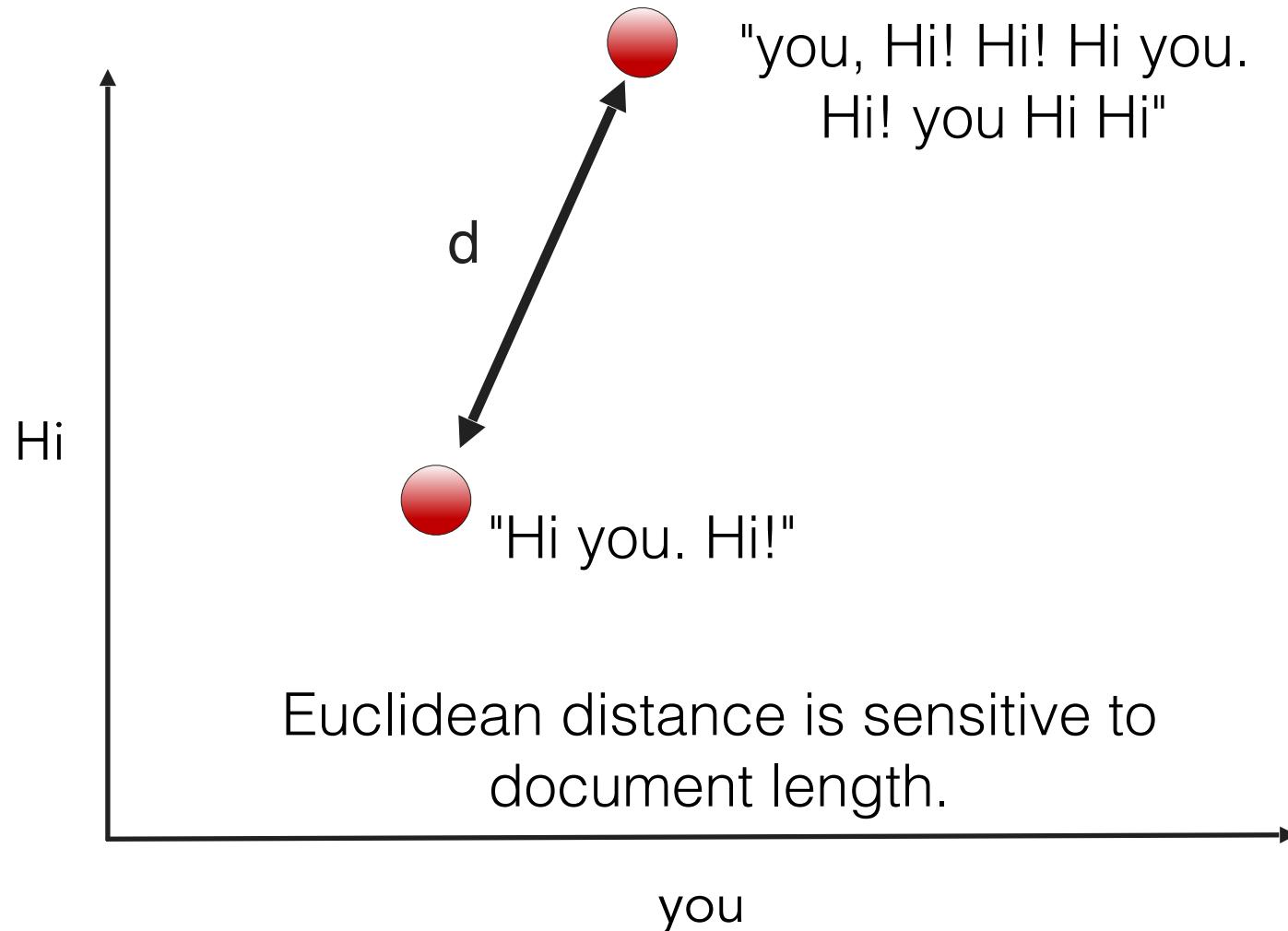
# Cosine Distance



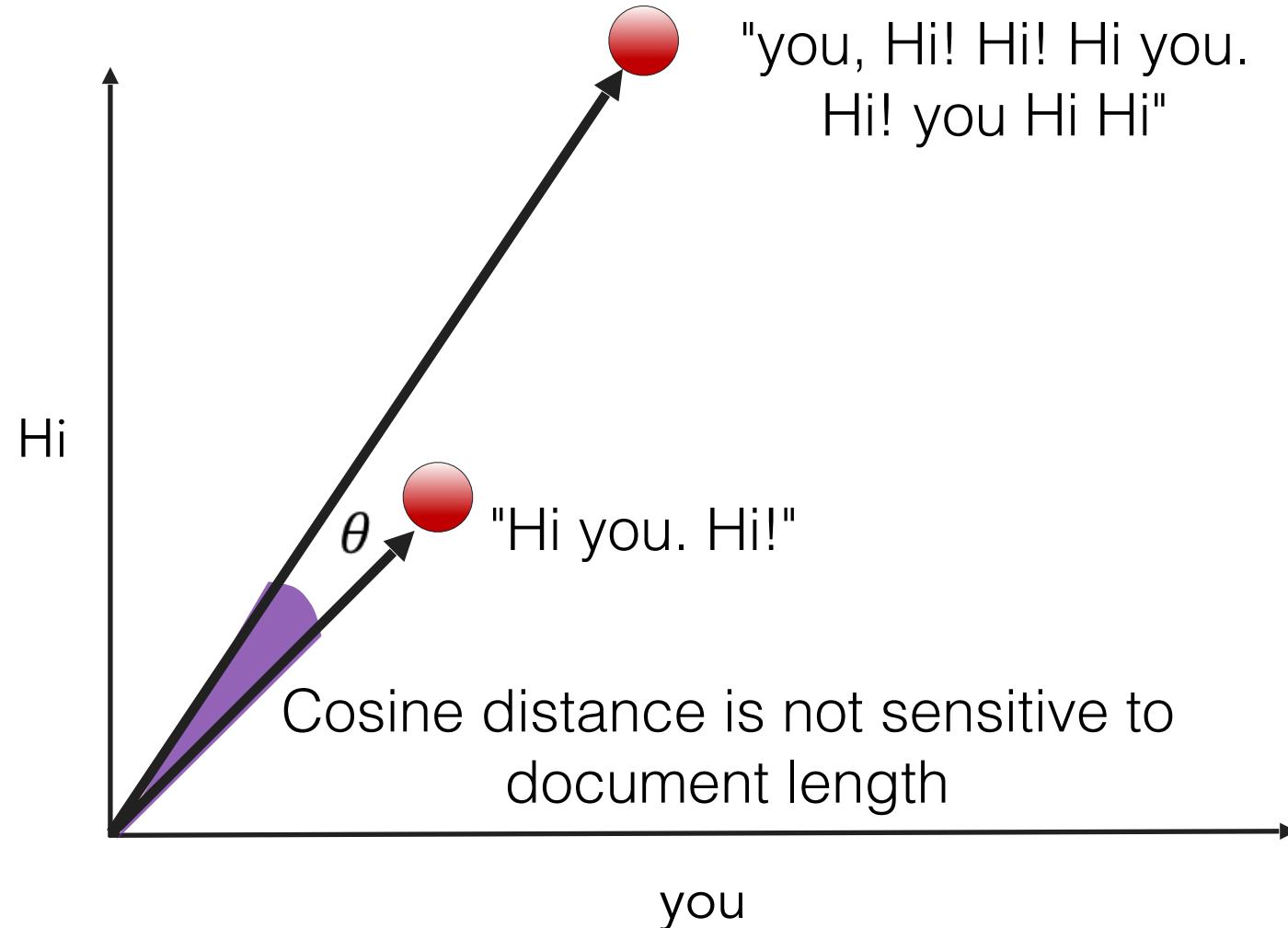
# Cosine Distance



# Distance Metrics for Text



# Distance Metrics for Text



# Practice Clustering

Walkthrough

05\_Hierarchical\_Clustering\_Walkthrough

Exercises

06\_Hierarchical\_Clustering\_Exercises



# Topic Modeling



- Goal: Discover topics in a large collection of documents
- For example, articles arranged by topic on a news site



# Document Structure



- Assume a document is a distribution of topics
- An article can be 30% business, 30% tech, 40% science



# Topic Structure

- Assume a topic is a distribution over words
- Topic 1 is about sports so it uses “game” and “team” frequently
- Topic 2 can also use the words “team” or “game” but less frequently than topic 1

Topic 1		Topic 2		Topic 3	
term	weight	term	weight	term	weight
game	0.014	space	0.021	drive	0.021
team	0.011	nasa	0.006	card	0.015
hockey	0.009	earth	0.006	system	0.013
play	0.008	henry	0.005	scsi	0.012
games	0.007	launch	0.004	hard	0.011



# The Goal

Topics

gene 0.04  
dna 0.02  
genetic 0.01  
...

life 0.02  
evolve 0.01  
organism 0.01  
...

brain 0.04  
neuron 0.02  
nerve 0.01  
...

data 0.02  
number 0.02  
computer 0.01  
...

Documents

## Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,\* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions

\* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

"are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden. "We arrived at the 800 number, but coming up with a consensus answer may be more than just a genetic numbers game; particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing all

genes in common between the two

organisms, he found 233 genes in common.

Genes needed for最基本生存

~22 genes

Rid of redundant genes

~4 genes

Rid of ancient genes

~122 genes

Minimal gene set

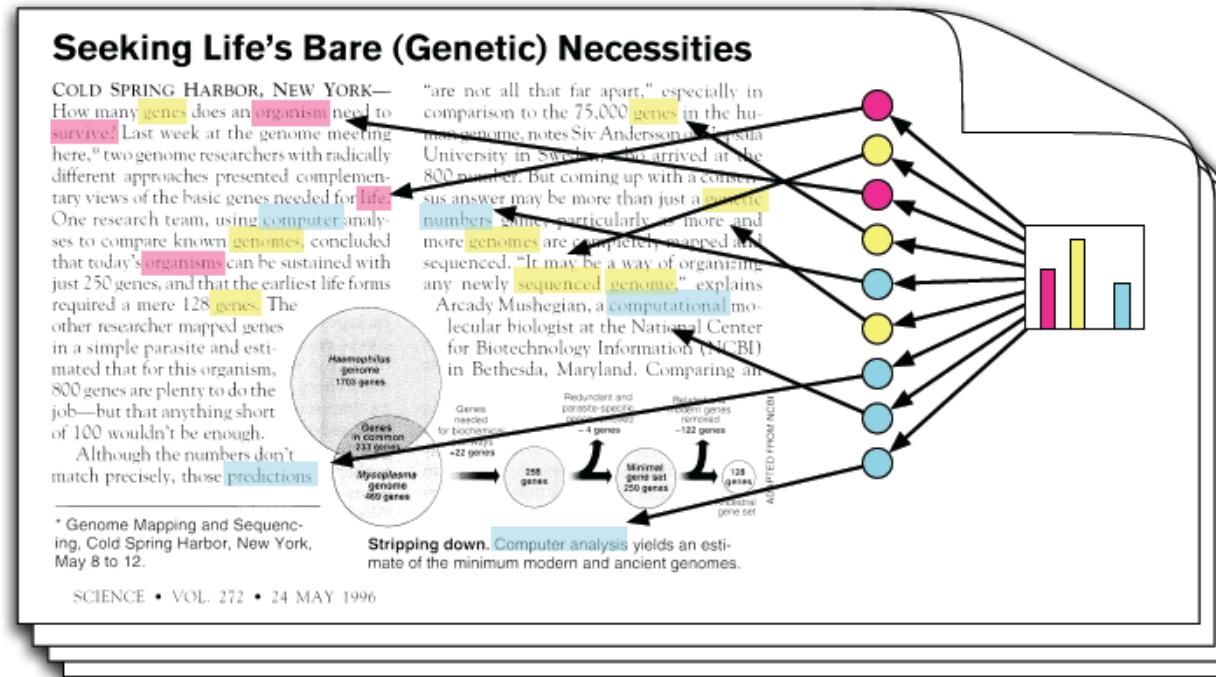
250 genes

Final gene set

128 genes

Applied from NCBI

Topic proportions and assignments



- Find document topic distribution
- Based on topic word distribution



# Latent Dirichlet Allocation (LDA)

- Generative model: models how the data is generated
- Assume a set of topics shared by all documents
- To create a document:
  - Generate a topic from the document topic distribution
  - Generate a word from that topic
- Repeat



# How is LDA Used?

- Assume a set of documents were generated in this way
- Reverse the generative process to understand topics in a given document
- We won't go into the math
- Let's practice in Python



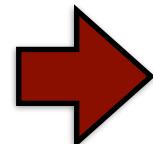
# Practice Topic Modeling

- We will complete the first part of 07\_LDA\_Walkthrough\_and\_Exercises together
- Then answer the remaining questions

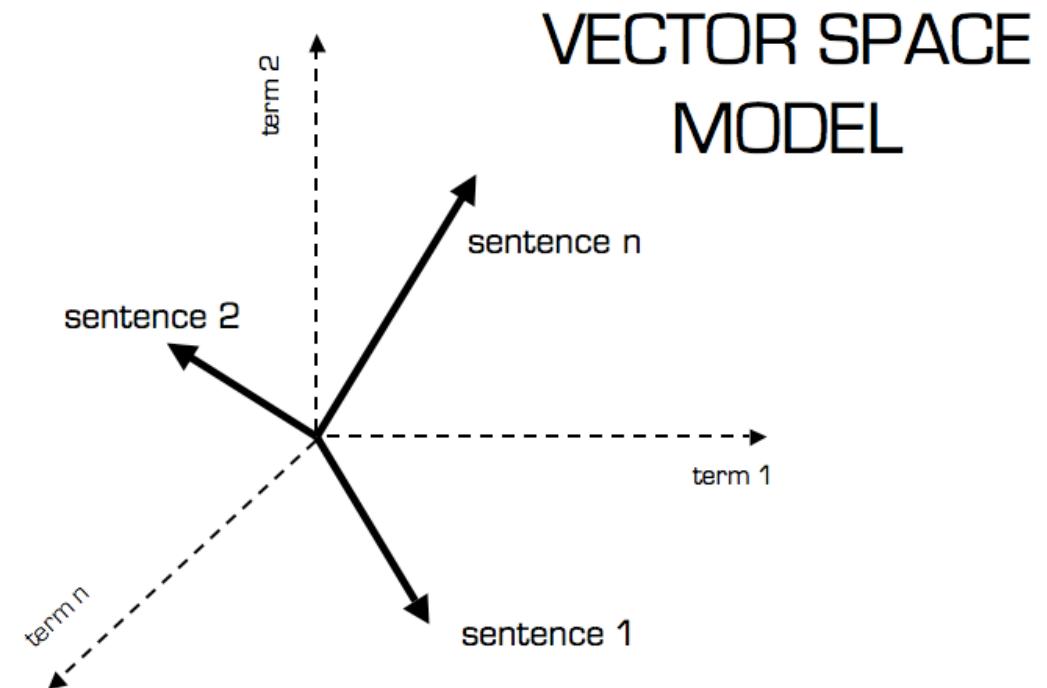


# Deep Learning with Text

- **Problem:** taxonomy count increases dramatically with document size
- **Solution:** use a neural network to create a numerical representation (vector) for each word



Word2Vec



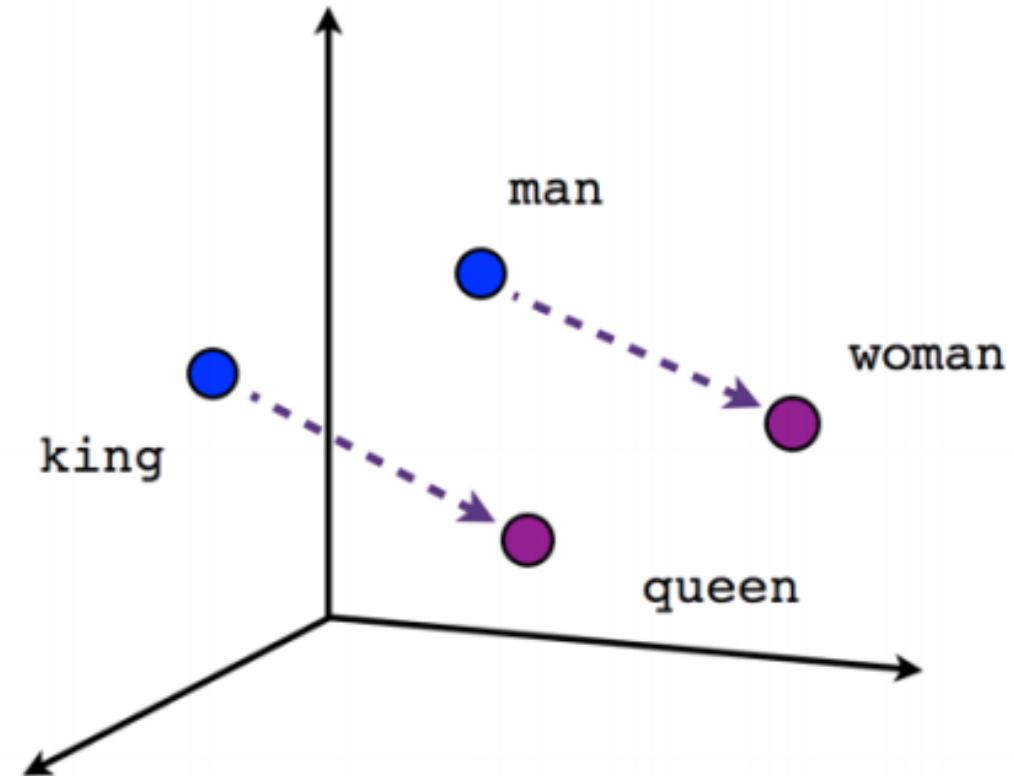
# Word Embeddings with Word2Vec

- Word vectors can be used to understand relationships between words

- For example:

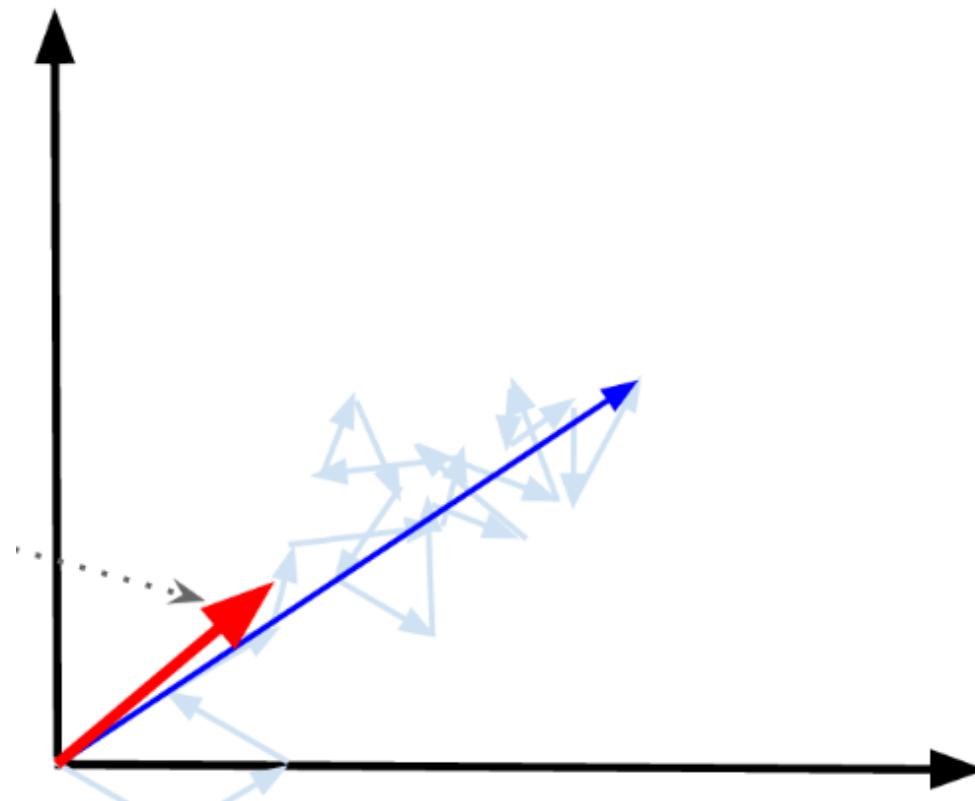
**man --> woman**

**king --> queen**



# How Can We Use Word2Vec?

- Vectors can be combined to create features for documents
- Neural network requires large training set



# Practice Word2Vec

- We will complete the first part of  
08\_Word2Vec\_Walkthrough\_and\_Exercises together
- Then answer the remaining questions



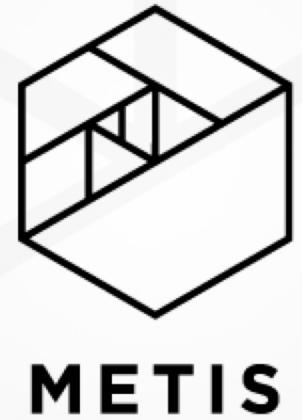
# Thank You

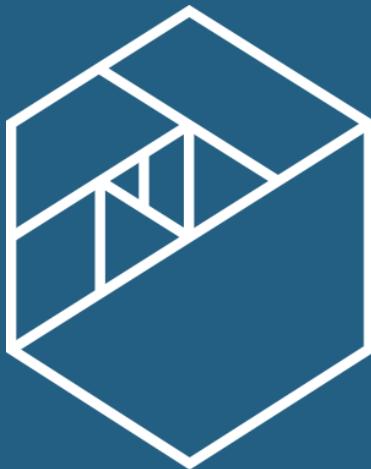
Contacting me:

- michelle@thisismetis.com
- @modernscientist
- [michellelynngill.com](http://michellelynngill.com)
- [linkedin.com/in/michellelynngill](http://linkedin.com/in/michellelynngill)
- [github.com/mlgill](http://github.com/mlgill)

Contacting Metis:

- @thisismetis
- [thisismetis.com](http://thisismetis.com)





METIS